aws

Custom Labels-Leitfaden

Rekognition

Rekognition: Custom Labels-Leitfaden

Table of Contents

Was ist Amazon Rekognition Custom Labels?

Mit Amazon Rekognition Custom Labels können Sie Objekte, Logos und Szenen in Bildern, die für Ihre geschäftlichen Anforderungen charakteristisch sind, für Ihre geschäftlichen Anforderungen charakteristisch sind. Sie können Ihr Logo beispielsweise in Posts in sozialen Netzwerken finden, Ihre Produkte in Ladenregalen identifizieren, Maschinenteile in einer Montagelinie klassifizieren, gesunde und infizierte Pflanzen unterscheiden oder animierte Charaktere in Bildern erkennen.

Die Entwicklung eines benutzerdefinierten Modells zur Analyse von Bildern ist ein bedeutendes Unterfangen, das Zeit, Fachwissen und Ressourcen erfordert. Die Fertigstellung dauert oft Monate. Darüber hinaus können Tausende oder Zehntausende von handbeschrifteten Bildern erforderlich sein, um das Modell mit genügend Daten zu versorgen, um genaue Entscheidungen zu treffen. Die Generierung dieser Daten kann Monate in Anspruch nehmen und große Teams von Etikettierern erfordern, um sie für den Einsatz im maschinellen Lernen vorzubereiten.

Amazon Rekognition Custom Labels erweitert die bestehenden Funktionen von Amazon Rekognition, die bereits an zig Millionen Bildern in vielen Kategorien trainiert wurden. Anstelle von Tausenden von Bildern können Sie einen kleinen Satz von Trainingsbildern (in der Regel einige hundert Bilder oder weniger) hochladen, die für Ihren Anwendungsfall spezifisch sind. Sie können dies mithilfe der easy-to-use Konsole tun. Wenn Ihre Bilder bereits beschriftet sind, kann Amazon Rekognition Custom Labels in kurzer Zeit mit dem Training eines Modells beginnen. Wenn nicht, können Sie die Bilder direkt in der Beschriftungsoberfläche beschriften, oder Sie können Amazon SageMaker Ground Truth verwenden, um sie für Sie zu beschriften.

Nachdem Amazon Rekognition Custom Labels mit dem Training anhand Ihres Bilddatensatzes begonnen hat, kann Amazon Rekognition innerhalb weniger Stunden ein benutzerdefiniertes Bildanalysemodell für Sie erstellen. Hinter den Kulissen lädt und überprüft Amazon Rekognition Custom Labels automatisch die Trainingsdaten, wählt die richtigen Algorithmen für maschinelles Lernen aus, trainiert ein Modell und stellt Leistungskennzahlen für Modelle bereit. Anschließend können Sie Ihr benutzerdefiniertes Modell über die Amazon Rekognition Custom Labels API verwenden und in Ihre Anwendungen integrieren.

Themen

Die wichtigsten Vorteile

Vereinfachte Datenkennzeichnung

Die Amazon Rekognition Custom Labels-Konsole bietet eine visuelle Oberfläche, mit der Sie Ihre Bilder schnell und einfach beschriften können. Über die Benutzeroberfläche können Sie dem gesamten Bild eine Bezeichnung zuweisen. Mithilfe von Begrenzungsfeldern mit einer click-and-drag Schnittstelle können Sie auch bestimmte Objekte in Bildern identifizieren und beschriften. Wenn Sie über einen großen Datensatz verfügen, können Sie alternativ Amazon SageMaker Ground Truth verwenden, um Ihre Bilder effizient und maßstabsgetreu zu beschriften.

Automated machine learning

Für die Erstellung Ihres benutzerdefinierten Modells sind keine Kenntnisse im Bereich maschinelles Lernen erforderlich. Amazon Rekognition Custom Labels umfasst Funktionen für automatisiertes maschinelles Lernen (AutoML), die das maschinelle Lernen für Sie übernehmen. Wenn die Trainingsbilder bereitgestellt werden, kann Amazon Rekognition Custom Labels die Daten automatisch laden und überprüfen, die richtigen Algorithmen für maschinelles Lernen auswählen, ein Modell trainieren und Leistungskennzahlen für Modelle bereitstellen.

Vereinfachte Modellbewertung, Inferenz und Feedback

Sie bewerten die Leistung Ihres benutzerdefinierten Modells auf Ihrem Testset. Für jedes Bild im Testsatz können Sie den side-by-side Vergleich der Vorhersage des Modells mit der zugewiesenen Bezeichnung sehen. Sie können auch detaillierte Leistungskennzahlen wie Präzision, Erinnerungsvermögen, F1-Ergebnisse und Konfidenzwerte überprüfen. Sie können sofort damit beginnen, Ihr Modell für die Bildanalyse zu verwenden, oder Sie können neue Versionen mit mehr Bildern iterieren und neu trainieren, um die Leistung zu verbessern. Nachdem Sie mit der Verwendung Ihres Modells begonnen haben, verfolgen Sie Ihre Vorhersagen, korrigieren etwaige Fehler und verwenden die Feedback-Daten, um neue Modellversionen neu zu trainieren und die Leistung zu verbessern.

Wenn Sie Amazon Rekognition Custom Labels

Amazon Rekognition bietet zwei Funktionen, mit denen Sie Bezeichnungen (Objekte, Szenen und Konzepte) in Bildern finden können: Amazon Rekognition Custom Labels und Amazon Rekognition Image Label Detection. Verwenden Sie die folgenden Informationen, um zu bestimmen, welche Funktion Sie verwenden sollten.

Amazon Rekognition Erkennung von Bildbeschriftungen

Sie können die Labelerkennungsfunktion in Amazon Rekognition Image verwenden, um gängige Bezeichnungen in Bildern und Videos zu identifizieren, zu klassifizieren und nach ihnen zu suchen — in großem Maßstab und ohne ein Modell für maschinelles Lernen erstellen zu müssen. Sie können beispielsweise problemlos Tausende gängiger Objekte wie Autos und Lastwagen, Tomaten, Basketbälle und Fußbälle erkennen.

Wenn Ihre Anwendung gängige Bezeichnungen finden muss, empfehlen wir die Amazon Rekognition Image Label Detection, da Sie kein Modell trainieren müssen. Eine Liste der Labels, die Amazon Rekognition Image Label Detection findet, finden Sie unter Labels erkennen.

Wenn Ihre Anwendung Labels finden muss, die von Amazon Rekognition Image Labels nicht gefunden wurden, wie z. B. kundenspezifische Maschinenteile an einer Montagelinie, empfehlen wir Ihnen, Amazon Rekognition Custom Labels zu verwenden.

Amazon Rekognition Custom Labels

Sie können Amazon Rekognition Custom Labels verwenden, um auf einfache Weise ein Modell für maschinelles Lernen zu trainieren, das Bezeichnungen (Objekte, Logos, Szenen und Konzepte) in Bildern findet, die speziell auf Ihre Geschäftsanforderungen zugeschnitten sind.

Amazon Rekognition Custom Labels können Bilder klassifizieren (Vorhersagen auf Bildebene) oder Objektpositionen in einem Bild erkennen (Vorhersagen auf Objekt-/Bounding-Bounding-Box-Ebene).

Amazon Rekognition Custom Labels bietet mehr Flexibilität bei den Arten von Objekten und Szenen, die Sie erkennen können. Sie können beispielsweise die Amazon Rekognition Image Labelerkennung verwenden, um Pflanzen und Blätter zu finden. Um zwischen gesunden, geschädigten und infizierten Pflanzen zu unterscheiden, müssen Sie Amazon Rekognition Custom Labels verwenden.

Die folgenden Beispiele

- Identifizieren Sie Teamlogos auf Spielertrikots und Helmen
- Unterscheiden Sie zwischen bestimmten Maschinenteilen oder Produkten an einer Montagelinie
- Identifizieren Sie Zeichentrickfiguren in einer Mediathek
- Finden Sie Produkte einer bestimmten Marke in den Verkaufsregalen
- Klassifizieren Sie die Qualität landwirtschaftlicher Produkte (z. B. verfault, reif oder roh)

> **Note**
>
> Amazon Rekognition Custom Labels ist nicht für die Analyse von Gesichtern, die Erkennung von Text oder das Auffinden unsicherer Bildinhalte in Bildern konzipiert. Um diese Aufgaben auszuführen, können Sie Amazon Rekognition Image verwenden. Weitere Amazon Rekognition

Verwenden Sie Amazon Rekognition Custom Labels zum ersten Mal?

Wenn Sie Amazon Rekognition Custom Labels erstmals

1. Einrichten von Amazon Rekognition Custom Labels— In diesem Abschnitt legen Sie Ihre Kontodaten fest.
2. Grundlegendes zu Amazon Rekognition Custom Labels— In diesem Abschnitt erfahren Sie mehr über den Arbeitsablauf zum Erstellen eines Modells.
3. Erste Schritte mit Amazon Rekognition Custom Labels— In diesem Abschnitt trainieren Sie ein Modell anhand von Beispielprojekten, die von Amazon Rekognition Custom Labels erstellt wurden.
4. Tutorial: Bilder klassifizieren— In diesem Abschnitt lernen Sie, wie Sie ein Modell trainieren, das Bilder anhand von Datensätzen, die Sie erstellen, klassifiziert.

Einrichten von Amazon Rekognition Custom Labels

Die folgenden Anweisungen verdeutlichen, wie Sie die Amazon Rekognition Custom Labels-Konsole und das SDK einrichten.

Beachten Sie, dass Sie die Amazon Rekognition Custom Labels-Konsole mit den folgenden Browsern verwenden können:

- Chrome — Version 21 oder höher
- Firefox — Version 27 oder höher
- Microsoft Edge — Version 88 oder höher
- Safari — Version 7 oder höher Außerdem können Sie Safari nicht verwenden, um mit der Amazon Rekognition Custom Labels-Konsole Begrenzungsrahmen zu zeichnen. Weitere Informationen finden Sie unter Objekte mit Begrenzungsrahmen mit Labels versehen.

Bevor Sie Amazon Rekognition Custom Labels zum ersten Mal verwenden, führen Sie die folgenden Schritte aus:

Themen

Schritt 1: Erstellen Sie ein AWS Konto

In diesem Schritt erstellen Sie ein AWS Konto und einen Administratorbenutzer und erfahren, wie Sie programmatischen Zugriff auf das AWS SDK gewähren.

Themen

- Programmgesteuerter Zugriff

Melden Sie sich an für ein AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Aus Sicherheitsgründen sollten Sie einem Benutzer Administratorzugriff zuweisen und nur den Root-Benutzer verwenden, um Aufgaben auszuführen, für die Root-Benutzerzugriff erforderlich ist.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen Sie einen Benutzer mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

Anweisungen finden Sie unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto Root-Benutzer (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie einen Benutzer mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.
2. Gewähren Sie einem Benutzer in IAM Identity Center Administratorzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit der Standardeinstellung konfigurieren IAM-Identity-Center-Verzeichnis im AWS IAM Identity Center Benutzerhandbuch.

Melden Sie sich als Benutzer mit Administratorzugriff an

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Zugriffsportal.

Weisen Sie weiteren Benutzern Zugriff zu

1. Erstellen Sie in IAM Identity Center einen Berechtigungssatz, der der bewährten Methode zur Anwendung von Berechtigungen mit den geringsten Rechten folgt.

   Anweisungen finden Sie im Benutzerhandbuch unter Einen Berechtigungssatz erstellen.AWS IAM Identity Center
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Anweisungen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter Gruppen hinzufügen.

Programmgesteuerter Zugriff

Benutzer benötigen programmatischen Zugriff, wenn sie mit AWS außerhalb des interagieren möchten. AWS Management Console Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt vom Benutzertyp ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zu AWS IAM Identity Center verwendenden im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs, Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch für AWS SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Folgen Sie den Anweisungen unter Verwenden temporäre r Anmeldeinformationen mit AWS Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
	programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	• Informationen dazu finden Sie unter Authentifizierung mithilfe von IAM-Benutzeranmeldedaten im Benutzerhandbuch. AWS CLIAWS Command Line Interface • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch für AWS SDKs und Tools. • Informationen zu AWS APIs finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

Schritt 2: Einrichten von Amazon Rekognition Custom Labels-Konsolenberechtigungen

Um die Amazon Rekognition-Konsole verwenden zu können, müssen Sie über die entsprechenden Berechtigungen verfügen. Wenn Sie Ihre Trainingsdateien in einem anderen Bucket als dem Konsolen-Bucket speichern möchten, benötigen Sie zusätzliche Berechtigungen.

Themen

- Erlauben von Konsolenzugriff
- Zugreifen auf externe Amazon-S3-Buckets
- Zuweisen von Berechtigungen

Erlauben von Konsolenzugriff

Um die Amazon Rekognition Custom Labels-Konsole verwenden zu können, benötigen Sie die folgende IAM-Richtlinie, die Amazon S3, SageMaker Ground Truth und Amazon Rekognition Custom Labels abdeckt. Weitere Informationen zum Zuweisen von Berechtigungen finden Sie unter Zuweisen von Berechtigungen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "s3Policies",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:CreateBucket",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
                "s3:GetObjectTagging",
                "s3:GetBucketVersioning",
                "s3:GetObjectVersionTagging",
                "s3:PutBucketCORS",
                "s3:PutLifecycleConfiguration",
                "s3:PutBucketPolicy",
                "s3:PutObject",
                "s3:PutObjectTagging",
                "s3:PutBucketVersioning",
                "s3:PutObjectVersionTagging"
            ],
            "Resource": [
                "arn:aws:s3:::custom-labels-console-*"
```

```
            ]
        },
        {
            "Sid": "rekognitionPolicies",
            "Effect": "Allow",
            "Action": [
                "rekognition:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "groundTruthPolicies",
            "Effect": "Allow",
            "Action": [
                "groundtruthlabeling:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Zugreifen auf externe Amazon-S3-Buckets

Wenn Sie die Amazon Rekognition Custom Labels-Konsole zum ersten Mal in einer neuen AWS Region öffnen, erstellt Amazon Rekognition Custom Labels einen Bucket (Konsolen-Bucket), der zum Speichern von Projektdateien verwendet wird. Alternativ können Sie Ihren eigenen Amazon-S3-Bucket (externer Bucket) verwenden, um die Bilder oder die Manifestdatei auf die Konsole hochzuladen. Um einen externen Bucket zu verwenden, fügen Sie der vorherigen Richtlinie den folgenden Richtlinienblock hinzu. Ersetzen Sie `my-bucket` durch den Namen Ihres Buckets.

```
        {
            "Sid": "s3ExternalBucketPolicies",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectVersion",
                "s3:GetObjectTagging",
                "s3:ListBucket",
                "s3:PutObject"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket*"
            ]
        }
```

Zuweisen von Berechtigungen

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in: AWS IAM Identity Center

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.
- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

Schritt 3: Erstellen Sie den Konsolen-Bucket.

Sie verwenden ein Projekt in Amazon Rekognition Custom Labels, um Ihre Modelle zu erstellen und zu verwalten. Wenn Sie die Amazon Rekognition Custom Labels-Konsole zum ersten Mal in einer neuen AWS Region öffnen, erstellt Amazon Rekognition Custom Labels einen Amazon S3 S3-Bucket (Konsolen-Bucket) zum Speichern Ihrer Projekte. Sie sollten sich den Namen des Konsolen-Buckets an einer Stelle notieren, wo Sie später darauf zurückgreifen können, da Sie den Bucket-Namen möglicherweise bei AWS SDK-Vorgängen oder Konsolenaufgaben wie der Erstellung eines Datensatzes verwenden müssen.

Das Format des Bucket-Namens ist `custom-labels-console-<region>-<random value>`. Der Zufallswert stellt sicher, dass es nicht zu einer Kollision zwischen Bucket-Namen kommt.

So erstellen Sie den Konsolen-Bucket

1. Stellen Sie sicher, dass der Benutzer über die richtigen Berechtigungen verfügt. Weitere Informationen finden Sie unter Erlauben von Konsolenzugriff.
2. Melden Sie sich bei der Amazon Rekognition Rekognition-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/rekognition/.
3. Wählen Sie Erste Schritte.
4. Wenn Sie die Konsole in der aktuellen AWS-Region zum ersten Mal öffnen, gehen Sie im Dialogfeld Erstmalige Einrichtung wie folgt vor:
   a. Kopieren Sie den Namen des Amazon-S3-Buckets, der angezeigt wird. Sie benötigen diese Informationen später wieder.
   b. Wählen Sie S3-Bucket erstellen, damit Amazon Rekognition Custom Labels in Ihrem Namen einen Amazon-S3-Bucket (Konsolen-Bucket) erstellen kann.
5. Schließen Sie das Browserfenster.

Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS

Sie können Amazon Rekognition Custom Labels mit den SDKs AWS Command Line Interface (AWS CLI) und AWS den SDKs verwenden. Wenn Sie Amazon Rekognition Custom Labels-Operationen vom Terminal aus ausführen müssen, installieren Sie AWS CLI. Wenn Sie eine Anwendung erstellen, laden Sie das AWS SDK für die Programmiersprache herunter, die Sie verwenden.

Themen

- Installieren Sie das AWS SDKS
- Erteilen programmgesteuerten Zugriffs
- Einrichten von SDK-Berechtigungen
- Rufen Sie einen Amazon Rekognition Custom Labels-Vorgang auf

Installieren Sie das AWS SDKS

Folgen Sie den Schritten, um die AWS SDKs herunterzuladen und zu konfigurieren.

Um die AWS CLI und die SDKs AWS einzurichten

- Laden Sie die AWS CLIund die AWS SDKs herunter, die Sie verwenden möchten, und installieren Sie sie. Dieses Handbuch enthält Beispiele für Java und Python. AWS CLI Informationen zur Installation von AWS SDKs finden Sie unter Tools für Amazon Web Services.

Erteilen programmgesteuerten Zugriffs

Sie können die Codebeispiele AWS CLI und die Codebeispiele in diesem Handbuch auf Ihrem lokalen Computer oder in anderen AWS Umgebungen, z. B. einer Amazon Elastic Compute Cloud-Instance, ausführen. Um die Beispiele auszuführen, müssen Sie Zugriff auf die AWS SDK-Operationen gewähren, die in den Beispielen verwendet werden.

Themen

- Ausführen von Code auf Ihrem lokalen Computer
- Code in AWS Umgebungen ausführen

Ausführen von Code auf Ihrem lokalen Computer

Um Code auf einem lokalen Computer auszuführen, empfehlen wir, dass Sie kurzfristige Anmeldeinformationen verwenden, um einem Benutzer Zugriff auf AWS SDK-Operationen zu gewähren. Spezifische Informationen zum Ausführen von AWS CLI und zu Codebeispielen auf einem lokalen Computer finden Sie unterVerwenden eines Profils auf Ihrem lokalen Computer.

Benutzer benötigen programmgesteuerten Zugriff, wenn sie mit AWS außerhalb des AWS Management Console interagieren möchten. Die Art und Weise, wie programmatischer Zugriff gewährt wird, hängt vom Benutzertyp ab, der zugreift. AWS

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Befolgen Sie die Anweisung en für die Schnittstelle, die Sie verwenden möchten.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		• Informationen zu den AWS CLI finden Sie unter Konfiguration der AWS CLI zu AWS IAM Identity Center verwendenden im AWS Command Line Interface Benutzerhandbuch. • Informationen zu AWS SDKs, Tools und AWS APIs finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch für AWS SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Folgen Sie den Anweisungen unter Verwenden temporäre r Anmeldeinformationen mit AWS Ressourcen im IAM-Benutzerhandbuch.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmatische Anfragen an die AWS CLI, AWS SDKs oder APIs zu signieren. AWS	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen dazu finden Sie unter Authentifizierung mithilfe von IAM-Benutzeranmeldedaten im Benutzerhandbuch. AWS CLIAWS Command Line Interface • Informationen zu AWS SDKs und Tools finden Sie unter Authentifizieren mit langfristigen Anmeldeinformationen im Referenzhandbuch für AWS SDKs und Tools. • Informationen zu AWS APIs finden Sie unter Verwaltung von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

Verwenden eines Profils auf Ihrem lokalen Computer

Sie können die Codebeispiele AWS CLI und die Codebeispiele in diesem Handbuch mit den kurzfristigen Anmeldeinformationen ausführen, in denen Sie sie erstellen. Ausführen von Code auf Ihrem lokalen Computer Um die Anmeldeinformationen und andere Einstellungsinformationen abzurufen, verwenden die Beispiele ein Profil mit dem Namen `custom-labels-access`. Zum Beispiel:

```
session = boto3.Session(profile_name='custom-labels-access')
```

```
rekognition_client = session.client("rekognition")
```

Der Benutzer, den das Profil repräsentiert, muss berechtigt sein, die Amazon Rekognition Custom Labels SDK-Operationen und andere AWS SDK-Operationen, die in den Beispielen benötigt werden, aufzurufen. Weitere Informationen finden Sie unter Einrichten von SDK-Berechtigungen. Informationen zum Zuweisen von Berechtigungen finden Sie unter Einrichten von SDK-Berechtigungen.

Um ein Profil zu erstellen, das mit den Codebeispielen AWS CLI und -Codebeispielen funktioniert, wählen Sie eine der folgenden Optionen. Stellen Sie sicher, dass der Name des von Ihnen erstellten Profils `custom-labels-access` lautet.

- Von IAM verwaltete Benutzer — Folgen Sie den Anweisungen unter Zu einer IAM-Rolle wechseln (AWS CLI).
- Personalidentität (Benutzer verwaltet von AWS IAM Identity Center) — Folgen Sie den Anweisungen unter Konfiguration der zu verwendenden AWS-CLI AWS IAM Identity Center. Für die Codebeispiele empfehlen wir die Verwendung einer integrierten Entwicklungsumgebung (IDE), die das AWS-Toolkit unterstützt und die Authentifizierung über das IAM Identity Center ermöglicht. Die Java-Beispiele finden Sie unter Mit Java entwickeln. Die Python-Beispiele finden Sie unter Mit Python entwickeln. Weitere Informationen finden Sie unter Anmeldeinformation für das IAM Identity Center.

> **Note**
>
> Sie können Code verwenden, um kurzfristige Anmeldeinformationen zu erhalten. Weitere Informationen finden Sie unter Wechseln zu einer IAM-Rolle (AWS API). Rufen Sie für IAM Identity Center die kurzfristigen Anmeldeinformationen für eine Rolle ab, indem Sie den Anweisungen unter Abrufen von IAM-Rollenanmeldeinformationen für den CLI-Zugriff folgen.

Code in AWS Umgebungen ausführen

Sie sollten keine Benutzeranmeldedaten verwenden, um AWS SDK-Aufrufe in AWS Umgebungen zu signieren, wie z. B. Produktionscode, der in einer AWS Lambda Funktion ausgeführt wird. Stattdessen konfigurieren Sie eine Rolle, die die Berechtigungen definiert, die Ihr Code benötigt. Anschließend weisen Sie die Rolle der Umgebung zu, in der Ihr Code ausgeführt wird. Wie Sie die

Rolle zuordnen und temporäre Anmeldeinformationen verfügbar machen, hängt von der Umgebung ab, in der Ihr Code ausgeführt wird:

- AWS Lambda function — Verwenden Sie die temporären Anmeldeinformationen, die Lambda Ihrer Funktion automatisch zur Verfügung stellt, wenn sie die Ausführungsrolle der Lambda-Funktion übernimmt. Die Anmeldeinformationen sind in den Lambda-Umgebungsvariablen verfügbar. Sie müssen kein Profil angeben. Weitere Informationen finden Sie unter Lambda-Ausführungsrolle.
- Amazon EC2 — Verwenden Sie den Anbieter der Anmeldeinformation für den Amazon EC2 Instance-Metadatenendpunkt Der Anbieter generiert und aktualisiert automatisch Anmeldeinformationen für Sie mithilfe des Amazon EC2 Instance-Profils, das Sie der Amazon EC2-Instance anhängen. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon-EC2-Instances ausgeführt werden
- Amazon Elastic Container Service — Verwenden Sie den Anbieter für Container-Anmeldeinformationen. Amazon ECS sendet und aktualisiert Anmeldeinformationen an einen Metadaten-Endpunkt. Eine von Ihnen angegebene Aufgaben-IAM-Rolle bietet eine Strategie für die Verwaltung der Anmeldeinformationen, die Ihre Anwendung verwendet. Weitere Informationen finden Sie unter Interagieren mit AWS-Services.

Weitere Informationen zu Anbietern von Anmeldeinformationen finden Sie unter Standardisierte Anmeldeinformationsanbieter.

Einrichten von SDK-Berechtigungen

Um Amazon Rekognition Custom Labels SDK-Operationen verwenden zu können, benötigen Sie Zugriffsberechtigungen für die Amazon Rekognition Custom Labels API und den Amazon-S3-Bucket, der für das Modelltraining verwendet wird.

Themen

- Erteilen von SDK-Betriebsberechtigungen
- Aktualisierungen der Richtlinien für die Verwendung des AWS SDK
- Zuweisen von Berechtigungen

Erteilen von SDK-Betriebsberechtigungen

Es wird empfohlen, nur die Berechtigungen zu erteilen, die zum Ausführen einer Aufgabe erforderlich sind (geringste Berechtigungen). Um beispielsweise aufzurufen DetectCustomLabels, benötigen Sie

eine Genehmigung zur Ausführung. `rekognition:DetectCustomLabels` Die Berechtigungen für einen Vorgang finden Sie in der API-Referenz.

Wenn Sie gerade erst mit einer Anwendung beginnen, wissen Sie möglicherweise nicht, welche spezifischen Berechtigungen Sie benötigen, sodass Sie mit umfassenderen Berechtigungen beginnen können. AWS verwaltete Richtlinien stellen Berechtigungen bereit, die Ihnen bei den ersten Schritten helfen. Sie können die `AmazonRekognitionCustomLabelsFullAccess` AWS verwaltete Richtlinie verwenden, um vollständigen Zugriff auf die Amazon Rekognition Custom Labels API zu erhalten. Weitere Informationen finden Sie unter AWS-verwaltete Richtlinie: AmazonRekognitionCustomLabelsFullAccess. Wenn Sie wissen, welche Berechtigungen Ihre Anwendung benötigt, können Sie die Berechtigungen weiter reduzieren, indem Sie vom Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen zu den vom Kunden verwalteten Richtlinien finden Sie unter Vom Kunden verwaltete Richtlinien.

Informationen zum Zuweisen von Berechtigungen finden Sie unter Zuweisen von Berechtigungen.

Aktualisierungen der Richtlinien für die Verwendung des AWS SDK

Um das AWS SDK mit der neuesten Version von Amazon Rekognition Custom Labels zu verwenden, müssen Sie Amazon Rekognition Custom Labels keine Berechtigungen mehr erteilen, um auf den Amazon S3 S3-Bucket zuzugreifen, der Ihre Trainings- und Testbilder enthält. Wenn Sie zuvor Berechtigungen hinzugefügt haben, müssen Sie diese nicht entfernen. Wenn Sie möchten, entfernen Sie alle Richtlinien aus dem Bucket, in dem der Dienst für den Prinzipal `rekognition.amazonaws.com` ist. Beispielsweise:

```
"Principal": {
    "Service": "rekognition.amazonaws.com"
}
```

Weitere Informationen finden Sie unter Verwenden von Bucket-Richtlinien.

Zuweisen von Berechtigungen

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in: AWS IAM Identity Center

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center -Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.
- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

Rufen Sie einen Amazon Rekognition Custom Labels-Vorgang auf

Führen Sie den folgenden Code aus, um zu bestätigen, dass Sie Aufrufe an die Amazon Rekognition Custom Labels API tätigen können. Der Code listet die Projekte in Ihrem AWS Konto in der aktuellen AWS Region auf. Wenn Sie noch kein Projekt erstellt haben, ist die Antwort leer, bestätigt aber, dass Sie den `DescribeProjects`-Vorgang aufrufen können.

Im Allgemeinen erfordert das Aufrufen einer Beispielfunktion einen AWS SDK Rekognition-Client und alle anderen erforderlichen Parameter. Der AWS SDK-Client ist in der Hauptfunktion ausgewiesen.

Wenn der Code fehlschlägt, überprüfen Sie, ob der von Ihnen verwendete Benutzer über die richtigen Berechtigungen verfügt. Überprüfen Sie auch, welche AWS Region Sie verwenden, da Amazon Rekognition Custom Labels nicht in allen AWS Regionen verfügbar ist.

So rufen Sie einen Amazon Rekognition Custom Labels-Vorgang auf

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Beispielcode, um Ihre Projekte anzusehen.

   CLI

   Verwenden Sie den `describe-projects`-Befehl, um die Projekte in Ihrem Konto aufzulisten.

   ```
   aws rekognition describe-projects \
   ```

```
--profile custom-labels-access
```

Python

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
This example shows how to describe your Amazon Rekognition Custom Labels
 projects.
If you haven't previously created a project in the current AWS Region,
the response is an empty list, but does confirm that you can call an
Amazon Rekognition Custom Labels operation.
"""
from botocore.exceptions import ClientError
import boto3

def describe_projects(rekognition_client):
    """
    Lists information about the projects that are in in your AWS account
    and in the current AWS Region.

    : param rekognition_client: A Boto3 Rekognition client.
    """
    try:
        response = rekognition_client.describe_projects()
        for project in response["ProjectDescriptions"]:
            print("Status: " + project["Status"])
            print("ARN: " + project["ProjectArn"])
            print()
        print("Done!")
    except ClientError as err:
        print(f"Couldn't describe projects. \n{err}")
        raise


def main():
    """
    Entrypoint for script.
    """
```

```
    session = boto3.Session(profile_name='custom-labels-access')
    rekognition_client = session.client("rekognition")

    describe_projects(rekognition_client)


if __name__ == "__main__":
    main()
```

Java V2

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetMetadata;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class Hello {

    public static final Logger logger = Logger.getLogger(Hello.class.getName());

    public static void describeMyProjects(RekognitionClient rekClient) {

        DescribeProjectsRequest descProjects = null;

        // If a single project name is supplied, build projectNames argument
```

```
        List<String> projectNames = new ArrayList<String>();


        descProjects = DescribeProjectsRequest.builder().build();

        // Display useful information for each project.

        DescribeProjectsResponse resp =
 rekClient.describeProjects(descProjects);

        for (ProjectDescription projectDescription : resp.projectDescriptions())
 {

            System.out.println("ARN: " + projectDescription.projectArn());
            System.out.println("Status: " +
 projectDescription.statusAsString());
            if (projectDescription.hasDatasets()) {
                for (DatasetMetadata datasetDescription :
 projectDescription.datasets()) {
                    System.out.println("\tdataset Type: " +
 datasetDescription.datasetTypeAsString());
                    System.out.println("\tdataset ARN: " +
 datasetDescription.datasetArn());
                    System.out.println("\tdataset Status: " +
 datasetDescription.statusAsString());
                }
            }
            System.out.println();
        }

    }

    public static void main(String[] args) {

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();
```

```
            // Describe projects

            describeMyProjects(rekClient);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Schritt 5: (Optional) Verschlüsseln von Trainingsdateien

Sie können eine der folgenden Optionen wählen, um die Amazon Rekognition Custom Labels-Manifestdateien und Bilddateien zu verschlüsseln, die sich in einem Konsolen-Bucket oder einem externen Amazon-S3-Bucket befinden.

- Verwenden Sie einen Amazon S3-Schlüssel (SSE-S3)
- Benutze deine AWS KMS key.

> **Note**
>
> Der aufrufende IAM-Prinzipal benötigt Berechtigungen, um die Dateien zu entschlüsseln. Weitere Informationen finden Sie unter Entschlüsseln von Dateien verschlüsselt mit AWS Key Management Service.

Weitere Informationen zur Amazon S3 Bucket-Verschlüsselung finden Sie unter Amazon S3-SSE-Standardverschlüsselung für Amazon-S3-Buckets einrichten.

Entschlüsseln von Dateien verschlüsselt mit AWS Key Management Service

Wenn Sie AWS Key Management Service (KMS) verwenden, um Ihre Amazon Rekognition Custom Labels-Manifestdateien und Bilddateien zu verschlüsseln, fügen Sie den IAM-Prinzipal,

der Amazon Rekognition Custom Labels aufruft, zur Schlüsselrichtlinie des KMS-Schlüssels hinzu. Auf diese Weise kann Amazon Rekognition Custom Labels Ihre Manifest- und Bilddateien vor dem Training entschlüsseln. Weitere Informationen finden Sie unter Mein Amazon-S3-Bucket hat eine Standardverschlüsselung über einen benutzerdefinierten AWS-KMS-Schlüssel. Wie kann ich Benutzern erlauben, Daten aus dem Bucket herunterzuladen und in ihn hochzuladen?

Der IAM-Prinzipal benötigt die folgenden Berechtigungen für den KMS-Schlüssel.

- km: GenerateDataKey
- kms:Decrypt

Weitere Informationen finden Sie unter Schutz von Daten mit serverseitiger Verschlüsselung mit in AWS Key Management Service (SSE-KMS) gespeicherten KMS-Schlüsseln.

Verschlüsseln kopierter Trainings- und Testbilder

Um Ihr Modell zu trainieren, erstellt Amazon Rekognition Custom Labels eine Kopie Ihrer ursprünglichen Trainings- und Testbilder. Standardmäßig werden die kopierten Bilder im Ruhezustand mit einem Schlüssel verschlüsselt, der AWS gehört und von AWS verwaltet wird. Sie haben auch die Möglichkeit, Ihren eigenen AWS KMS key zu verwenden. Wenn Sie Ihren eigenen KMS-Schlüssel verwenden, benötigen Sie die folgenden Berechtigungen für den KMS-Schlüssel.

- km: CreateGrant
- km: DescribeKey

Sie können den KMS-Schlüssel optional angeben, wenn Sie das Modell mit der Konsole trainieren oder wenn Sie den `CreateProjectVersion` Vorgang aufrufen. Der KMS-Schlüssel, den Sie verwenden, muss nicht derselbe KMS-Schlüssel sein, den Sie zum Verschlüsseln von Manifest- und Bilddateien in Ihrem Amazon-S3-Bucket verwenden. Weitere Informationen finden Sie unter Schritt 5: (Optional) Verschlüsseln von Trainingsdateien.

Weitere Informationen finden Sie unter AWS Key Management Service-Konzepte. Ihre Quellbilder sind davon nicht betroffen.

Für weitere Informationen zum Schulen eines Modells siehe Schulung eines Amazon-Rekognition-Custom-Labels-Modells.

Schritt 6: (Optional) Zuordnen früherer Datensätze zu mit neuen Projekten

Amazon Rekognition Custom Labels verwaltet jetzt Datensätze mit Projekten. Frühere Datensätze, die Sie erstellt haben, sind schreibgeschützt und müssen einem Projekt zugeordnet werden, bevor Sie sie verwenden können. Wenn Sie die Detailseite für ein Projekt mit der Konsole öffnen, ordnen wir automatisch die Datensätze, mit denen die neueste Version des Projektmodells trainiert wurde, dem Projekt zu. Die automatische Zuordnung eines Datensatzes zu einem Projekt erfolgt nicht, wenn Sie das AWS SDK verwenden.

Nicht zugeordnete frühere Datensätze wurden nie zum Trainieren eines Modells oder zum Trainieren einer früheren Version eines Modells verwendet. Auf der Seite „Frühere Datensätze“ werden all Ihre zugeordneten und nicht zugeordneten Datensätze angezeigt.

Um einen früheren Datensatz zu verwenden, der nicht zugeordnet ist, erstellen Sie auf der Seite Frühere Datensätze ein neues Projekt. Der Datensatz wird zum Trainingsdatensatz für das neue Projekt. Sie können auch ein Projekt für einen bereits zugeordneten Datensatz erstellen, da frühere Datensätze mehrere Zuordnungen haben können.

Um einen früheren Datensatz einem neuen Projekt zuzuordnen

1. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie im linken Bereich Benutzerdefinierte Labels verwenden aus. Die Landingpage von Amazon Rekognition Custom Labels wird angezeigt.
3. Wählen Sie im linken Navigationsbereich Frühere Datensätze aus.
4. Wählen Sie in der Datensatzansicht den vorherigen Datensatz aus, den Sie einem Projekt zuordnen möchten.
5. Wählen Sie Projekt mit Datensatz erstellen aus.
6. Geben Sie auf der Seite Projekt erstellen unter Projektname einen Namen für Ihr neues Projekt ein.
7. Wählen Sie Projekt erstellen, um das Projekt zu erstellen. Die Erstellung des Projekts kann eine Weile dauern.
8. Verwenden Sie das Projekt. Weitere Informationen finden Sie unter Grundlegendes zu Amazon Rekognition Custom Labels.

Verwenden eines früheren Datensatzes als Testdatensatz

Sie können einen früheren Datensatz als Testdatensatz für ein vorhandenes Projekt verwenden, indem Sie den vorherigen Datensatz zunächst einem neuen Projekt zuordnen. Anschließend kopieren Sie den Trainingsdatensatz des neuen Projekts in den Testdatensatz des vorhandenen Projekts.

So verwenden Sie einen früheren Datensatz als Testdatensatz

1. Folgen Sie den Anweisungen unter Schritt 6: (Optional) Zuordnen früherer Datensätze zu mit neuen Projekten, um den vorherigen Datensatz einem neuen Projekt zuzuordnen.
2. Erstellen Sie den Testdatensatz im vorhandenen Projekt, indem Sie den Trainingsdatensatz aus dem neuen Projekt kopieren. Weitere Informationen finden Sie unter Bestehender Datensatz.
3. Folgen Sie den Anweisungen unter Löschen eines Amazon Rekognition Custom Labels-Projekts (Konsole), um das neue Projekt zu löschen.

Alternativ können Sie den Testdatensatz erstellen, indem Sie die Manifestdatei für den vorherigen Datensatz verwenden. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

Grundlegendes zu Amazon Rekognition Custom Labels

Dieser Abschnitt gibt Ihnen einen Überblick über den Arbeitsablauf zum Trainieren und Verwenden eines Amazon Rekognition Custom Labels-Modells mit der Konsole und dem AWS SDK.

> **Note**
>
> Amazon Rekognition Custom Labels verwaltet jetzt Datensätze innerhalb eines Projekts. Sie können Datensätze für Ihre Projekte mit der Konsole und dem SDK erstellen. AWS Wenn Sie zuvor Amazon Rekognition Custom Labels verwendet haben, müssen Ihre älteren Datensätze möglicherweise einem neuen Projekt zugeordnet werden. Weitere Informationen finden Sie unter Schritt 6: (Optional) Zuordnen früherer Datensätze zu mit neuen Projekten.

Themen

Ihren Modelltyp festlegen

Sie entscheiden zunächst, welche Art von Modell Sie trainieren möchten, was von Ihren Geschäftszielen abhängt. Sie könnten einem Modell beispielsweise beibringen, Ihr Logo in Social-Media-Posts zu finden, Ihre Produkte in den Verkaufsregalen zu identifizieren oder Maschinenteile auf einem Fließband zu klassifizieren.

Amazon Rekognition Custom Labels kann die folgenden Modelltypen trainieren:

- Objekte, Szenen und Konzepte finden
- Nach Objektpositionen suchen
- Position von Marken finden

Um Ihnen bei der Entscheidung zu helfen, welche Art von Modell Sie trainieren möchten, bietet Amazon Rekognition Custom Labels Beispielprojekte, die Sie verwenden können. Weitere Informationen finden Sie unter Erste Schritte mit Amazon Rekognition Custom Labels.

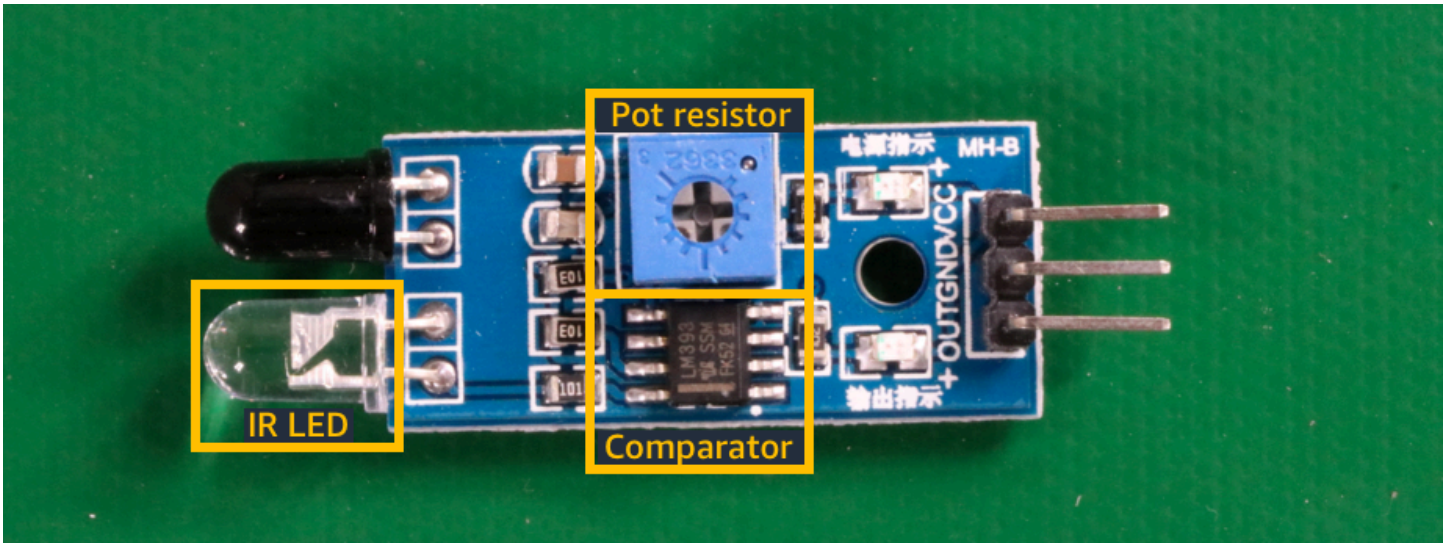
Objekte, Szenen und Konzepte finden

Das Modell sagt Klassifizierungen für die Objekte, Szenen und Konzepte voraus, die einem gesamten Bild zugeordnet sind. Sie können beispielsweise ein Modell trainieren, das bestimmt, ob ein Bild eine Touristenattraktion enthält oder nicht. Ein Beispielobjekt finden Sie unter Bildklassifizierung. Das folgende Bild eines Sees ist ein Beispiel für die Art von Bild, in dem Sie Objekte, Szenen und Konzepte erkennen können.

Alternativ können Sie ein Modell trainieren, das Bilder in mehrere Kategorien einteilt. Das vorherige Bild könnte beispielsweise Kategorien wie Himmelsfarbe, Spiegelung oder See enthalten. Ein Beispielobjekt finden Sie unter Bildklassifizierung (mehrere Label).

Nach Objektpositionen suchen

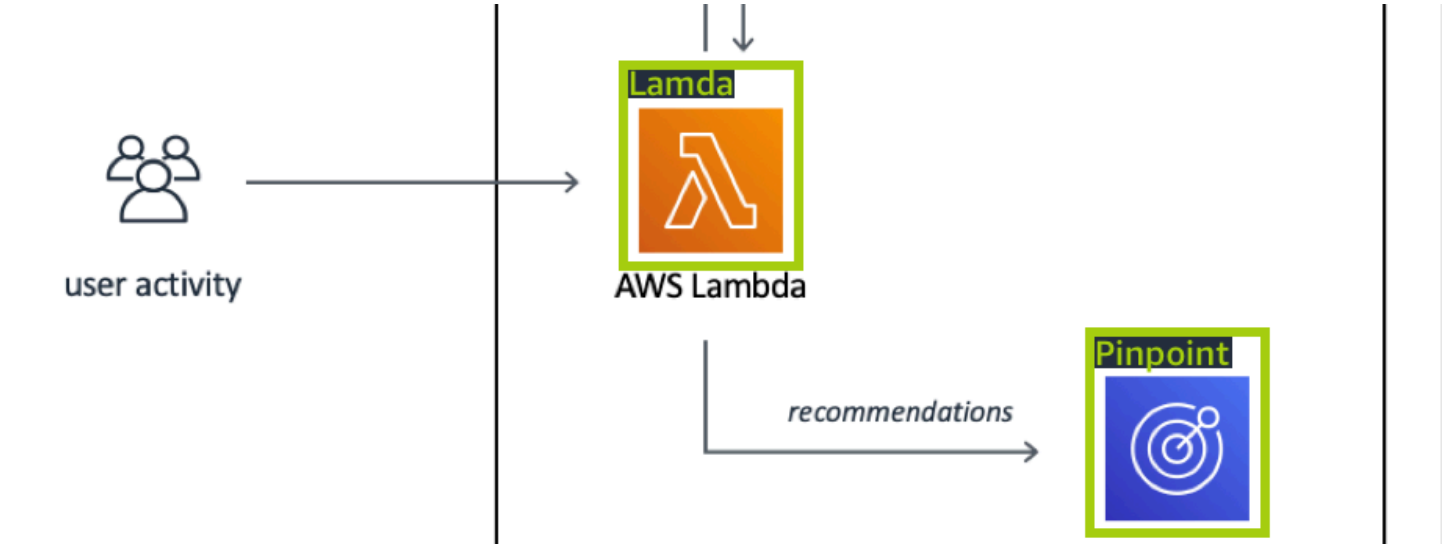
Das Modell sagt die Position eines Objekts auf einem Bild voraus. Die Vorhersage umfasst Begrenzungsrahmen-Informationen für die Objektposition und ein Label, das das Objekt innerhalb des Begrenzungsrahmens identifiziert. Die folgende Abbildung zeigt beispielsweise Begrenzungsrahmen, die verschiedene Teile einer Leiterplatte umgeben, z. B. einen Komparator oder einen Pot-Widerstand.



Das Lokalisierung von Objekten-Beispielprojekt zeigt, wie Amazon Rekognition Custom Labels mit Labeln versehene Begrenzungsrahmen verwendet, um ein Modell zu trainieren, das Objektpositionen findet.

Position von Marken finden

Mit Amazon Rekognition Custom Labels kann ein Modell trainiert werden, das die Position von Marken, z. B. Logos, auf einem Bild ermittelt. Die Vorhersage umfasst Informationen zum Begrenzungsrahmen für die Position der Marke und ein Label, das das Objekt innerhalb des Begrenzungsrahmens identifiziert. Ein Beispielobjekt finden Sie unter Erkennung von Marken. Die folgende Abbildung zeigt ein Beispiel für einige Marken, die das Modell erkennen kann.



Erstellen eines Modells

Die Schritte zum Erstellen eines Modells umfassen das Erstellen eines Projekts, das Erstellen von Trainings- und Testdatensätzen und das Trainieren des Modells.

Erstellen eines Projekts

Ein Amazon Rekognition Custom Labels-Projekt ist eine Gruppe von Ressourcen, die zum Erstellen und Verwalten eines Modells benötigt werden. Ein Projekt verwaltet Folgendes:

- Datensätze – Die Bilder und Bildlabels, die zum Trainieren eines Modells verwendet werden. Ein Projekt hat einen Trainingsdatensatz und einen Testdatensatz.
- Modelle – Die Software, die Sie trainieren, um Konzepte, Szenen und Objekte zu finden, die für Ihr Unternehmen einzigartig sind. Sie können mehrere Versionen eines Modells in einem Projekt haben.

Es wird empfohlen, ein Projekt für einen einzigen Anwendungsfall zu verwenden, z. B. für die Suche nach Leiterplattenteilen auf einer Leiterplatte.

Sie können ein Projekt mit der Amazon Rekognition Custom Labels-Konsole und mit der CreateProjectAPI erstellen. Weitere Informationen finden Sie unter Erstellen eines eines Projekts.

Erstellen von Trainings- und Testdatensätzen

Ein Datensatz besteht aus einer Reihe von Bildern und Labels, die diese Bilder beschreiben. In Ihrem Projekt erstellen Sie einen Trainingsdatensatz und einen Testdatensatz, den Amazon Rekognition Custom Labels zum Trainieren und Testen Ihres Modells verwendet.

Ein Label identifiziert ein Objekt, eine Szene, ein Konzept oder einen Begrenzungsrahmen, der ein Objekt in einem Bild umgibt. Labels werden entweder einem ganzen Bild (Bildebene) oder einem Begrenzungsrahmen zugewiesen, der ein Objekt auf einem Bild umgibt.

> **Important**
>
> Wie Sie die Bilder in Ihren Datensätzen beschriften, bestimmt den Modelltyp, den Amazon Rekognition Custom Labels erstellt. Um beispielsweise ein Modell zu trainieren, das Objekte, Szenen und Konzepte findet, weisen Sie den Bildern in Ihren Trainings- und Testdatensätzen Labels auf Bildebene zu. Weitere Informationen finden Sie unter Datensätzen einen Zweck geben.

Bilder müssen im PNG- und JPEG-Format vorliegen, und Sie sollten die Empfehlungen für Eingabebilder befolgen. Weitere Informationen finden Sie unter Vorbereiten der Bilder.

Erstellen von Trainings- und Testdatensätzen (Konsole)

Sie können ein Projekt mit einem einzelnen Datensatz oder mit separaten Trainings- und Testdatensätzen beginnen. Wenn Sie mit einem einzelnen Datensatz beginnen, teilt Amazon Rekognition Custom Labels Ihren Datensatz während des Trainings auf, um einen Trainingsdatensatz (80 %) und einen Testdatensatz (20 %) für Ihr Projekt zu erstellen. Beginnen Sie mit einem einzigen Datensatz, wenn Amazon Rekognition Custom Labels entscheiden soll, welche Bilder zum Trainieren und Testen verwendet werden. Um die vollständige Kontrolle über Trainings, Tests und Leistungsoptimierungen zu haben, empfehlen wir, dass Sie Ihr Projekt mit separaten Trainings- und Testdatensätzen beginnen.

Um die Datensätze für ein Projekt zu erstellen, importieren Sie die Bilder auf eine der folgenden Arten:

- Importieren Sie Bilder von Ihrem lokalen Computer.
- Importieren Sie Bilder aus einem S3-Bucket. Amazon Rekognition Custom Labels kann die Bilder anhand der Ordnernamen beschriften, die die Bilder enthalten.

- Importieren Sie eine Amazon SageMaker Ground Truth Manifestdatei.
- Kopieren Sie einen bestehenden Amazon Rekognition Custom Labels-Datensatz.

Weitere Informationen finden Sie unter Erstellen von Trainings- und Testdatensätzen mit Bildern.

Je nachdem, von wo Sie Ihre Bilder importieren, haben Ihre Bilder möglicherweise keine Labels. Beispielsweise haben Bilder, die von einem lokalen Computer importiert wurden, keine Label. Bilder, die aus einer Amazon SageMaker Ground Truth Manifest-Datei importiert wurden, sind beschriftet. Sie können die Amazon Rekognition Custom Labels-Konsole verwenden, um Labels hinzuzufügen, zu ändern und zuzuweisen. Weitere Informationen finden Sie unter Labeling von Bildern.

Informationen zum Erstellen Ihrer Trainings- und Testdatensätze mit der Konsole finden Sie unter Erstellen von Trainings- und Testdatensätzen mit Bildern. Ein Tutorial, das das Erstellen von Trainings- und Testdatensätzen beinhaltet, finden Sie unter Tutorial: Bilder klassifizieren.

Erstellen von Trainings- und Testdatensätzen (SDK)

Um Ihre Trainings- und Testdatensätze zu erstellen, verwenden Sie die `CreateDataset`-API. Sie können einen Datensatz erstellen, indem Sie eine Manifestdatei im Amazon Sagemaker-Format verwenden oder einen vorhandenen Amazon Rekognition Custom Labels-Datensatz kopieren. Weitere Informationen finden Sie unter Erstellen von Trainings- und Testdatensätzen (SDK) . Bei Bedarf können Sie Ihre eigene Manifestdatei erstellen. Weitere Informationen finden Sie unter the section called “Erstellen einer Manifestdatei”.

Trainieren Ihres Modells

Trainieren Sie Ihr Modell mit dem Trainingsdatensatz. Bei jedem Training wird eine neue Version eines Modells erstellt. Während des Trainings testet Amazon Rekognition Custom Labels die Leistung Ihres trainierten Modells. Sie können die Ergebnisse verwenden, um Ihr Modell zu bewerten und zu verbessern. Es dauert eine Weile, bis das Training abgeschlossen ist. Ihnen wird nur ein erfolgreiches Modelltraining in Rechnung gestellt. Weitere Informationen finden Sie unter Schulung eines Amazon-Rekognition-Custom-Labels-Modells. Wenn das Modelltraining fehlschlägt, stellt Amazon Rekognition Custom Labels Debugging-Informationen bereit, die Sie verwenden können. Weitere Informationen finden Sie unter Debuggen eines fehlgeschlagenen Modelltrainings.

Ihr Modell trainieren (Konsole)

Informationen zum Trainieren Ihres Modells mit der Konsole finden Sie unter Ein Model trainieren (Konsole).

Ein Modell trainieren (SDK)

Sie trainieren ein Amazon Rekognition Custom Labels-Modell, indem Sie Version aufrufenCreateProject. Weitere Informationen finden Sie unter Ein Modell trainieren (SDK).

Verbessern Ihres Modells

Während des Tests erstellt Amazon Rekognition Custom Labels Bewertungsmetriken, mit denen Sie Ihr trainiertes Modell verbessern können.

Bewerten Ihres Modells

Bewerten Sie die Leistung Ihres Modells anhand der beim Testen erstellten Leistungsmetriken. Leistungsmetriken wie F1, Präzision und Rückruf ermöglichen es Ihnen, die Leistung Ihres trainierten Modells zu verstehen und zu entscheiden, ob Sie es in der Produktion einsetzen möchten. Weitere Informationen finden Sie unter Metriken für die Bewertung Ihres Modells.

Bewerten eines Modells (Konsole)

Die Leistungsmetriken finden Sie unter Zugreifen auf Bewertungsmetriken (Konsole).

Bewerten eines Modells (SDK)

Um Leistungskennzahlen zu erhalten, rufen Sie DescribeProjectVersions auf, um die Testergebnisse abzurufen. Weitere Informationen finden Sie unter Zugreifen auf Amazon Rekognition Custom Labels-Bewertungsmetriken (SDK). Die Testergebnisse enthalten Metriken, die in der Konsole nicht verfügbar sind, z. B. eine Konfusionsmatrix für Klassifizierungsergebnisse. Die Testergebnisse werden in den folgenden Formaten zurückgegeben:

- F1-Score – Ein einzelner Wert, der die Gesamtleistung des Modells in Bezug auf Präzision und Rückrufvermögen darstellt. Weitere Informationen finden Sie unter F1.
- Speicherort der Datei mit der Zusammenfassung – Die Testzusammenfassung umfasst aggregierte Bewertungsmetriken für den gesamten Testdatensatz und Messwerte für jedes einzelne Label. `DescribeProjectVersions` gibt den S3-Bucket und den Speicherort des Ordners der Datei mit der Zusammenfassung zurück. Weitere Informationen finden Sie unter Übersichtsdatei.
- Speicherort des Snapshots des Bewertungsmanifests – Der Snapshot enthält Details zu den Testergebnissen, einschließlich der Konfidenzwerte und der Ergebnisse binärer

Klassifizierungstests, z. B. falsch positiver Ergebnisse. `DescribeProjectVersions` gibt den S3-Bucket und den Speicherort der Snapshot-Dateien zurück. Weitere Informationen finden Sie unter Bewertungsmanifest-Snapshot.

Verbessern Ihres Modells

Wenn Verbesserungen erforderlich sind, können Sie weitere Trainingsbilder hinzufügen oder die Datensatz-Labels verbessern. Weitere Informationen finden Sie unter Verbessern eines Amazon Rekognition Custom Labels-Modells. Sie können auch Feedback zu den Vorhersagen geben, die Ihr Modell macht, und es verwenden, um Ihr Modell zu verbessern. Weitere Informationen finden Sie unter Feedback-Lösung modellieren.

Verbessern Sie Ihr Modell (Konsole)

Informationen zum Hinzufügen von Bildern zu einem Datensatz finden Sie unter Hinzufügen weiterer Bilder zu einem Datensatz. Informationen zum Hinzufügen oder Ändern von Labels finden Sie unter the section called “Labeling von Bildern”.

Informationen zum erneuten Trainieren Ihres Modells finden Sie unter Ein Model trainieren (Konsole).

Verbessern Sie Ihr Modell (SDK)

Verwenden Sie die `UpdateDatasetEntries`-API, um Bilder zu einem Datensatz hinzuzufügen oder die Labels für ein Bild zu ändern. `UpdateDatasetEntries` aktualisiert oder fügt JSON-Zeilen zu einer Manifestdatei hinzu. Jede JSON-Zeile enthält Informationen für ein einzelnes Bild, z. B. zugewiesene Labels oder Begrenzungsrahmen-Informationen. Weitere Informationen finden Sie unter Weitere Bilder hinzufügen (SDK). Verwenden Sie die `ListDatasetEntries`-API, um die Einträge in einem Datensatz anzuzeigen.

Informationen zum erneuten Trainieren Ihres Modells finden Sie unter Ein Modell trainieren (SDK).

Starten Ihres Modells

Bevor Sie Ihr Modell verwenden können, starten Sie das Modell mithilfe der Amazon Rekognition Custom Labels-Konsole oder der `StartProjectVersion`-API. Ihnen wird die Zeit in Rechnung gestellt, während der Ihr Modell ausgeführt wird. Weitere Informationen finden Sie unter Ein trainiertes Modell ausführen.

Starten Ihres Modells (Konsole)

Informationen zum Starten Ihres Modells mit der Konsole finden Sie unter Starten eines Amazon Rekognition Custom Labels-Modells (Konsole).

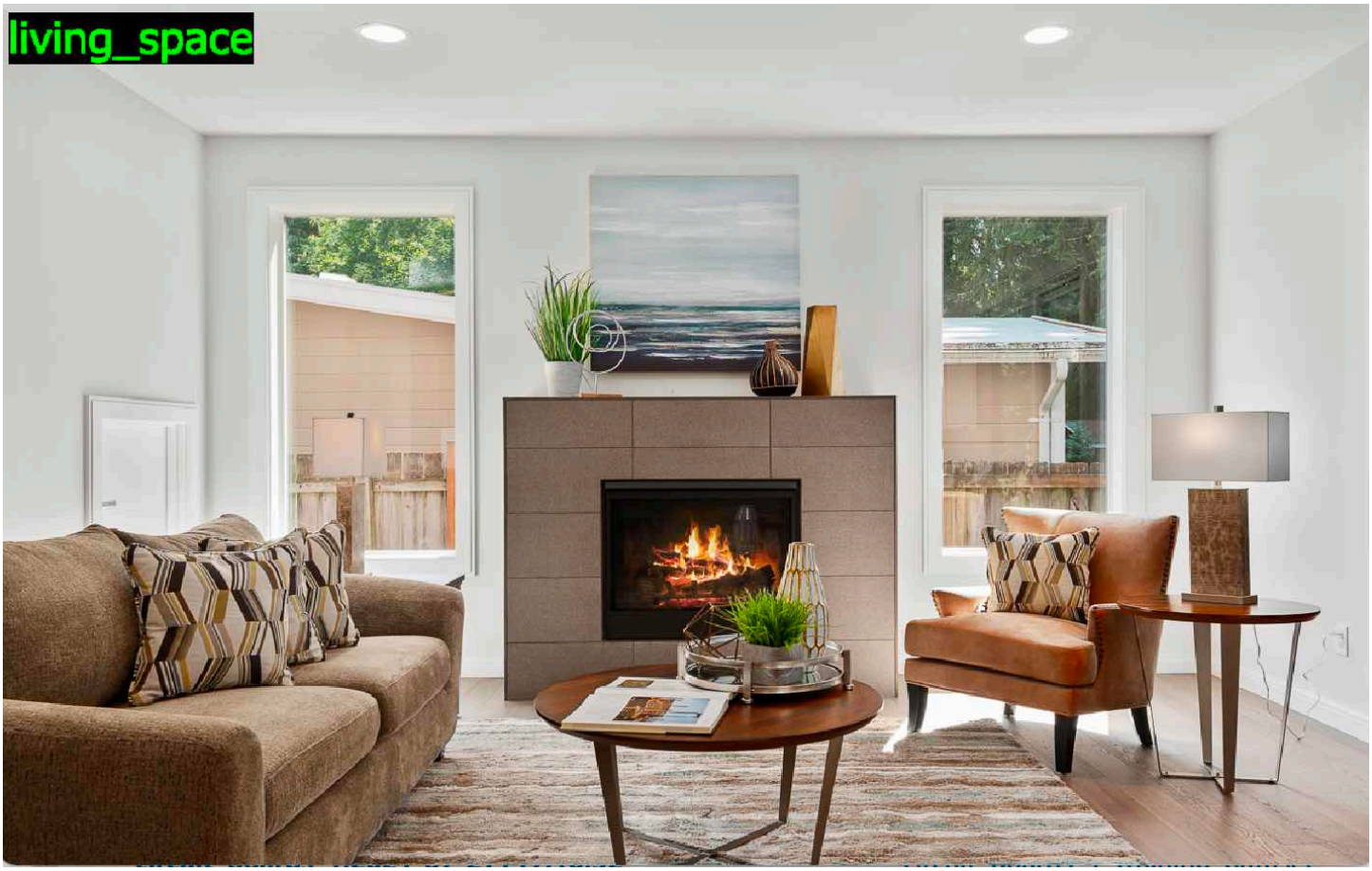
Starten Ihres Modells

Sie starten Ihr Modell mit dem Aufruf StartProjectvon Version. Weitere Informationen finden Sie unter Starten eines Amazon Rekognition Custom Labels-Modells (SDK).

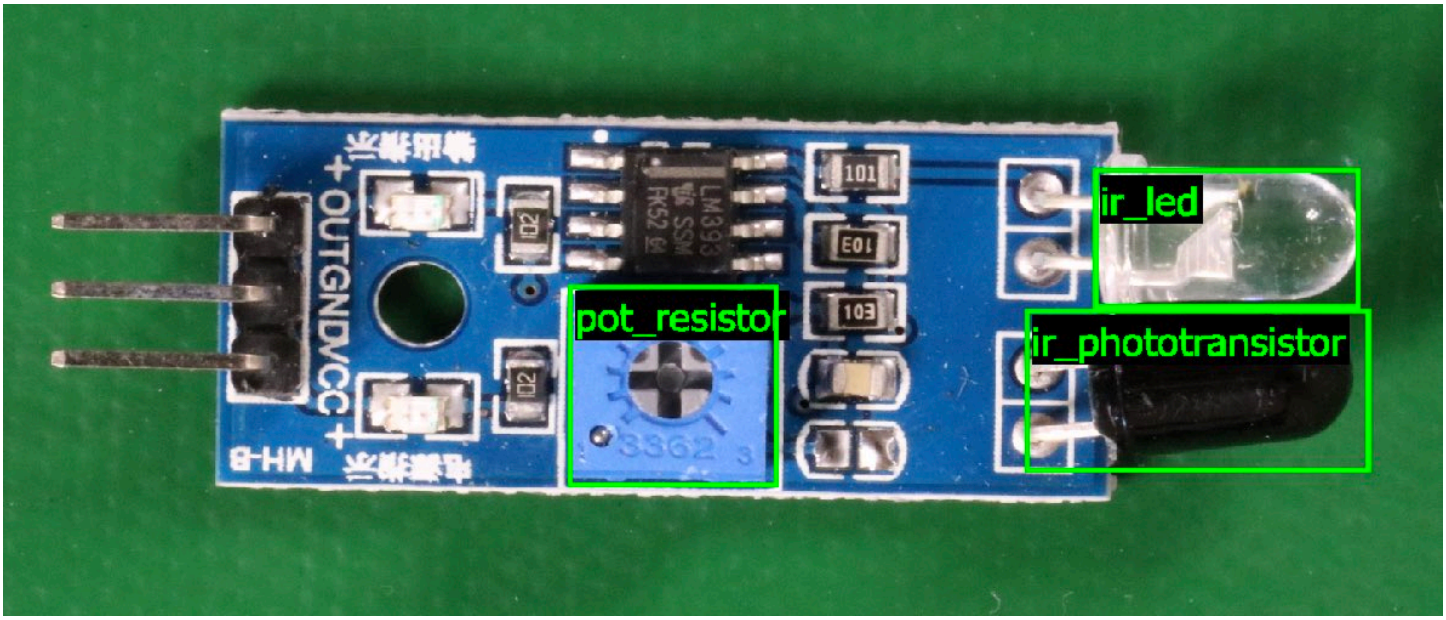
Analysieren eines Bilds

Um ein Bild mit Ihrem Modell zu analysieren, verwenden Sie die `DetectCustomLabels`-API. Sie können ein lokales Bild oder ein in einem S3-Bucket gespeichertes Bild angeben. Für den Vorgang ist auch der Amazon-Ressourcenname (ARN) des Modells, das Sie verwenden möchten, erforderlich.

Wenn Ihr Modell Objekte, Szenen und Konzepte findet, enthält die Antwort eine Liste der Labels auf Bildebene, die im Bild gefunden wurden. In der folgenden Abbildung werden beispielsweise die Labels auf Bildebene angezeigt, die mit dem Beispielprojekt Zimmer gefunden wurden.



Wenn das Modell Objektpositionen findet, enthält die Antwort eine Liste der mit Labels versehenen Begrenzungsrahmen, die im Bild zu finden sind. Ein Begrenzungsrahmen stellt die Position eines Objekts auf einem Bild dar. Sie können die Informationen zum Begrenzungsrahmen verwenden, um einen Begrenzungsrahmen um ein Objekt zu zeichnen. Die folgende Abbildung zeigt beispielsweise Begrenzungsrahmen rund um Leiterplattenteile, die im Beispielprojekt Leiterplatten gefunden wurden.



Weitere Informationen finden Sie unter Analysieren eines Bildes mit einem trainierten Modell.

Stoppen Ihres Modells

Ihnen wird die Zeit in Rechnung gestellt, während der Ihr Modell ausgeführt wird. Wenn Sie Ihr Modell nicht mehr verwenden, stoppen Sie es mithilfe der Amazon Rekognition Custom Labels-Konsole oder mithilfe der `StopProjectVersion`-API. Weitere Informationen finden Sie unter Stoppen eines Amazon Rekognition Custom Labels-Modells.

Stoppen Ihres Modells (Konsole)

Informationen zum Stoppen eines Modells, das mit der Konsole ausgeführt wird, finden Sie unter Stoppen eines Amazon Rekognition Custom Labels-Modells (Konsole).

Stoppen Ihres Modells (SDK)

Um ein laufendes Modell zu beenden, rufen Sie StopProject Version auf. Weitere Informationen finden Sie unter Stoppen eines Amazon Rekognition Custom Labels-Modells (SDK).

Erste Schritte mit Amazon Rekognition Custom Labels

Bevor Sie mit diesen Anleitungen für die Ersten Schritte beginnen, empfehlen wir Ihnen, Grundlegendes zu Amazon Rekognition Custom Labels zu lesen.

Sie verwenden Amazon Rekognition Custom Labels, um ein Machine Learning-Modell zu trainieren. Das trainierte Modell analysiert Bilder, um die Objekte, Szenen und Konzepte zu finden, die für Ihre Geschäftsanforderungen einzigartig sind. Sie können einem Modell beispielsweise beibringen, Bilder von Häusern zu klassifizieren oder die Position von elektronischen Bauteilen auf einer Leiterplatte zu ermitteln.

Um Ihnen den Einstieg zu erleichtern, enthält Amazon Rekognition Custom Labels Videotutorials und Beispielprojekte.

> **Note**
>
> Informationen zu den AWS Regionen und Endpunkten, die Amazon Rekognition Custom Labels unterstützt, finden Sie unter Rekognition Endpoints and quota.

Videotutorials

Die Videos zeigen Ihnen, wie Sie Amazon Rekognition Custom Labels verwenden, um ein Modell zu trainieren und zu verwenden.

So sehen Sie sich die Videotutorials an

1. Melden Sie sich bei der Amazon Rekognition Rekognition-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie im linken Navigationsbereich Custom Labels verwenden aus. Die Landingpage von Amazon Rekognition Custom Labels wird angezeigt. Wenn die Option Benutzerdefinierte Labels verwenden nicht angezeigt wird, überprüfen Sie, ob die AWS Region, die Sie verwenden, Amazon Rekognition Custom Labels unterstützt.
3. Wählen Sie im Navigationsbereich Erste Schritte aus.
4. In Was ist Amazon Rekognition Custom Labels?, wählen Sie das Video aus, um das Übersichtsvideo anzusehen.
5. Wählen Sie im Navigationsbereich Tutorials aus.

6. Wählen Sie auf der Seite Tutorials die Videotutorials aus, die Sie sich ansehen möchten.

Beispielprojekte

Amazon Rekognition Custom Labels bietet die folgenden Beispielprojekte.

Bildklassifizierung

Das Projekt zur Bildklassifizierung (Zimmer) trainiert ein Modell, das einen oder mehrere Standorte eines Haushalts in einem Bild findet, z. B. Hinterhof, Küche und Terrasse. Die Trainings- und Testbilder stellen einen einzelnen Standort dar. Jedes Bild ist mit einer einzigen Bezeichnung auf Bildebene beschriftet, z. B. Küche, Terrasse oder Wohnraum. Für ein analysiertes Bild gibt das trainierte Modell eine oder mehrere übereinstimmende Labels aus dem Satz von Labels auf Bildebene zurück, die für das Training verwendet wurden. Beispielsweise könnte das Modell in der folgenden Abbildung die Bezeichnung Wohnraum finden. Weitere Informationen finden Sie unter Objekte, Szenen und Konzepte finden.

Bildklassifizierung (mehrere Label)

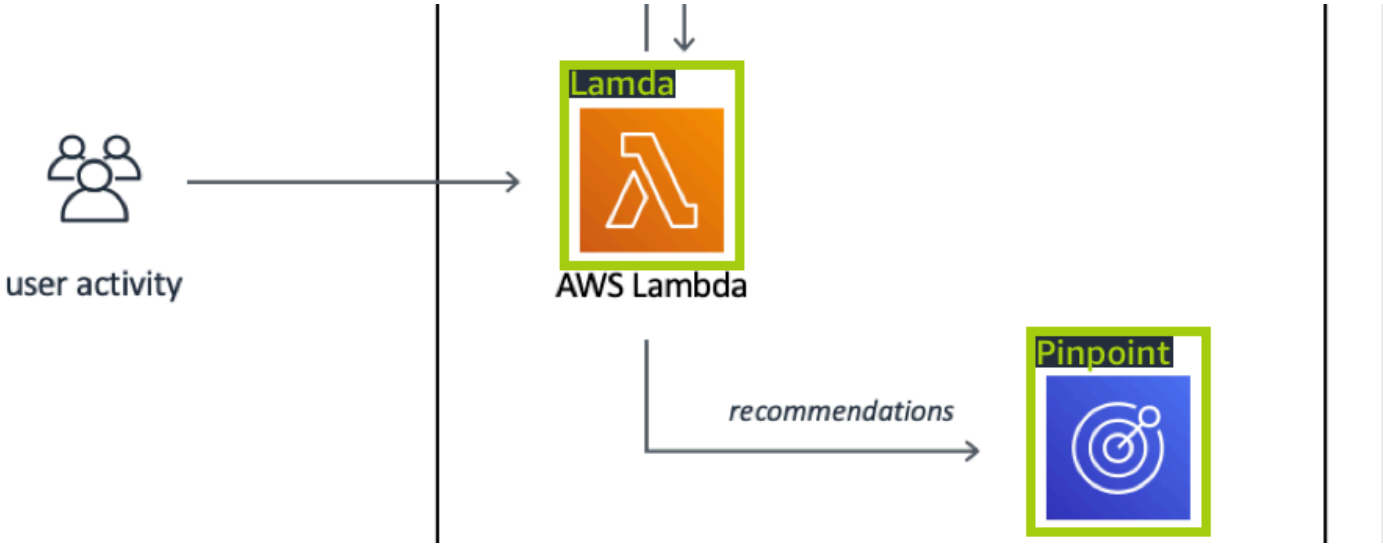
Das Projekt zur Klassifizierung von Bildern mit mehreren Bezeichnungen (Blumen) trainiert ein Modell, das Bilder von Blumen in drei Kategorien unterteilt (Blütentyp, Blattpräsenz und Wachstumsphase).

Die Trainings- und Testbilder verfügen über Labels auf Bildebene für jedes Konzept, z. B. Kamelie für einen Blumentyp, mit_Blättern für eine Blume mit Blättern und ausgewachsen für eine Blume, die ausgewachsen ist.

Für ein analysiertes Bild gibt das trainierte Modell übereinstimmende Labels aus dem Satz von Labels auf Bildebene zurück, die für das Training verwendet wurden. Das Modell gibt beispielsweise die Labels Mediterranean_Spurge und mit_Blättern für das folgende Bild zurück. Weitere Informationen finden Sie unter Objekte, Szenen und Konzepte finden.

Erkennung von Marken

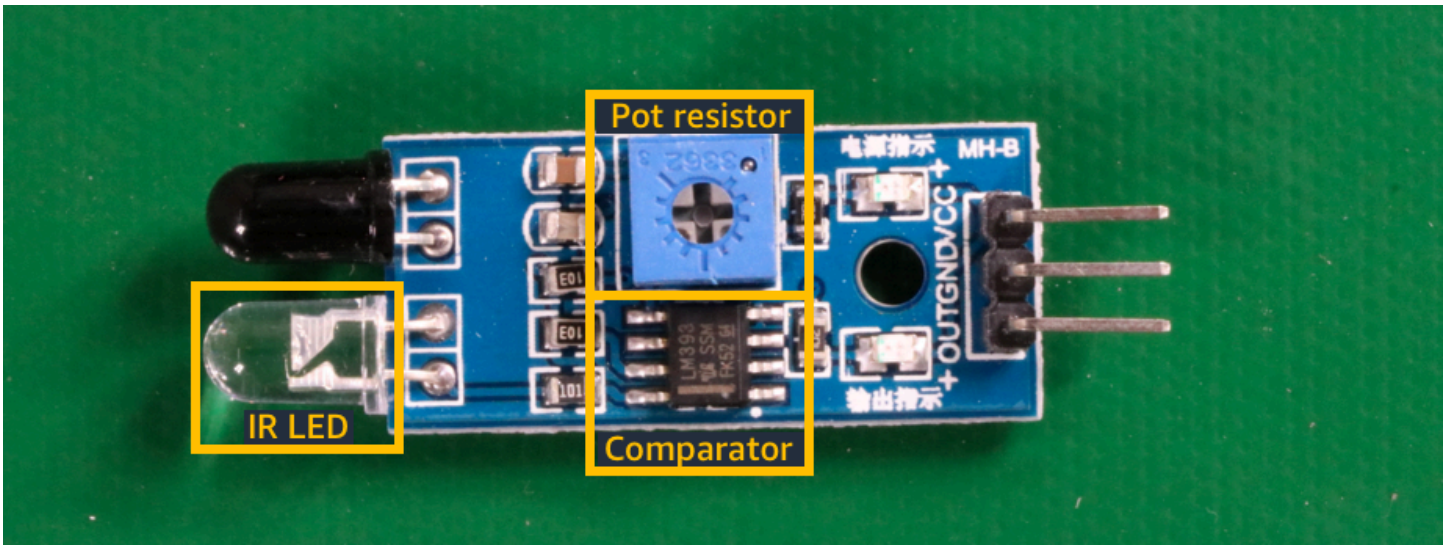
Das Brand Detection Project (Logos) trainiert ein Modell, das die Position bestimmter AWS Logos wie Amazon Textract und AWS Lambda findet. Die Trainingsbilder zeigen nur das Logo und haben ein einziges Label auf Bildebene, z. B. Lambda oder Textract. Es ist auch möglich, ein Modell zur Markenerkennung mit Trainingsbildern zu trainieren, die Begrenzungsrahmen für Markenpositionen enthalten. Die Testbilder verfügen über mit Labeln versehene Begrenzungsrahmen, die die Position von Logos an natürlichen Orten darstellen, z. B. in einem Architekturdiagramm. Das trainierte Modell findet die Logos und gibt für jedes gefundene Logo einen mit Labeln versehenen Begrenzungsrahmen zurück. Weitere Informationen finden Sie unter Marken finden.



Lokalisierung von Objekten

Das Projekt zur Objektlokalisierung (Leiterplatten) trainiert ein Modell, das die Position von Teilen auf einer Leiterplatte findet, z. B. eines Komparators oder einer Infrarot-Leuchtdiode. Die Trainings- und Testbilder enthalten Begrenzungsrahmen, die die Leiterplattenteile umgeben, und ein Label,

das das Teil innerhalb des Begrenzungsrahmens identifiziert. Im folgenden Beispielbild lauten die Labelnamen ir_phototransistor, ir_led, pot_resistor und comparator. Das trainierte Modell findet die Leiterplattenteile und gibt für jedes gefundene Schaltungsteil eine mit Label versehene Begrenzung zurück. Weitere Informationen finden Sie unter Nach Objektpositionen suchen.



Verwenden der Beispielprojekte

In dieser Anleitung „Erste Schritte“ erfahren Sie, wie Sie ein Modell anhand von Beispielprojekten trainieren, die Amazon Rekognition Custom Labels für Sie erstellt. Außerdem erfahren Sie, wie Sie das Modell starten und es zur Analyse eines Bilds verwenden.

Erstellen des Beispielprojekts

Entscheiden Sie zunächst, welches Projekt Sie verwenden möchten. Weitere Informationen finden Sie unter Schritt 1: Auswählen eines Beispielprojekts.

Amazon Rekognition Custom Labels verwendet Datensätze, um ein Modell zu trainieren und zu bewerten (zu testen). Ein Datensatz verwaltet Bilder und die Labels, die den Inhalt von Bildern identifizieren. Die Beispielprojekte umfassen einen Trainingsdatensatz und einen Testdatensatz, in dem alle Bilder mit Labels versehen sind. Sie müssen keine Änderungen vornehmen, bevor Sie Ihr Modell trainieren. Die Beispielprojekte zeigen die zwei Möglichkeiten, wie Amazon Rekognition Custom Labels Labels verwendet, um verschiedene Modelltypen zu trainieren.

- Bildebene — Das Label identifiziert ein Objekt, eine Szene oder ein Konzept, das das gesamte Bild repräsentiert.

- Begrenzungsrahmen — Das Label identifiziert den Inhalt eines Begrenzungsrahmens. Ein Begrenzungsrahmen ist ein Satz von Bildkoordinaten, die ein Objekt in einem Bild umgeben.

Wenn Sie später ein Projekt mit Ihren eigenen Bildern erstellen, müssen Sie Trainings- und Testdatensätze erstellen und Ihre Bilder auch mit Labels versehen. Weitere Informationen finden Sie unter Ihren Modelltyp festlegen.

Trainieren des Modells

Nachdem Amazon Rekognition Custom Labels das Beispielprojekt erstellt hat, können Sie das Modell trainieren. Weitere Informationen finden Sie unter Schritt 2: Trainieren Ihres Modells. Nach Abschluss des Trainings bewerten Sie normalerweise die Leistung des Modells. Die Bilder im Beispieldatensatz erzeugen bereits ein Hochleistungsmodell, und Sie müssen das Modell nicht bewerten, bevor Sie es ausführen. Weitere Informationen finden Sie unter Verbessern eines geschulten Amazon Rekognition Custom Labels-Modells.

Verwenden des Datenmodells

Als Nächstes starten Sie das Modell. Weitere Informationen finden Sie unter Schritt 3: Starten Ihres Modells.

Nachdem Sie mit der Ausführung Ihres Modells begonnen haben, können Sie es verwenden, um neue Bilder zu analysieren. Weitere Informationen finden Sie unter Schritt 4: Analysieren eines Bildes mit Ihrem Modell.

Ihnen wird die Zeit in Rechnung gestellt, während der Ihr Modell ausgeführt wird. Wenn Sie das Beispielmodell nicht mehr verwenden, sollten Sie das Modell stoppen. Weitere Informationen finden Sie unter Schritt 5: Stoppen Ihres Modells.

Nächste Schritte

Sobald Sie bereit sind, können Sie Ihre eigenen Projekte erstellen. Weitere Informationen finden Sie unter Schritt 6: Nächste Schritte.

Schritt 1: Auswählen eines Beispielprojekts

In diesem Schritt wählen Sie ein Beispielprojekt aus. Amazon Rekognition Custom Labels erstellt dann ein Projekt und einen Datensatz für Sie. Ein Projekt verwaltet die Dateien, die zum Trainieren Ihres Modells verwendet werden. Weitere Informationen finden Sie unter Verwalten eines Amazon

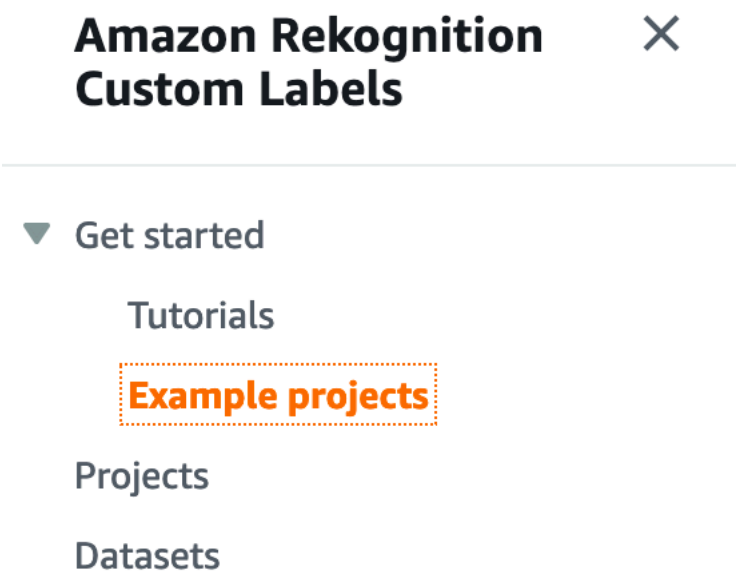
Rekognition Custom Labels-Projekts. Datensätze enthalten die Bilder, zugewiesenen Labels und Begrenzungsrahmen, die Sie zum Trainieren und Testen eines Modells verwenden. Weitere Informationen finden Sie unter the section called “Verwalten von Datensätzen”.

Weitere Informationen zu dieser Beispielprojekte finden Sie unter Beispielprojekte.

Auswählen eines Beispielprojekts

1. Melden Sie sich bei der Amazon Rekognition Rekognition-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie im linken Navigationsbereich Custom Labels verwenden aus. Die Landingpage von Amazon Rekognition Custom Labels wird angezeigt. Wenn die Option Benutzerdefinierte Labels verwenden nicht angezeigt wird, überprüfen Sie, ob die AWS Region, die Sie verwenden, Amazon Rekognition Custom Labels unterstützt.
3. Wählen Sie Erste Schritte aus.

   Der Abschnitt Amazon Rekognition Custom Labels enthält Erste Schritte, Tutorials mit hervorgehobenen „Beispielprojekten“, Projekten und Datensätzen.

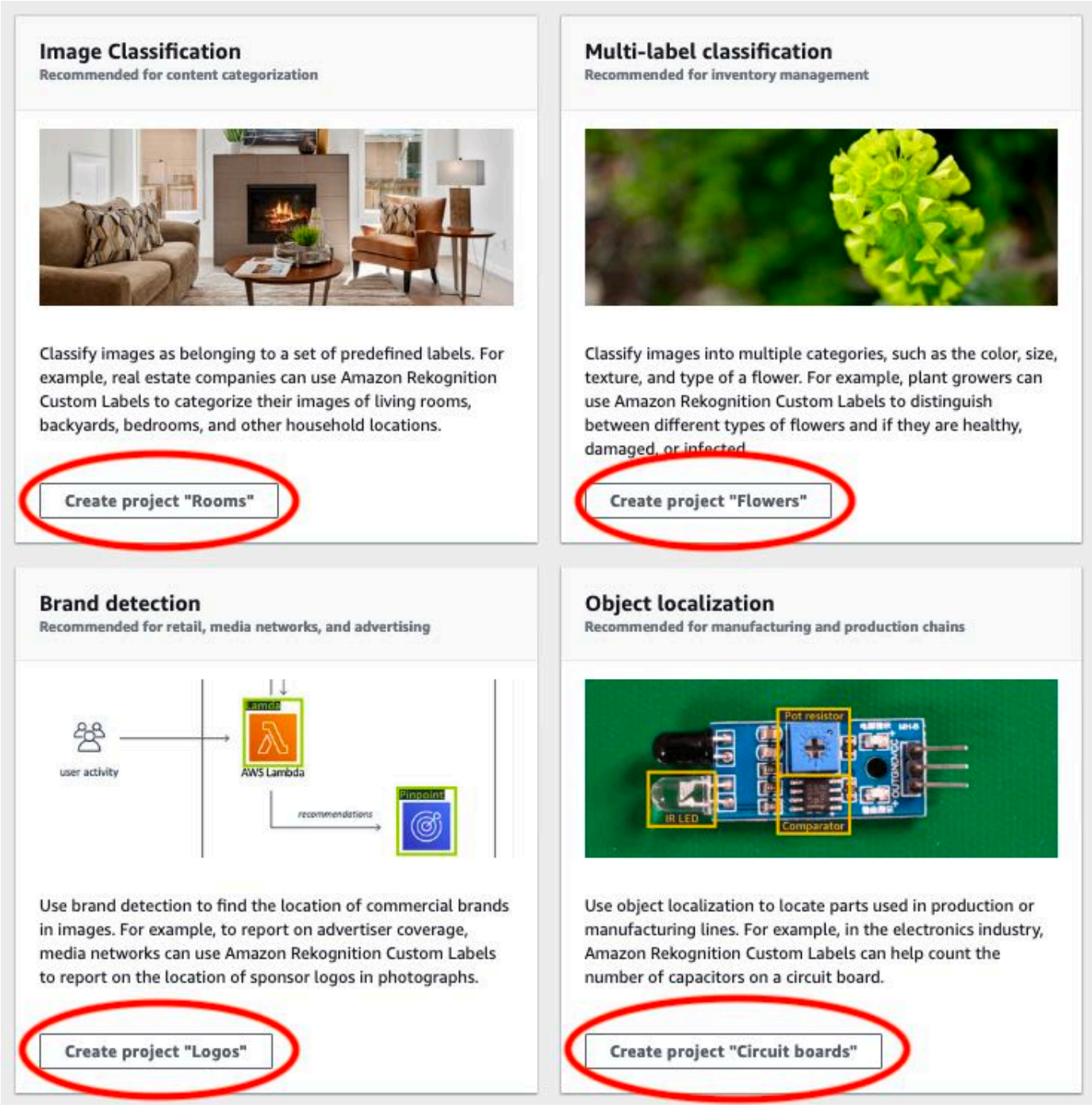


4. Wählen Sie im Bereich Beispielprojekte durchsuchen die Option Beispielprojekte ausprobieren aus.
5. Entscheiden Sie, welches Projekt Sie verwenden möchten, und wählen Sie im Beispielbereich Projekt erstellen „***Projektname***“ aus. Amazon Rekognition Custom Labels erstellt ein neues Projekt für Sie.

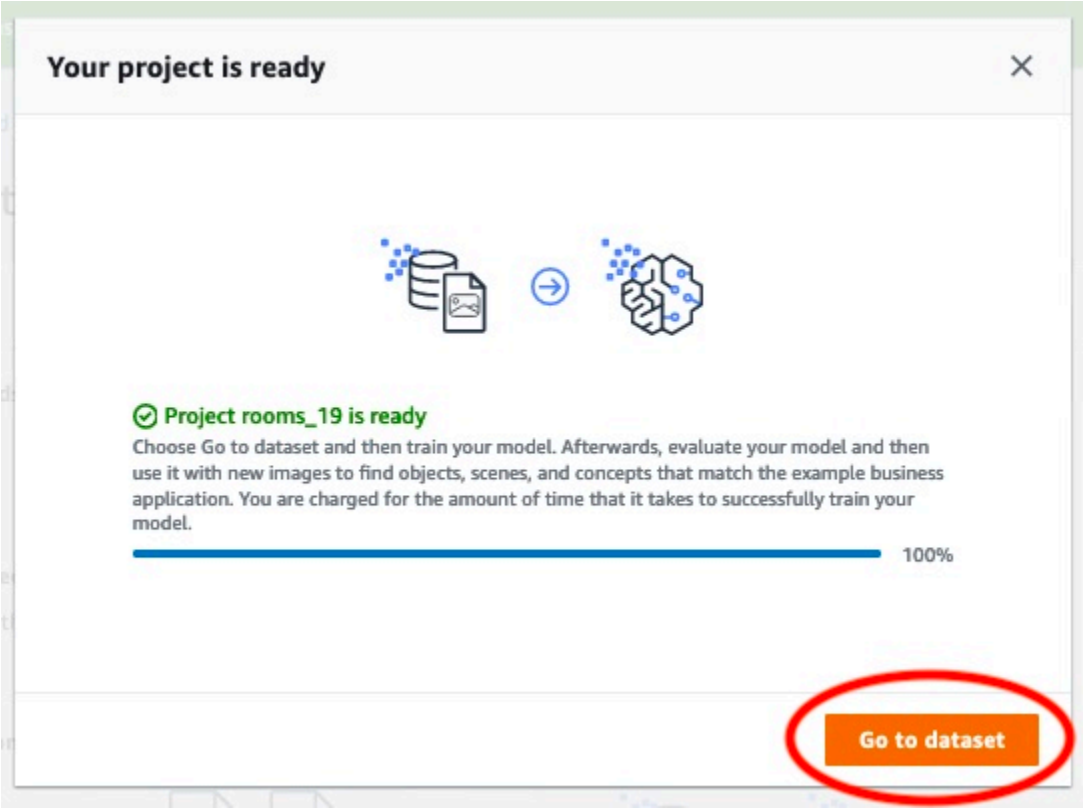
Note

Wenn Sie die Konsole in der aktuellen AWS Region zum ersten Mal öffnen, wird das Dialogfeld „Erste Einrichtung“ angezeigt. Gehen Sie wie folgt vor:

1. Notieren Sie sich den Namen des Amazon-S3-Buckets, der angezeigt wird.
2. Wählen Sie Weiter, damit Amazon Rekognition Custom Labels in Ihrem Namen einen Amazon-S3-Bucket (Konsolen-Bucket) erstellen kann. Das Bild der Konsole unten zeigt Beispiele mit den Schaltflächen „Projekt erstellen“ für Bildklassifizierung (Räume), Klassifizierung mehrerer Labels (Blumen), Markenerkennung (Logos) und Objektlokalisierung (Leiterplatten).



6. Wenn Ihr Projekt fertig ist, wählen Sie Zum Datensatz gehen aus. Die folgende Abbildung zeigt, wie das Projektfenster aussieht, wenn das Projekt fertig ist.

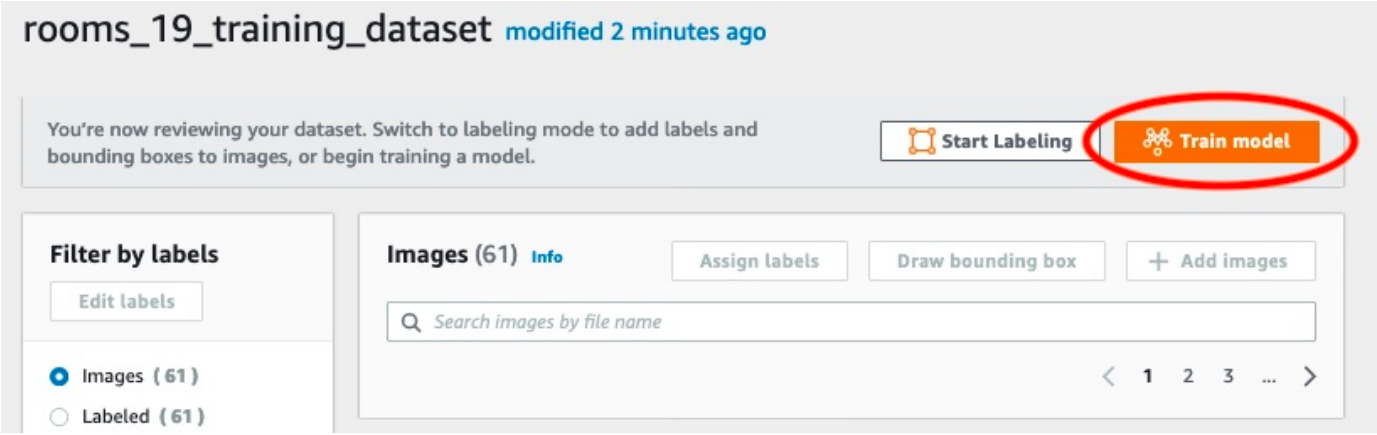


Schritt 2: Trainieren Ihres Modells

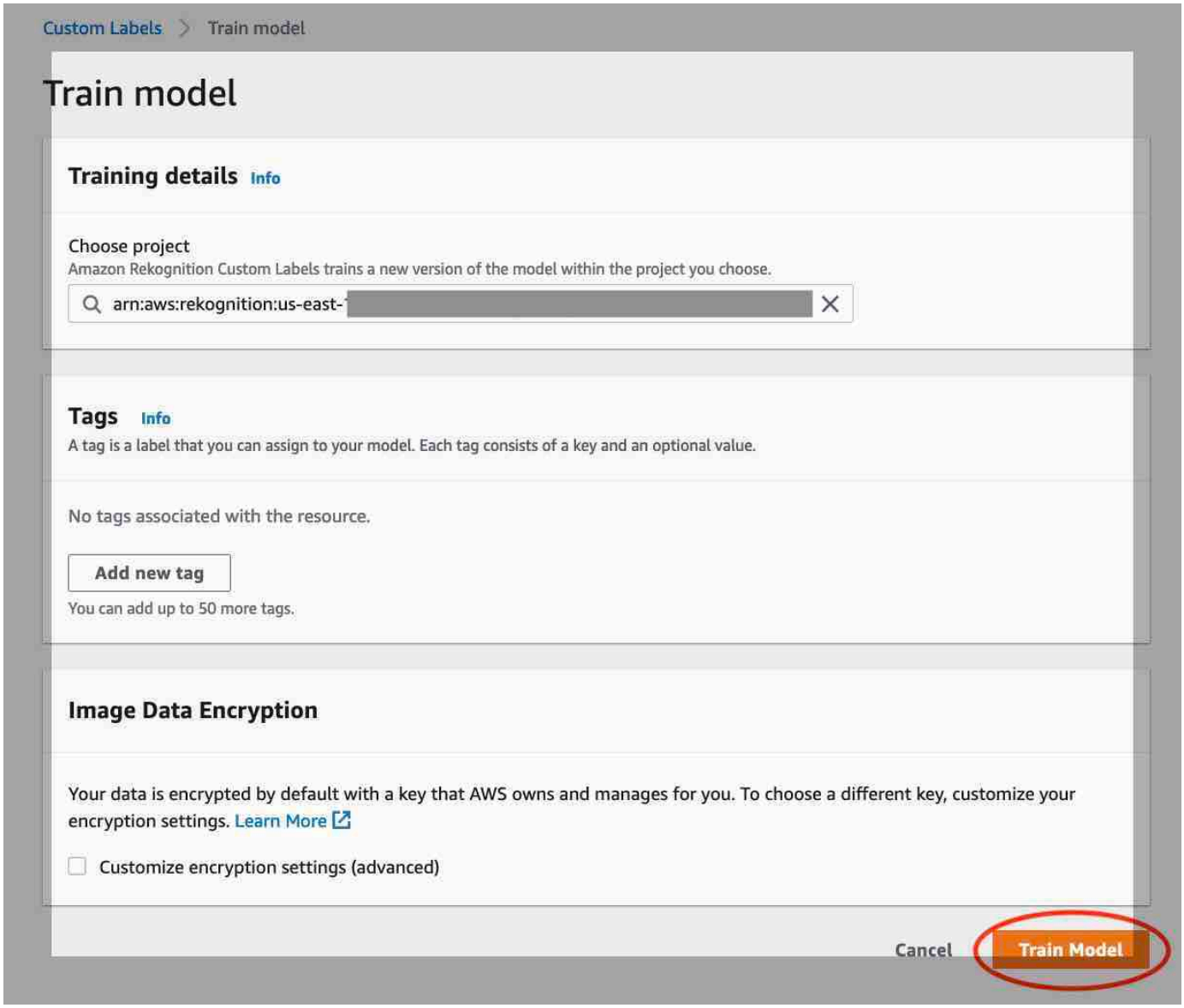
In diesem Schritt trainieren Sie Ihr Modell. Die Trainings- und Testdatensätze werden automatisch für Sie konfiguriert. Nach erfolgreichem Abschluss des Trainings können Sie die allgemeinen Bewertungsergebnisse und die Bewertungsergebnisse für einzelne Testbilder sehen. Weitere Informationen finden Sie unter Schulung eines Amazon-Rekognition-Custom-Labels-Modells.

So trainieren Sie Ihr Modell

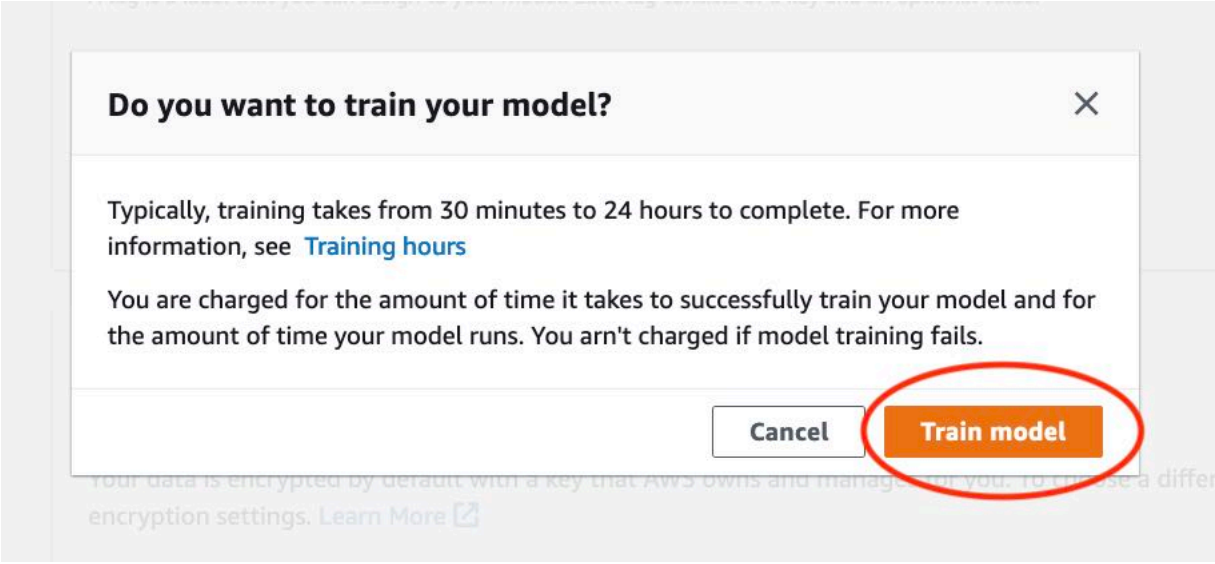
1. Wählen Sie auf der Datensatzseite das Modell Train aus. Die folgende Abbildung zeigt die Konsole mit der Schaltfläche für das Zugmodell.



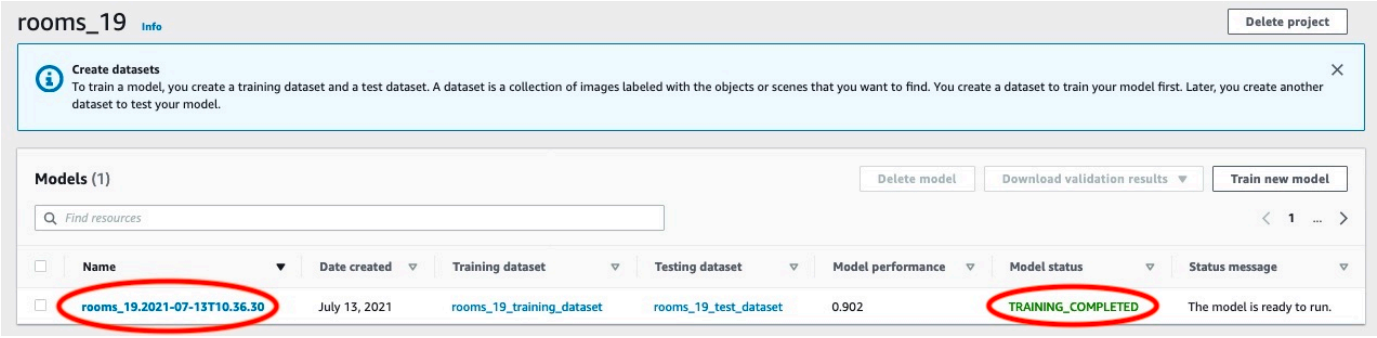
2. Wählen Sie auf der Seite Modell trainieren die Option Modell trainieren aus. Das Bild unten zeigt die Schaltfläche Modell trainieren. Beachten Sie, dass sich der Amazon-Ressourcenname (ARN) für Ihr Projekt im Bearbeitungsfeld Projekt auswählen befindet.



3. Auf der Seite Möchten Sie Ihr Modell trainieren? Wählen Sie im Dialogfeld, das in der folgenden Abbildung gezeigt wird, die Option Modell trainieren aus.

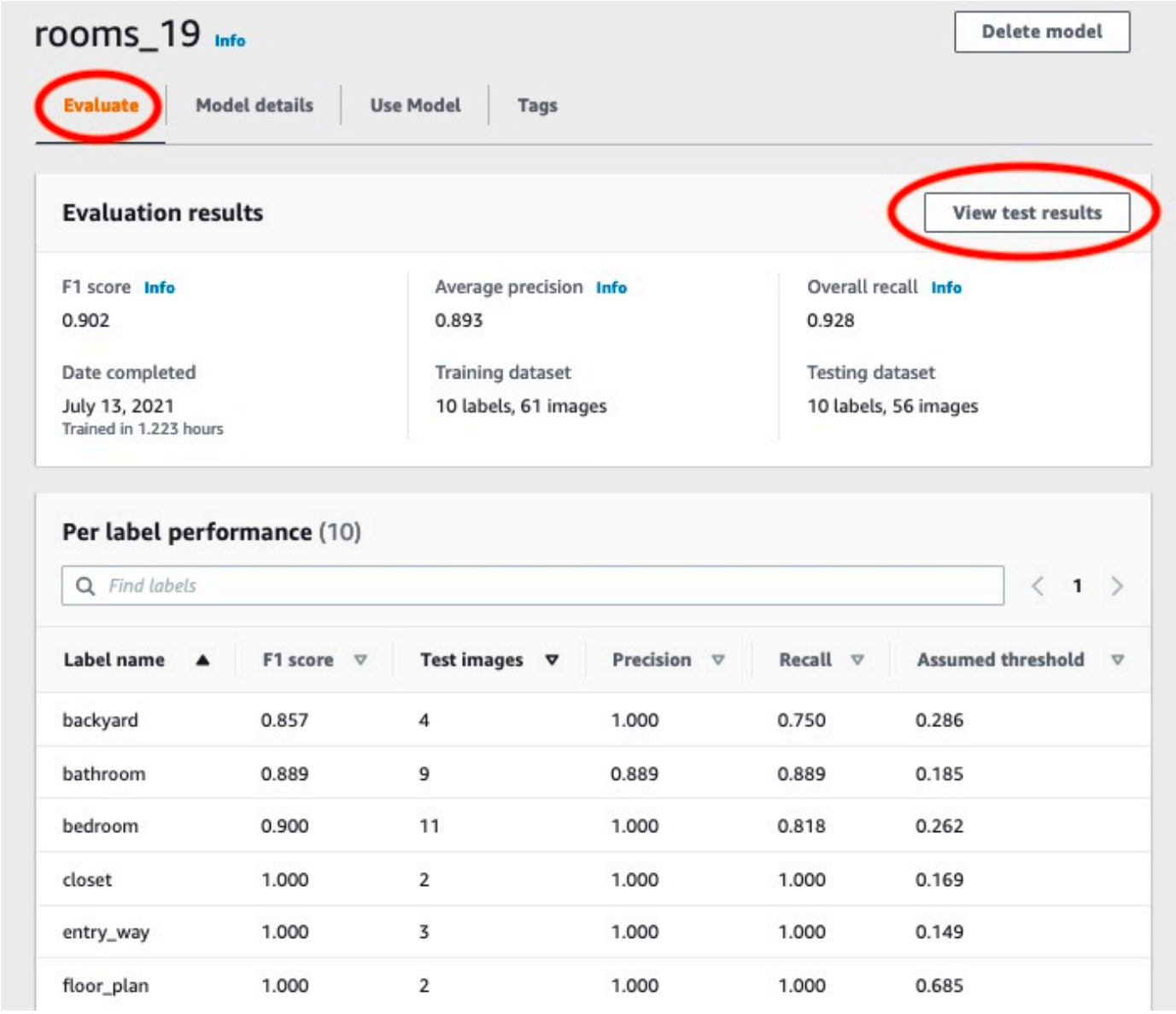


4. Wählen Sie nach Abschluss des Trainings den Modellnamen aus. Das Training ist abgeschlossen, wenn der Modellstatus TRAINING_COMPLETED lautet, wie im folgenden Konsolen-Screenshot gezeigt.

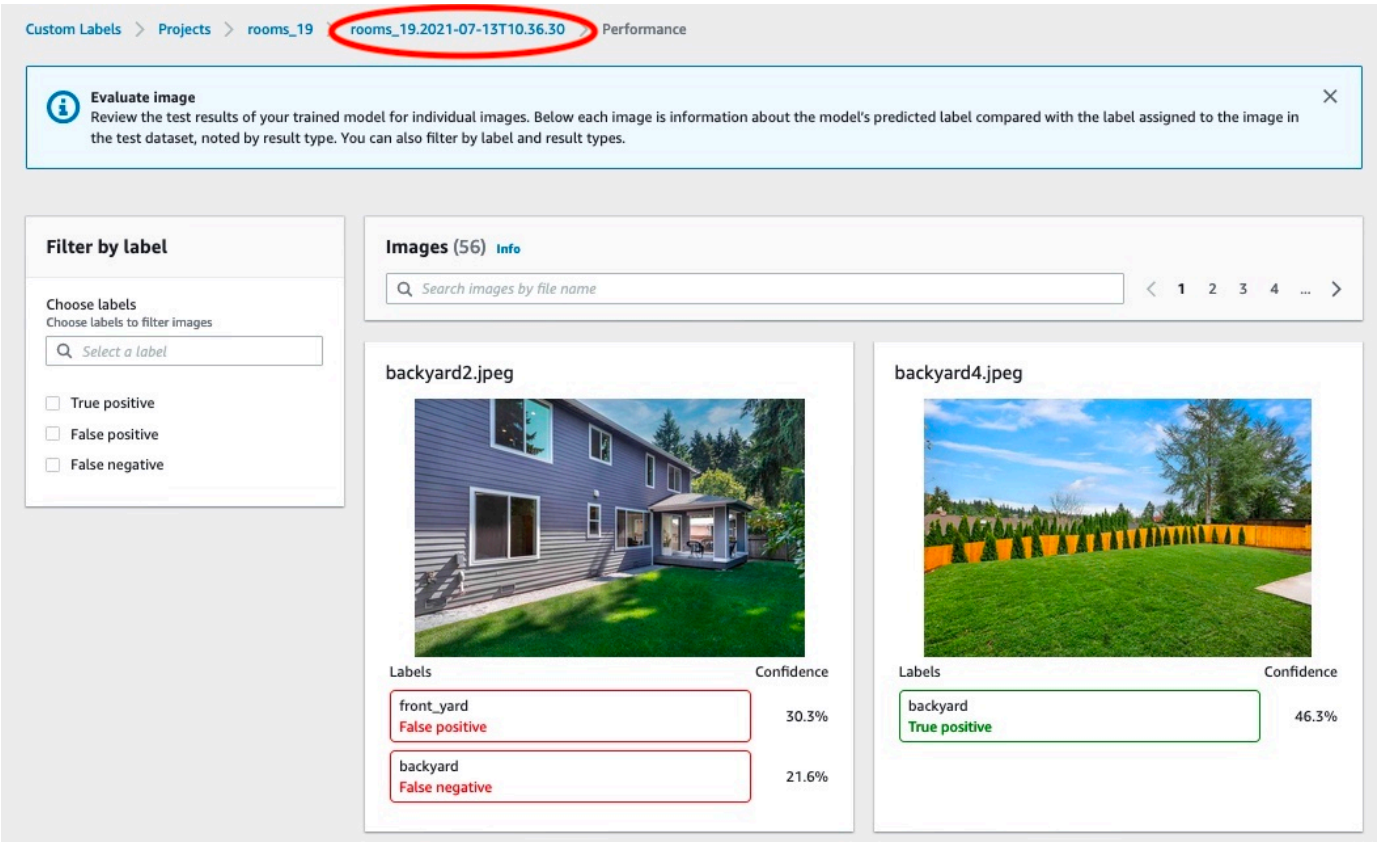


5. Wählen Sie die Schaltfläche Bewerten, um die Bewertungsergebnisse anzuzeigen. Weitere Informationen zum Bewerten eines Modells finden Sie unter Verbessern eines geschulten Amazon Rekognition Custom Labels-Modells.

6. Wählen Sie Testergebnisse anzeigen, um die Ergebnisse für einzelne Testbilder anzuzeigen. Wie im folgenden Screenshot zu sehen ist, zeigt das Bewertungs-Dashboard Kennzahlen wie F1-Ergebnis, Präzision und Erinnerungsvermögen für jedes Label zusammen mit der Anzahl der Testbilder. Allgemeine Kennzahlen wie Durchschnitt, Genauigkeit und Erinnerungsvermögen werden ebenfalls angezeigt.



7. Nachdem Sie sich die Testergebnisse angesehen haben, wählen Sie den Modellnamen, um zur Modellseite zurückzukehren. Der folgende Screenshot des Leistungs-Dashboards, auf den Sie klicken können, um zur Modellseite zurückzukehren.



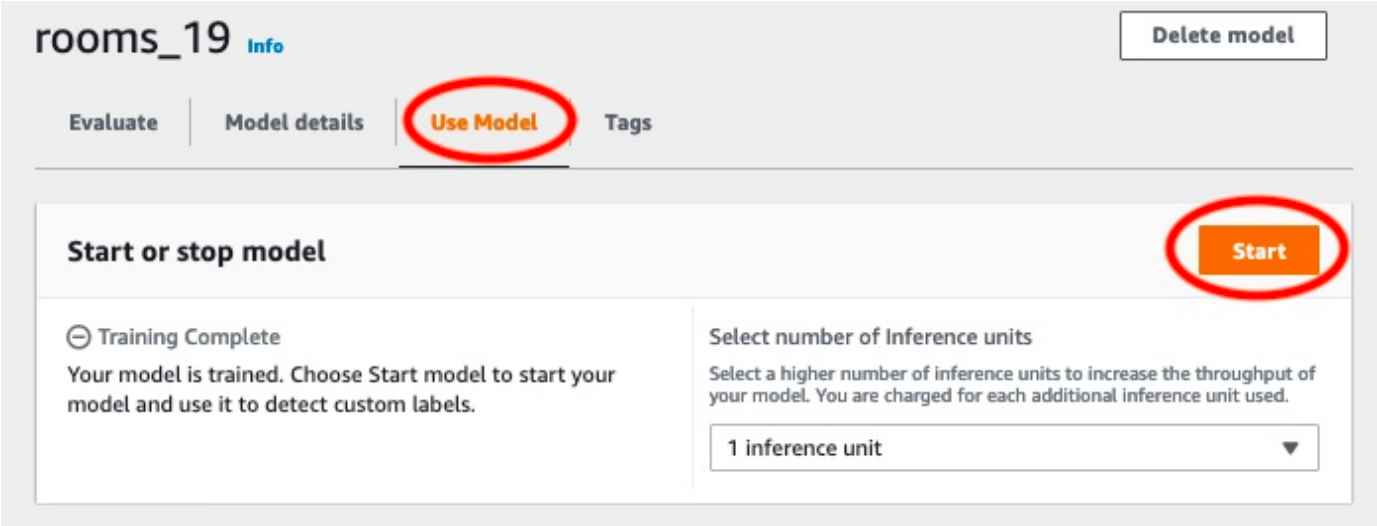
Schritt 3: Starten Ihres Modells

In diesem Schritt starten Sie Ihr Modell. Nachdem Ihr Modell gestartet ist, können Sie es zur Analyse von Bildern verwenden.

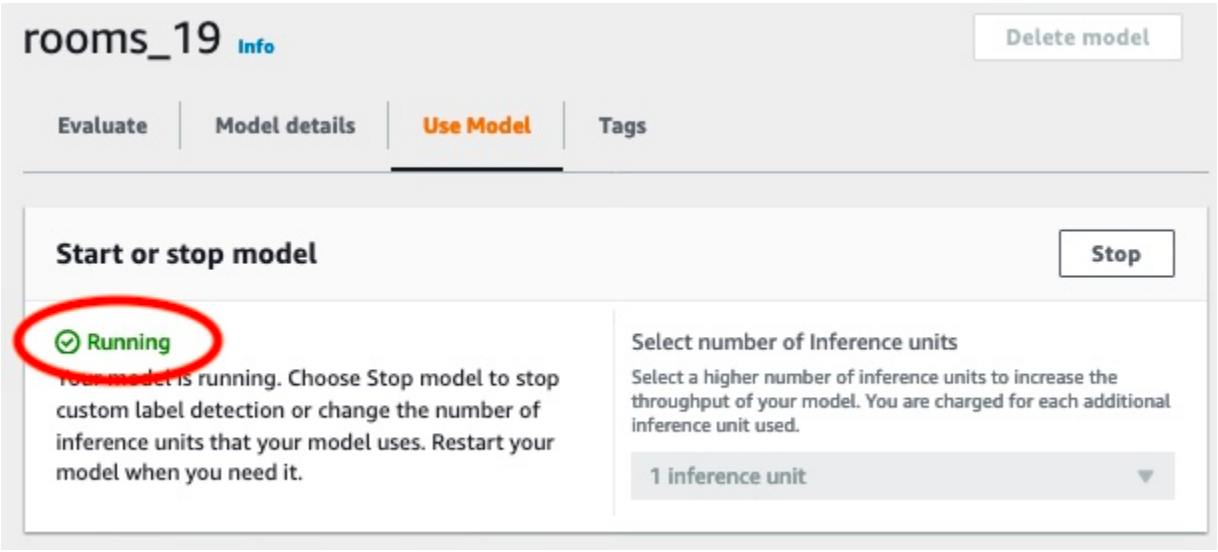
Ihnen wird die Zeit in Rechnung gestellt, während der Ihr Modell ausgeführt wird. Stoppen Sie Ihr Modell, wenn Sie keine Bilder analysieren müssen. Sie können Ihr Modell zu einem späteren Zeitpunkt neu starten. Weitere Informationen finden Sie unter Ausführen eines trainierten Amazon Rekognition Custom Labels-Modells.

So starten Sie Ihr Modell

1. Wählen Sie auf der Modellseite die Registerkarte Modell verwenden.
2. Gehen Sie im Abschnitt Modell starten oder stoppen wie folgt vor:
   a. Wählen Sie Starten.
   b. Wählen Sie im Dialogfeld Modell starten die Option Starten aus. Die folgende Abbildung zeigt die Schaltfläche Start im Bedienfeld des Modells.



3. Warten Sie, bis das Modell läuft. Der folgende Screenshot zeigt die Konsole, während das Modell läuft, wobei der Status im Abschnitt Modell starten oder beenden den Status Running lautet.



4. Verwenden Sie Ihr Modell, um Bilder zu klassifizieren. Weitere Informationen finden Sie unter Schritt 4: Analysieren eines Bildes mit Ihrem Modell.

Schritt 4: Analysieren eines Bildes mit Ihrem Modell

Sie analysieren ein Bild, indem Sie die DetectCustomLabels API aufrufen. In diesem Schritt verwenden Sie den Befehl `detect-custom-labels` AWS Command Line Interface (AWS CLI), um ein Beispielbild zu analysieren. Sie erhalten den AWS CLI Befehl von der Amazon Rekognition

Custom Labels-Konsole. Die Konsole konfiguriert den AWS CLI Befehl so, dass er Ihr Modell verwendet. Sie müssen nur ein Bild bereitstellen, das in einem Amazon-S3-Bucket gespeichert ist. Dieses Thema enthält ein Bild, das Sie für jedes Beispielprojekt verwenden können.

> **Note**
>
> Die Konsole bietet auch Python-Beispielcode.

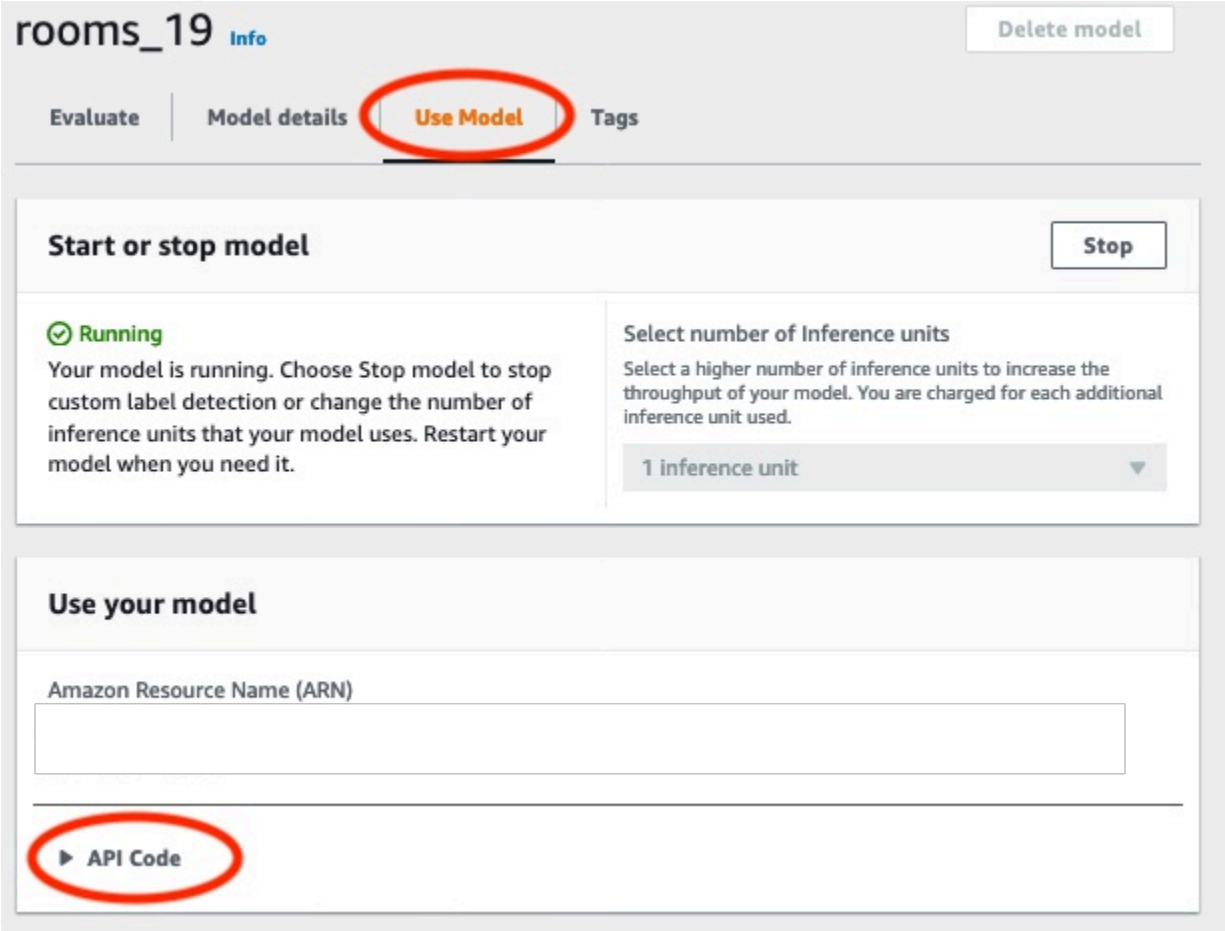
Die Ausgabe von `detect-custom-labels` umfasst eine Liste der im Bild gefundenen Labels, Begrenzungsrahmen (wenn das Modell Objektpositionen findet) und das Vertrauen, das das Modell in die Genauigkeit der Vorhersagen hat.

Weitere Informationen finden Sie unter Analysieren eines Bildes mit einem trainierten Modell.

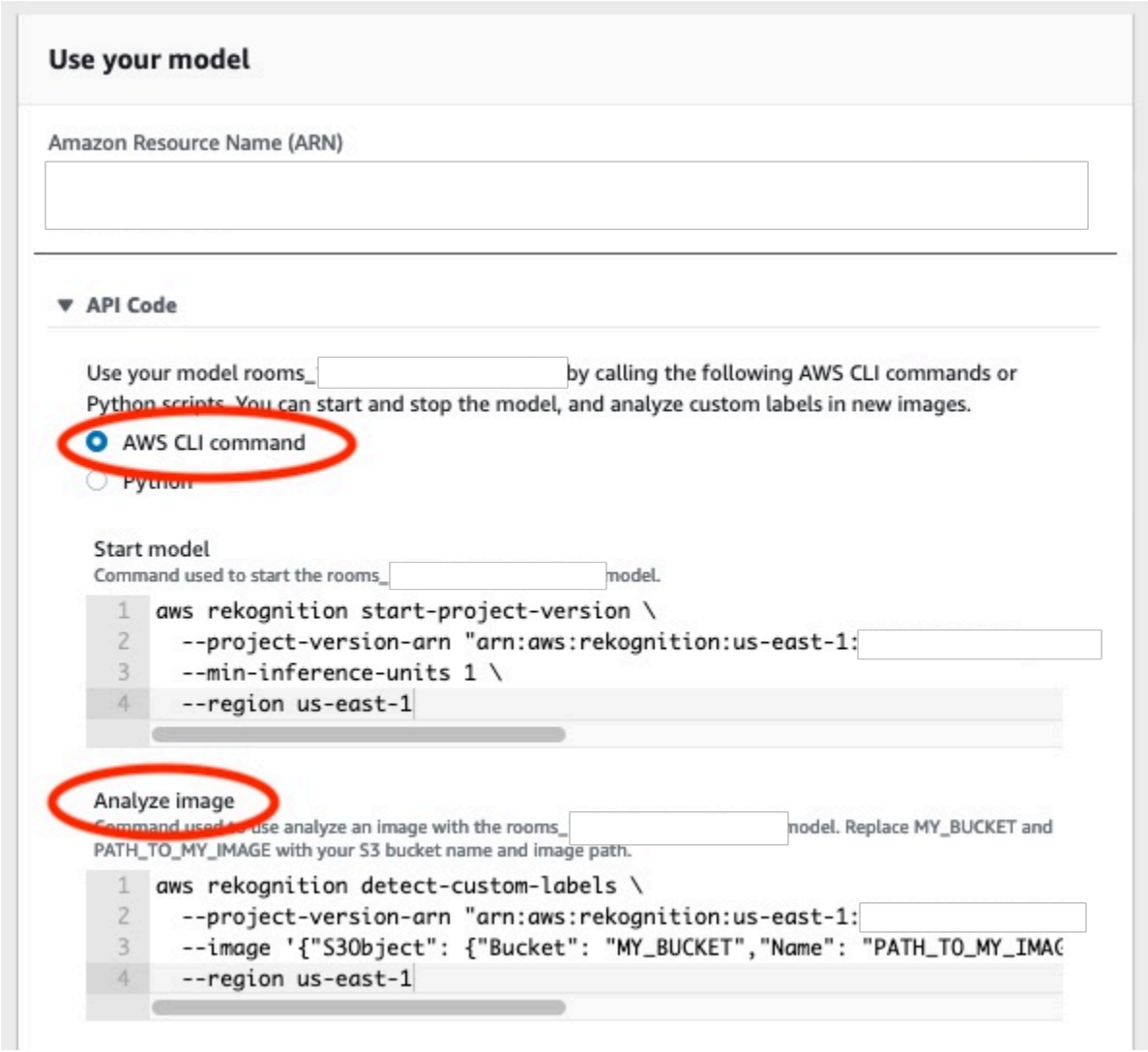
So analysieren Sie ein Bild (Konsole)

1. <textobject><phrase>Der Modellstatus wird als Wird ausgeführt angezeigt, mit der Schaltfläche Stopp wird das laufende Modell angehalten. </phrase></textobject>

   Wenn Sie es noch nicht getan haben, richten Sie das ein AWS CLI. Anweisungen finden Sie unter the section called “Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS”.
2. Wenn Sie es noch nicht getan haben, starten Sie Ihr Modell. Weitere Informationen finden Sie unter Schritt 3: Starten Ihres Modells.
3. Wählen Sie die Registerkarte Modell verwenden und wählen Sie dann API-Code aus. Im unten abgebildeten Modellstatusfenster wird das Modell als Wird ausgeführt angezeigt. Es gibt eine Stopp-Schaltfläche, um das laufende Modell zu stoppen, und eine Option zum Anzeigen der API.



4. Wählen Sie den AWS CLI-Befehl.
5. Kopieren Sie im Abschnitt Bild analysieren den AWS CLI Befehl, der den Aufruf ausführt`detect-custom-labels`. Die folgende Abbildung der Rekognition-Konsole zeigt den Abschnitt „Bild analysieren“ mit dem AWS-CLI-Befehl zur Erkennung benutzerdefinierter Labels auf einem Bild mithilfe eines Machine-Learning-Modells sowie Anweisungen zum Starten des Modells und zur Bereitstellung von Bilddetails.



6. Laden Sie ein Beispielbild in einen Amazon-S3-Bucket hoch. Anweisungen finden Sie unter Ein Beispielbild erhalten.
7. Geben Sie in der Befehlszeile den AWS CLI Befehl ein, den Sie im vorherigen Schritt kopiert haben. Es sollte wie das folgende Beispiel aussehen.

   Der Wert von `--project-version-arn` sollte der Amazon-Ressourcenname (ARN) Ihres Modells sein. Der Wert von `--region` sollte der AWS Region entsprechen, in der Sie das Modell erstellt haben.

   Ändern Sie `MY_BUCKET` und `PATH_TO_MY_IMAGE` in den Amazon-S3-Bucket und das Bild, das Sie im vorherigen Schritt verwendet haben.

   Wenn Sie das Custom-Labels-Access-Profil verwenden, um Anmeldeinformationen abzurufen, fügen Sie den Parameter `--profile custom-labels-access` hinzu.

```
aws rekognition detect-custom-labels \
  --project-version-arn "model_arn" \
  --image '{"S3Object": {"Bucket": "MY_BUCKET","Name": "PATH_TO_MY_IMAGE"}}' \
  --region us-east-1 \
  --profile custom-labels-access
```

Wenn das Modell Objekte, Szenen und Konzepte findet, sollte die JSON-Ausgabe des AWS CLI -Befehls etwa wie folgt aussehen. `Name` ist der Name des Labels auf Bildebene, das das Modell gefunden hat. `Confidence` (0-100) ist das Vertrauen des Modells in die Genauigkeit der Vorhersage.

```
{
    "CustomLabels": [
        {
            "Name": "living_space",
            "Confidence": 83.41299819946289
        }
    ]
}
```

Wenn das Modell Objektpositionen oder Marken findet, werden mit Labeln versehene Begrenzungsrahmen zurückgegeben. `BoundingBox` enthält die Position eines Rahmens, den das Objekt umgibt. `Name` ist das Objekt, das das Modell in dem Begrenzungsrahmen gefunden hat. `Confidence` ist die Gewissheit des Modells, dass der Begrenzungsrahmen das Objekt enthält.

```
{
    "CustomLabels": [
        {
            "Name": "textract",
            "Confidence": 87.7729721069336,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.198987677693367,
                    "Height": 0.31296101212501526,
                    "Left": 0.07924537360668182,
                    "Top": 0.4037395715713501
                }
            }
        }
```

```
    ]
}
```

8. Verwenden Sie das Modell weiterhin, um andere Bilder zu analysieren. Stoppen Sie das Modell, wenn Sie es nicht mehr verwenden. Weitere Informationen finden Sie unter Schritt 5: Stoppen Ihres Modells.

Ein Beispielbild erhalten

Sie können die folgenden Bilder für den `DetectCustomLabels`-Vorgang verwenden. Für jedes Projekt gibt es ein Bild. Um die Bilder zu verwenden, laden Sie sie in einen S3-Bucket hoch.

So verwenden Sie ein Beispielbild

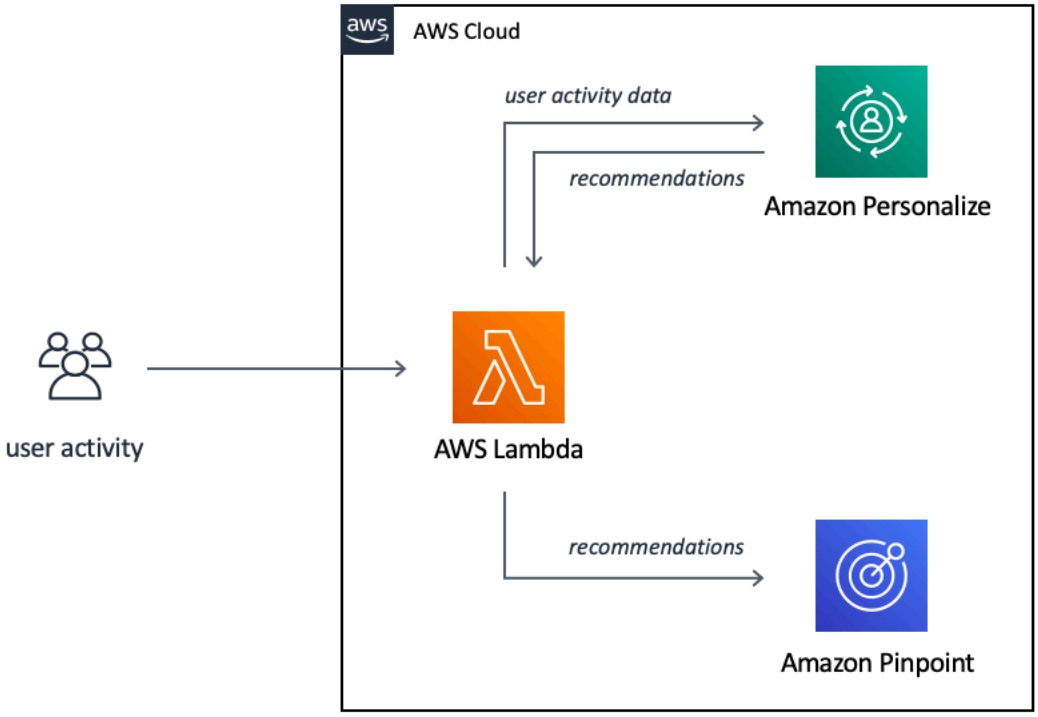
1. Klicken Sie mit der rechten Maustaste auf das folgende Bild, das dem Beispielprojekt entspricht, das Sie verwenden. Wählen Sie dann Bild speichern, um das Bild auf Ihrem Computer zu speichern. Je nachdem, welchen Browser Sie verwenden, kann die Menüoption unterschiedlich sein.
2. Laden Sie das Bild in einen Amazon S3 S3-Bucket hoch, der Ihrem AWS Konto gehört und sich in derselben AWS Region befindet, in der Sie Amazon Rekognition Custom Labels verwenden.

   Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service.

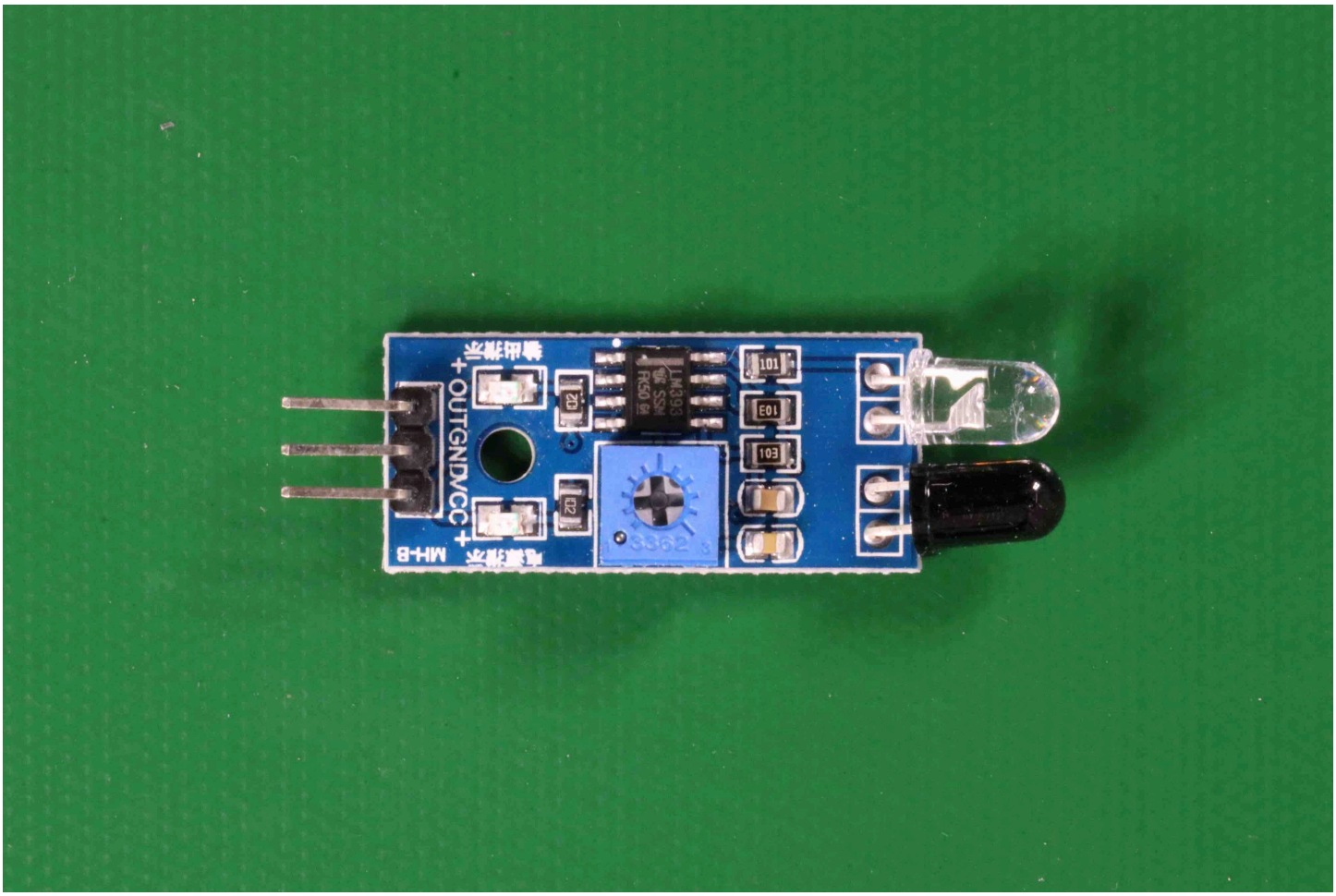
Bildklassifizierung

Klassifizierung mit mehreren Labels

Erkennung von Marken



Lokalisierung von Objekten

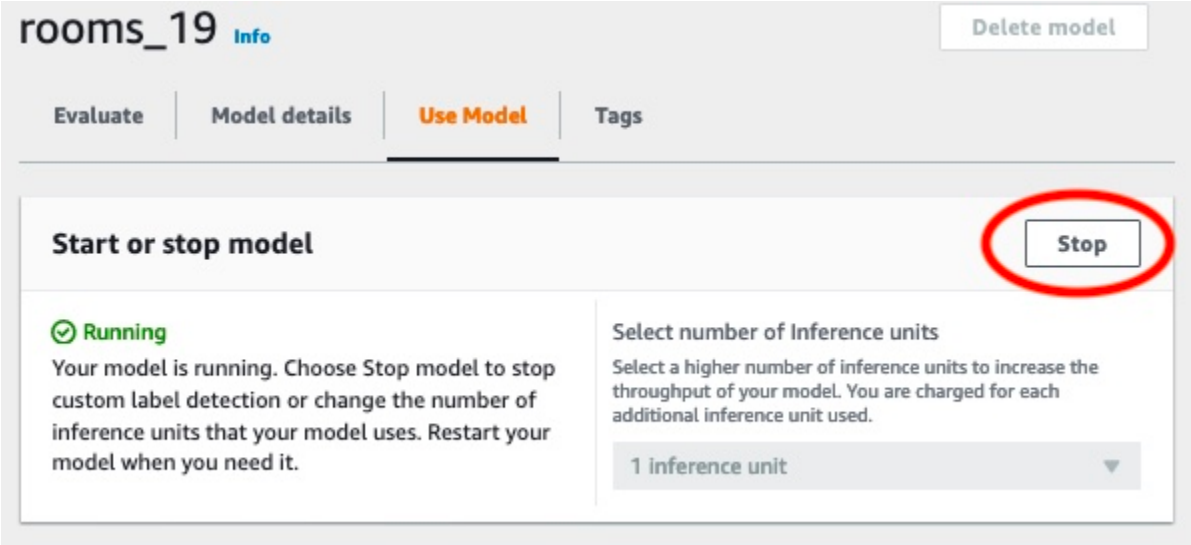


Schritt 5: Stoppen Ihres Modells

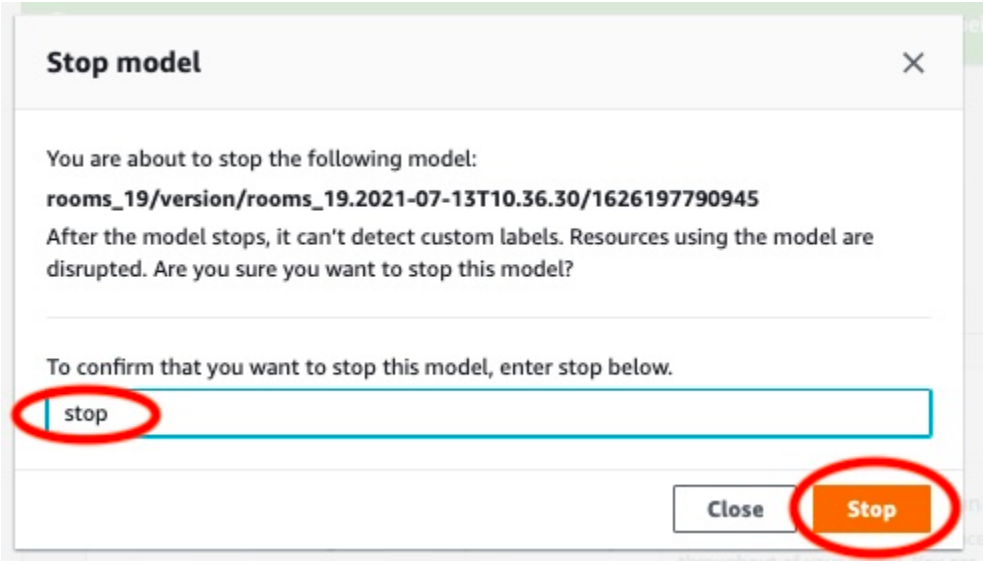
In diesem Schritt stoppen Sie die Ausführung Ihres Modells. Ihnen wird die Zeit in Rechnung gestellt, während der Ihr Modell ausgeführt wird. Wenn Sie das Modell nicht mehr verwenden, sollten Sie es stoppen.

So stoppen Sie Ihr Modell

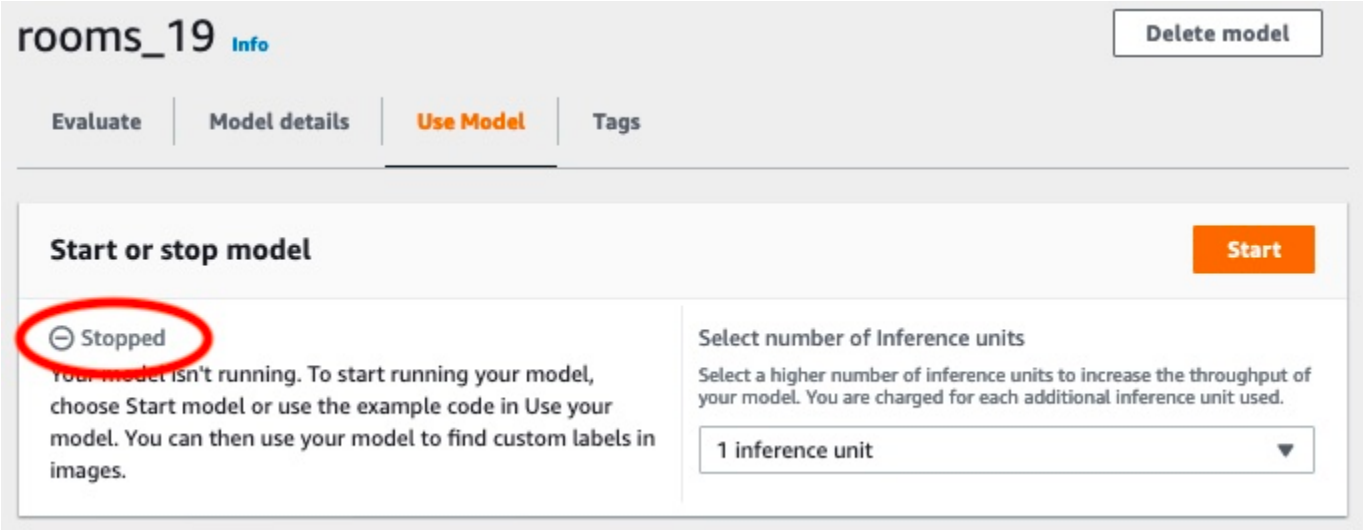
1. Wählen Sie im Abschnitt Modell starten oder stoppen die Option Stoppen.



2. Geben Sie im Dialogfeld Modell stoppen Stoppen ein, um zu bestätigen, dass Sie das Modell stoppen möchten.



3. Wählen Sie Stoppen, um das Modell zu stoppen. Das Modell wurde gestoppt, wenn der Status im Abschnitt Modell starten oder stoppen Gestoppt lautet. Im folgenden Screenshot bietet der Bereich Benutzeroberfläche die Option, ein Modell für maschinelles Lernen zu starten oder zu beenden. Der Status des Modells wird als „Gestoppt“ angezeigt, mit einer Schaltfläche „Start“, um das Modell zu starten, und einem Drop-down-Menü, mit dem die Anzahl der Inferenzeinheiten ausgewählt werden kann.



Schritt 6: Nächste Schritte

Nachdem Sie die Beispielprojekte ausprobiert haben, können Sie Ihre eigenen Bilder und Datensätze verwenden, um Ihr eigenes Modell zu erstellen. Weitere Informationen finden Sie unter Grundlegendes zu Amazon Rekognition Custom Labels.

Verwenden Sie die Label-Informationen in der folgenden Tabelle, um Modelle zu trainieren, die den Beispielprojekten ähneln.

Beispiel	Trainingsbilder	Testbilder
Bildklassifizierung (Zimmer)	1 Label auf Bildebene pro Bild	1 Label auf Bildebene pro Bild
Klassifizierung mit mehreren Labels (Blumen)	Mehrere Labels auf Bildebene pro Bild	Mehrere Labels auf Bildebene pro Bild
Markenerkennung (Logos)	Labels auf Bildebene (Sie können auch mit Labeln versehene Begrenzun gsrahmen verwenden)	Mit Labeln versehene Begrenzungsrahmen
Bildlokalisierung (Leiterplatten)	Mit Labeln versehene Begrenzungsrahmen	Mit Labeln versehene Begrenzungsrahmen

Das Tutorial: Bilder klassifizieren zeigt Ihnen, wie Sie ein Projekt, Datensätze und Modelle für ein Bildklassifizierungsmodell erstellen.

Ausführliche Informationen zum Erstellen von Datensätzen und Trainingsmodellen finden Sie unter. Erstellen eines Amazon Rekognition-Custom-Custom-Label-Erstellen eines neuen

Tutorial: Bilder klassifizieren

Dieses Tutorial zeigt Ihnen, wie Sie das Projekt und die Datensätze für ein Modell erstellen, das Objekte, Szenen und Konzepte in einem Bild klassifiziert. Das Modell klassifiziert das gesamte Bild. Wenn Sie beispielsweise diesem Tutorial folgen, können Sie einem Modell beibringen, Haushaltsstandorte wie ein Wohnzimmer oder eine Küche zu erkennen. Das Tutorial zeigt Ihnen auch, wie Sie das Modell zur Analyse von Bildern verwenden.

Bevor Sie mit dem Tutorial beginnen, empfehlen wir Ihnen,Grundlegendes zu Amazon Rekognition Custom Labels.

In diesem Tutorial erstellen Sie die Trainings- und Testdatensätze, indem Sie Bilder von Ihrem lokalen Computer hochladen. Später weisen Sie den Bildern in Ihren Trainings- und Testdatensätzen Labels auf Bildebene zu.

Das von Ihnen erstellte Modell klassifiziert Bilder als zu dem Satz von Bezeichnungen auf Bildebene gehörend, den Sie den Bildern des Trainingsdatensatzes zuweisen. Wenn der Satz von Labels auf Bildebene in Ihrem Trainingsdatensatz beispielsweise`kitchen`,`living_room`,`patio`, und`backyard`, kann das Modell potenziell all diese Beschriftungen auf Bildebene in einem einzigen Bild finden.

> **Note**
>
> Sie können Modelle für verschiedene Zwecke erstellen, z. B. um die Position von Objekten auf einem Bild zu ermitteln. Weitere Informationen finden Sie unter Ihren Modelltyp festlegen.

Schritt 1: Sammle deine Bilder

Sie benötigen zwei Bildersätze. Ein Set, das Sie Ihrem Trainingsdatensatz hinzufügen können. Ein weiteres Set, das Sie Ihrem Testdatensatz hinzufügen können. Die Bilder sollten die Objekte, Szenen und Konzepte darstellen, die Ihr Modell klassifizieren soll. Die Bilder müssen im PNG- oder JPEG-Format vorliegen. Weitere Informationen finden Sie unter Vorbereiten der Bilder.

Du solltest mindestens 10 Bilder für deinen Trainingsdatensatz und 10 Bilder für deinen Testdatensatz haben.

Wenn Sie noch keine Bilder haben, verwenden Sie die Bilder von derZimmerBeispiel für ein Klassifizierungsprojekt. Nach der Erstellung des Projekts befinden sich die Trainings- und Test-Images an den folgenden Amazon S3-Bucket-Standorten:

- Bilder aus dem Training —`s3://custom-labels-console-`*`region`*`-`*`numbers`*`/assets/rooms_`*`version number`*`_test_dataset/`
- Testbilder —`s3://custom-labels-console-`*`region`*`-`*`numbers`*`/assets/rooms_`*`version number`*`_test_dataset/`

`region`ist derAWSRegion, in der Sie die Amazon Rekognition Custom Labels-Konsole verwenden.`numbers`ist ein Wert, den die Konsole dem Bucket-Namen zuweist.`Version number`ist die Versionsnummer für das Beispielprojekt, beginnend bei 1.

Mit dem folgenden Verfahren werden Bilder aus dem Rooms-Projekt in lokalen Ordnern auf Ihrem Computer mit dem Namen gespeichert`training`und`test`.

Um die Beispiel-Projektbilddateien von Rooms herunterzuladen

1. Erstellen Sie das Projekt Rooms. Weitere Informationen finden Sie unter Schritt 1: Auswählen eines Beispielprojekts.
2. Öffnen Sie die Befehlszeile und geben Sie den folgenden Befehl ein, um die Trainingsbilder herunterzuladen.

   ```
   aws s3 cp s3://custom-labels-console-region-numbers/assets/rooms_version
    number_training_dataset/ training --recursive
   ```

3. Geben Sie in der Befehlszeile den folgenden Befehl ein, um die Testbilder herunterzuladen.

   ```
   aws s3 cp s3://custom-labels-console-region-numbers/assets/rooms_version
    number_test_dataset/ test --recursive
   ```

4. Verschieben Sie zwei der Bilder aus dem Trainingsordner in einen separaten Ordner Ihrer Wahl. Sie werden die Bilder verwenden, um Ihr trainiertes Modell in auszuprobierenSchritt 9: Analysieren Sie ein Bild mit Ihrem Modell.

Schritt 2: Wähle deine Klassen

Erstellen Sie eine Liste der Klassen, die Ihr Modell finden soll. Wenn Sie beispielsweise einem Modell beibringen, Räume in einem Haus zu erkennen, können Sie das folgende Bild klassifizieren als`living_room`.

Jede Klasse ist einem Label auf Bildebene zugeordnet. Später weisen Sie den Bildern in Ihren Trainings- und Testdatensätzen Labels auf Bildebene zu.

Wenn Sie die Bilder aus dem Rooms-Beispielprojekt verwenden, lauten die Beschriftungen auf BildebeneHinterhof,Badezimmer,Schlafzimmer,Kleiderschrank,Eingangsbereich,grundriss,vor_garten,Küche, undTerrasse.

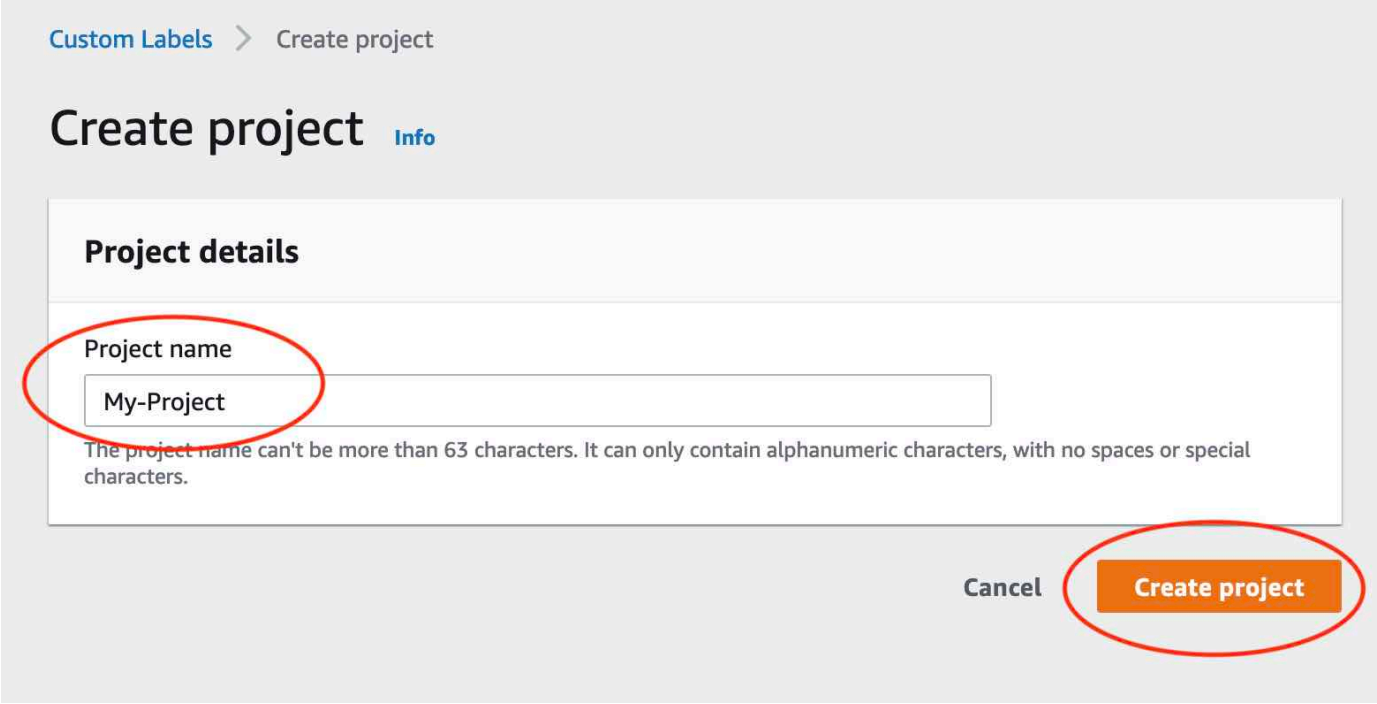
Schritt 3: Erstellen Sie ein Projekt

Um Ihre Datensätze und Modelle zu verwalten, erstellen Sie ein Projekt. Jedes Projekt sollte sich mit einem einzigen Anwendungsfall befassen, z. B. der Erkennung von Räumen in einem Haus.

Um ein Projekt zu erstellen (Konsole)

1. Falls Sie dies noch nicht getan haben, richten Sie die Amazon Rekognition Custom Labels-Konsole ein. Weitere Informationen finden Sie unter Einrichten von Amazon Rekognition Custom Labels.
2. Loggen Sie sich ein beiAWS Management Consoleund öffnen Sie die Amazon Rekognition-Konsole unterhttps://console.aws.amazon.com/rekognition/.
3. Wählen Sie im linken BereichVerwenden Sie benutzerdefinierte Labels. Die Landingpage von Amazon Rekognition Custom Labels wird angezeigt.

4. Wählen Sie auf der Amazon Rekognition Custom Labels-LandingpageFangen Sie an
5. Wählen Sie im linken NavigationsbereichProjekte.
6. Wählen Sie auf der ProjektseiteProjekt erstellen.
7. Geben Sie im Feld Project name (Projektname) einen Namen für Ihr Projekt an.
8. Wählen SieProjekt erstellenum Ihr Projekt zu erstellen.



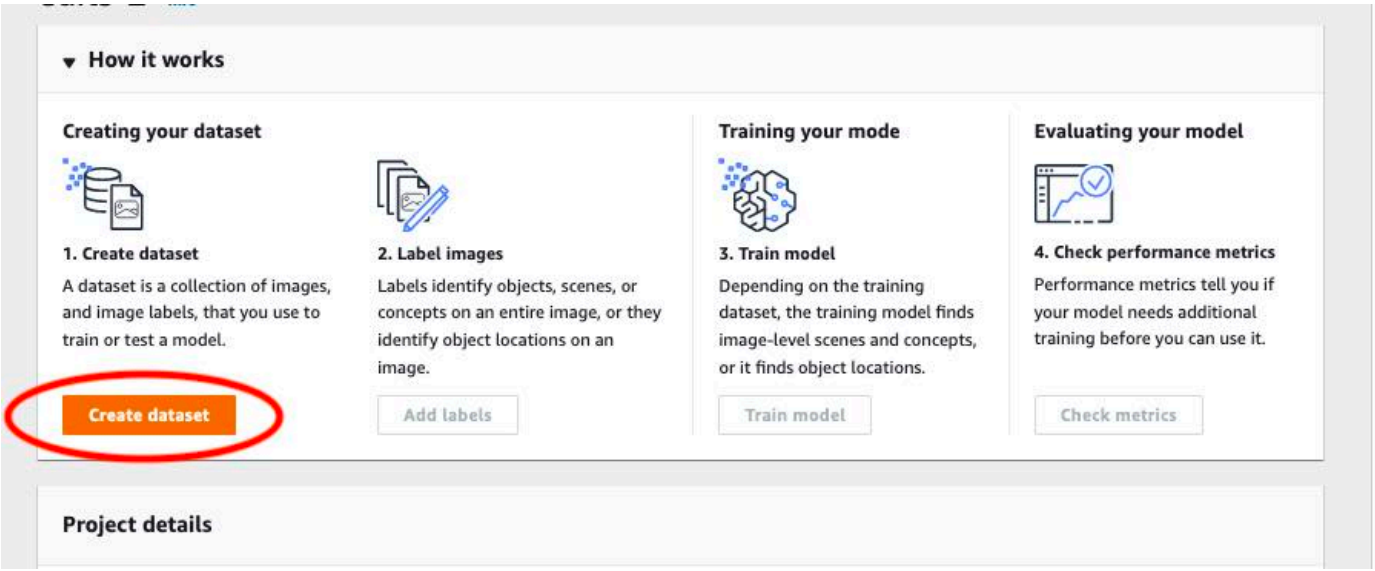
Schritt 4: Trainings- und Testdatensätze erstellen

In diesem Schritt erstellen Sie einen Trainingsdatensatz und einen Testdatensatz, indem Sie Bilder von Ihrem lokalen Computer hochladen. Sie können bis zu 30 Bilder gleichzeitig hochladen. Wenn Sie viele Bilder hochladen müssen, sollten Sie erwägen, die Datensätze zu erstellen, indem Sie die Bilder aus einem Amazon S3-Bucket importieren. Weitere Informationen finden Sie unter Amazon-S3-Bucket.

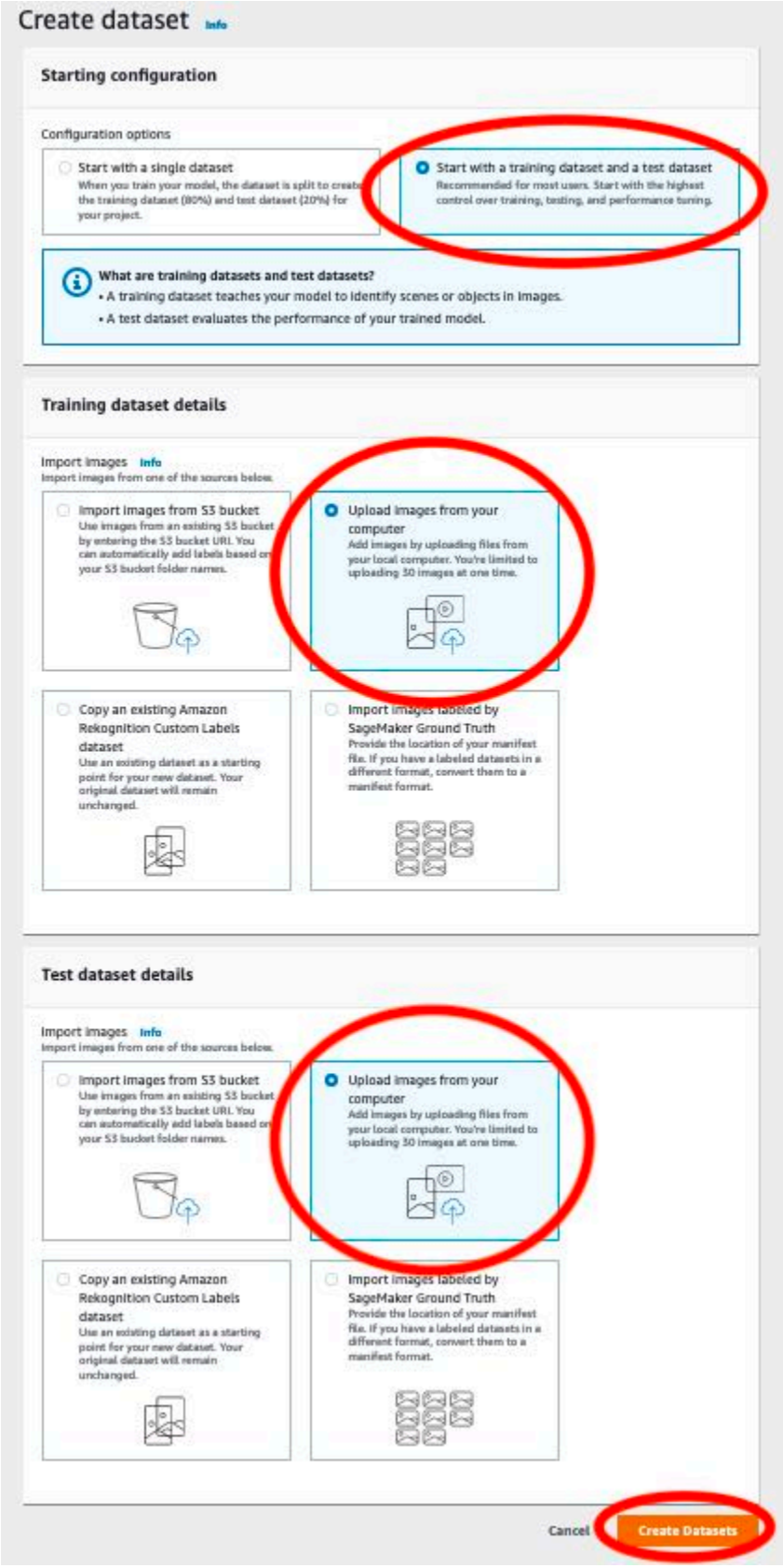
Weitere Informationen zu Datasets finden Sie unter Verwalten von Datensätzen.

Um einen Datensatz mithilfe von Bildern auf einem lokalen Computer (Konsole) zu erstellen

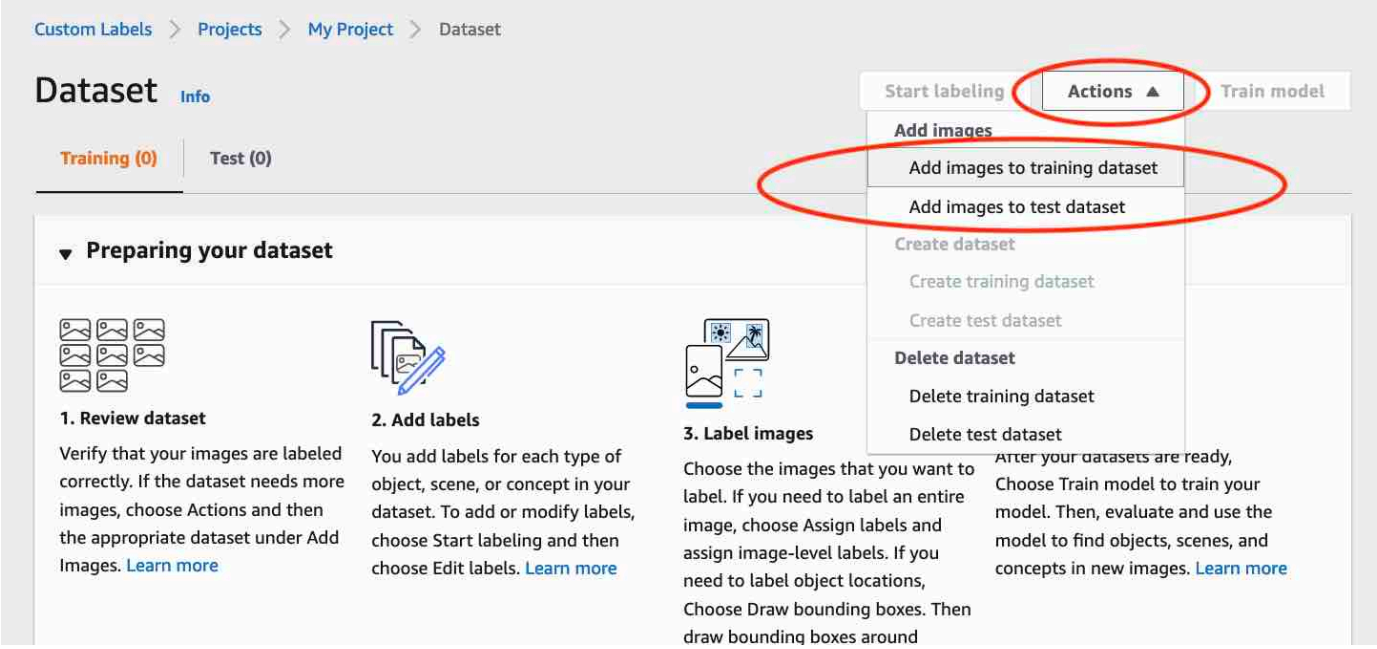
1. Wählen Sie auf der Seite mit den ProjektdetailsDatensatz erstellen.



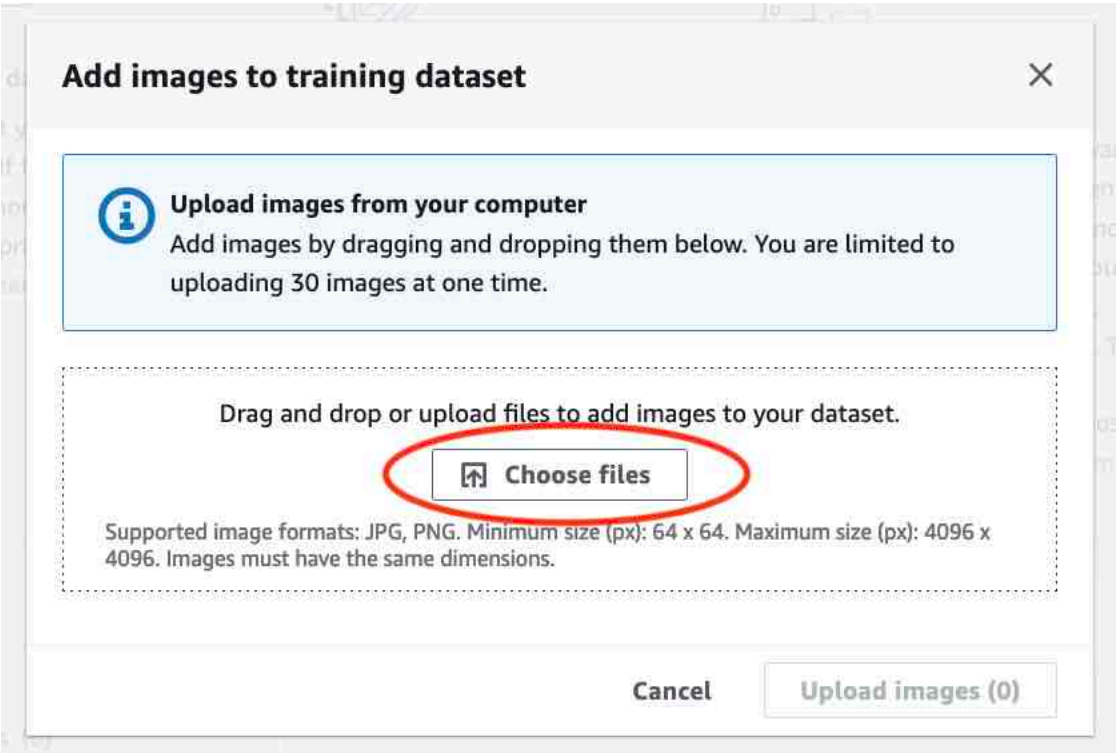
2. In derKonfiguration startenAbschnitt, wählenBeginnen Sie mit einem Trainingsdatensatz und einem Testdatensatz.
3. In derDetails zum TrainingsdatensatzAbschnitt, wählenLaden Sie Bilder von Ihrem Computer hoch.
4. In derDetails zum TestdatensatzAbschnitt, wählenLaden Sie Bilder von Ihrem Computer hoch.
5. WähleDatensätze erstellen.


Create dataset Info
Starting configuration
Configuration options
Start with a single dataset
When you train your model, the dataset is split to create the training dataset (80%) and test dataset (20%) for your project.
Start with a training dataset and a test dataset
Recommended for most users. Start with the highest control over training, testing, and performance tuning.
What are training datasets and test datasets?
• A training dataset teaches your model to identify scenes or objects in images.
• A test dataset evaluates the performance of your trained model.
Training dataset details
Import images Info
Import images from one of the sources below.
Import images from S3 bucket
Use images from an existing S3 bucket by entering the S3 bucket URL. You can automatically add labels based on your S3 bucket folder names.
Upload images from your computer
Add images by uploading files from your local computer. You're limited to uploading 30 images at one time.
Copy an existing Amazon Rekognition Custom Labels dataset
Use an existing dataset as a starting point for your new dataset. Your original dataset will remain unchanged.
Import images labeled by SageMaker Ground Truth
Provide the location of your manifest file. If you have a labeled datasets in a different format, convert them to a manifest format.
Test dataset details
Import images Info
Import images from one of the sources below.
Import images from S3 bucket
Use images from an existing S3 bucket by entering the S3 bucket URL. You can automatically add labels based on your S3 bucket folder names.
Upload images from your computer
Add images by uploading files from your local computer. You're limited to uploading 30 images at one time.
Copy an existing Amazon Rekognition Custom Labels dataset
Use an existing dataset as a starting point for your new dataset. Your original dataset will remain unchanged.
Import images labeled by SageMaker Ground Truth
Provide the location of your manifest file. If you have a labeled datasets in a different format, convert them to a manifest format.
Cancel
Create Datasets

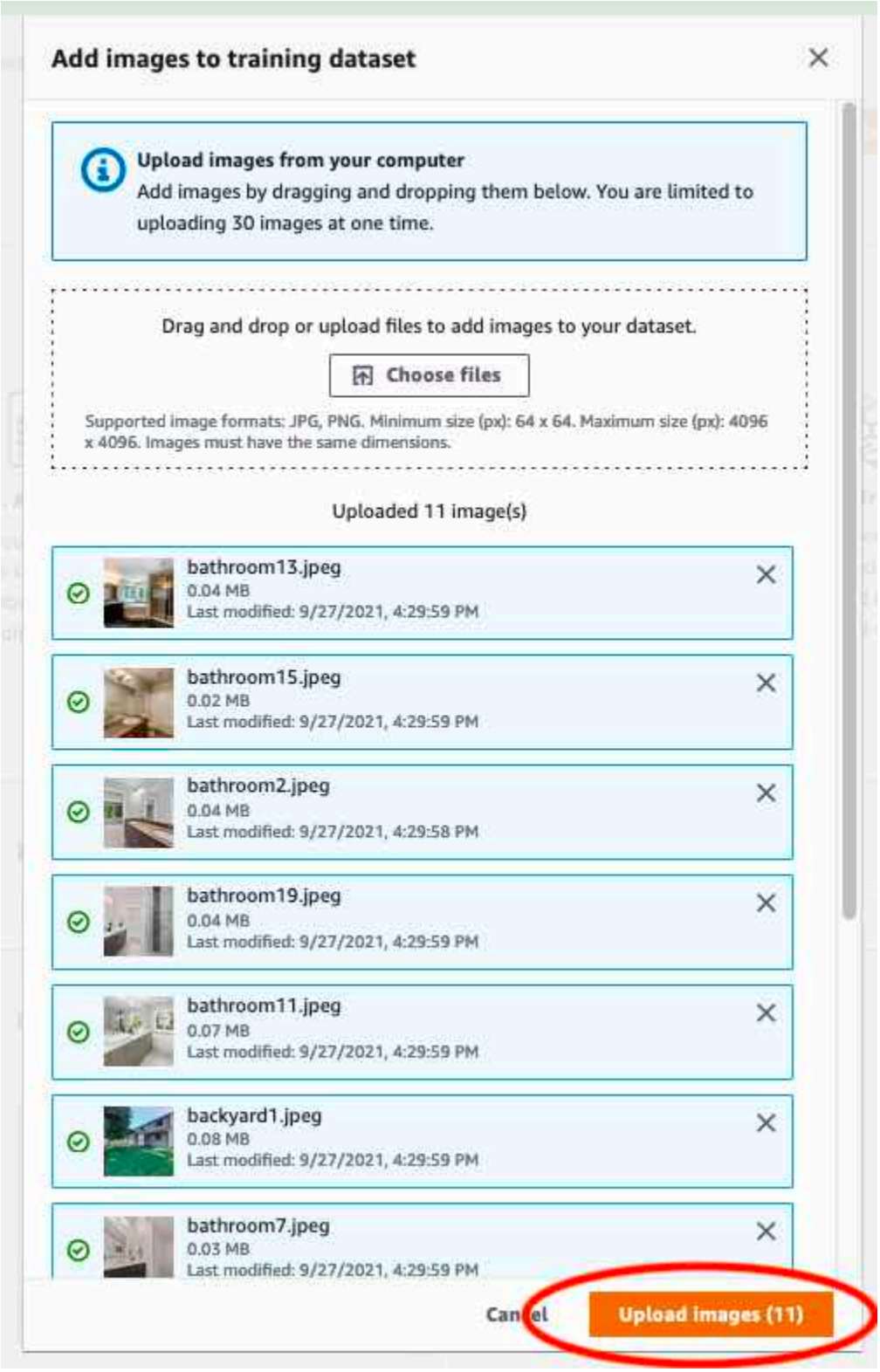
6. Es erscheint eine Datensatzseite mit einemSchulungTab und einTestenTab für die jeweiligen Datensätze.
7. Wählen Sie auf der Datensatzseite dieSchulungTab.
8. Wählen SieAktionenund dann wähleBilder zum Trainingsdatensatz hinzufügen.



9. In derBilder zum Trainingsdatensatz hinzufügenDialogfenster, wählen SieWählen Sie Dateien.



10. Wählen Sie die Bilder aus, die Sie in den Datensatz hochladen möchten. Sie können bis zu 30 Bilder gleichzeitig hochladen.
11. Wählen SieBilder hochladen. Es kann einige Sekunden dauern, bis Amazon Rekognition Custom Labels die Bilder zum Datensatz hinzufügt.



12. Wenn Sie dem Trainingsdatensatz weitere Bilder hinzufügen möchten, wiederholen Sie die Schritte 9—12.
13. Wählen Sie die Registerkarte Test.

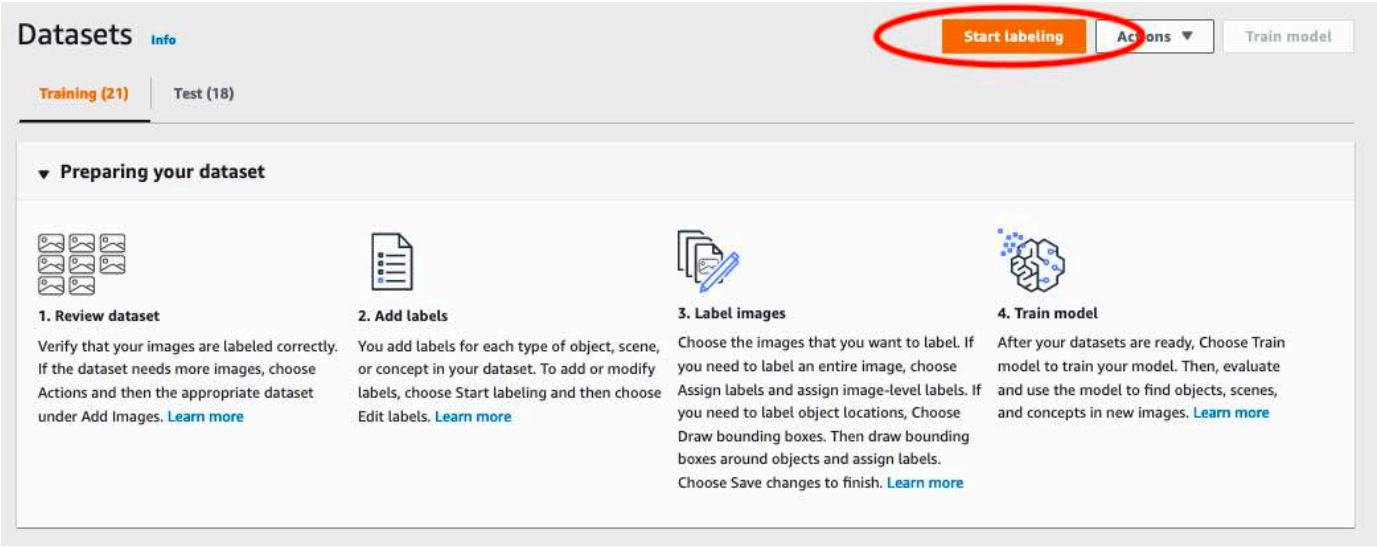
14. Wiederholen Sie die Schritte 8 bis 12, um dem Testdatensatz Bilder hinzuzufügen. Wählen Sie für Schritt 8Aktionenund dann wähleBilder zum Testdatensatz hinzufügen.

Schritt 5: Fügen Sie dem Projekt Labels hinzu

In diesem Schritt fügen Sie dem Projekt für jede der Klassen, die Sie in Schritt identifiziert haben, ein Label hinzu.Schritt 2: Wähle deine Klassen.

Um ein neues Label hinzuzufügen (Konsole)

1. Wählen Sie auf der Datensatz-Galerie-SeiteFangen Sie an zuum in den Kennzeichnungsmodus zu wechseln.



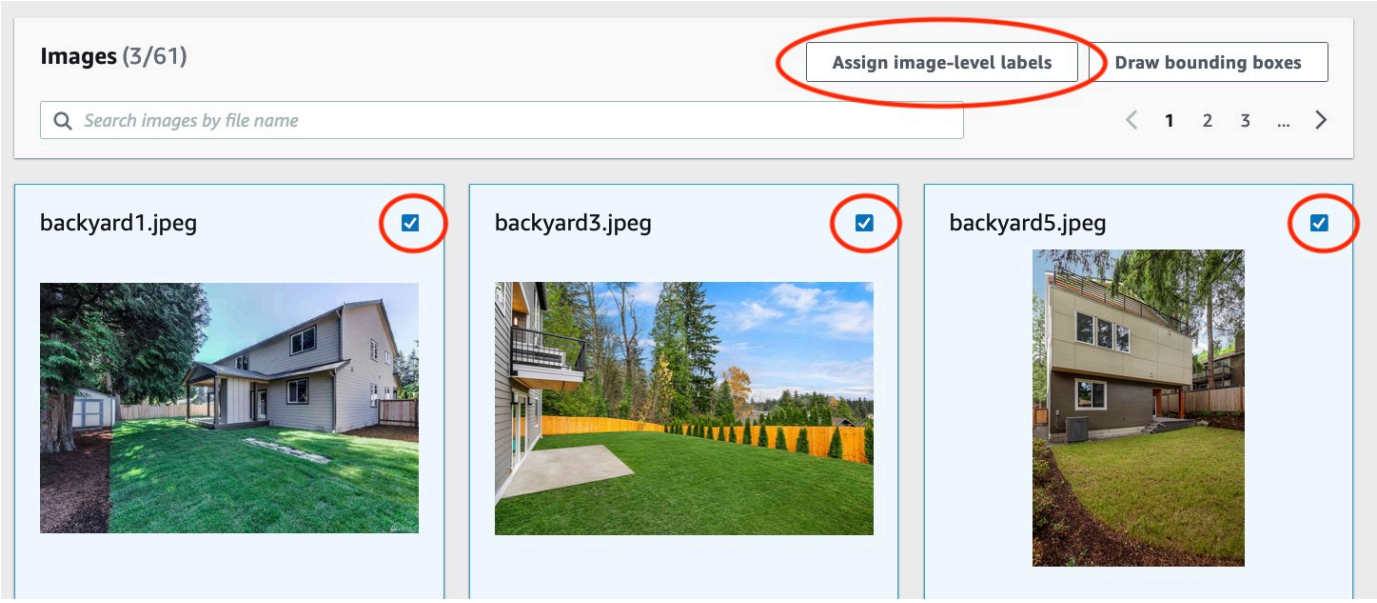
2. In derBeschriftungenAbschnitt der Datensatzgalerie, wählen SieLabels bearbeitenum das zu öffnenLabels verwaltenDialogfenster.
3. Geben Sie im Bearbeitungsfeld einen neuen Labelnamen ein.
4. Wählen SieBezeichnung hinzufügen.
5. Wiederholen Sie die Schritte 3 und 4, bis Sie alle benötigten Labels erstellt haben.
6. Wählen SieSpeichernum die von Ihnen hinzugefügten Labels zu speichern.

Schritt 6: Weisen Sie Trainings- und Testdatensätzen Labels auf Bildebene zu

In diesem Schritt weisen Sie jedem Bild in Ihren Trainings- und Testdatensätzen eine einzelne Bildebene zu. Das Label auf Bildebene ist die Klasse, die jedes Bild repräsentiert.

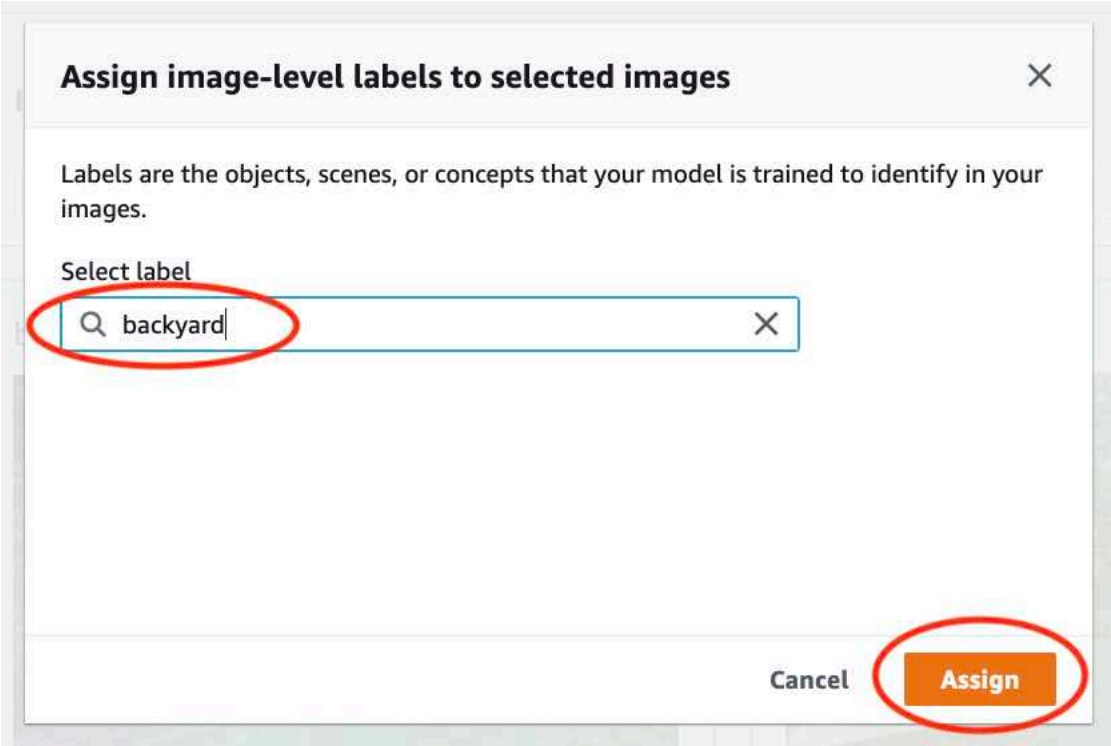
So weisen Sie einem Bild Labels auf Bildebene zu (Konsole)

1. Auf derDatensätzeSeite, wählen Sie dieSchulungTab.
2. Wählen SieFangen Sie an zuum in den Kennzeichnungsmodus zu wechseln.
3. Wählen Sie ein oder mehrere Bilder aus, denen Sie Beschriftungen hinzufügen möchten. Sie können jeweils nur Bilder auf einer einzelnen Seite auswählen. Um einen zusammenhängenden Bildbereich auf einer Seite auszuwählen:
   a. Wählen Sie das erste Bild aus.
   b. Halten Sie die Umschalttaste gedrückt.
   c. Wählen Sie das zweite Bild aus. Die Bilder zwischen dem ersten und zweiten Bild werden ebenfalls ausgewählt.
   d. Lassen Sie die Umschalttaste los.
4. Wählen SieLabels auf Bildebene zuweisen.



5. InWeisen Sie ausgewählten Bildern Labels auf Bildebene zuWählen Sie im Dialogfeld eine Bezeichnung aus, die Sie dem Bild oder den Bildern zuweisen möchten.

6. Wählen SieZuweisenum dem Bild ein Label zuzuweisen.



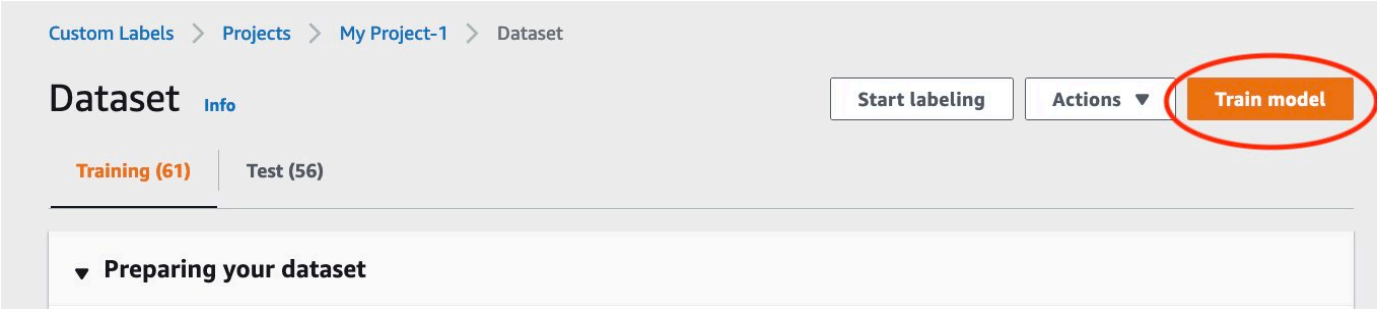
7. Wiederholen Sie die Beschriftung, bis jedes Bild mit den erforderlichen Beschriftungen versehen ist.
8. Wählen Sie die Registerkarte Test.
9. Wiederholen Sie die Schritte, um den Bildern des Testdatensatzes Labels auf Bildebene zuzuweisen.

Schritt 7: Trainiere dein Modell

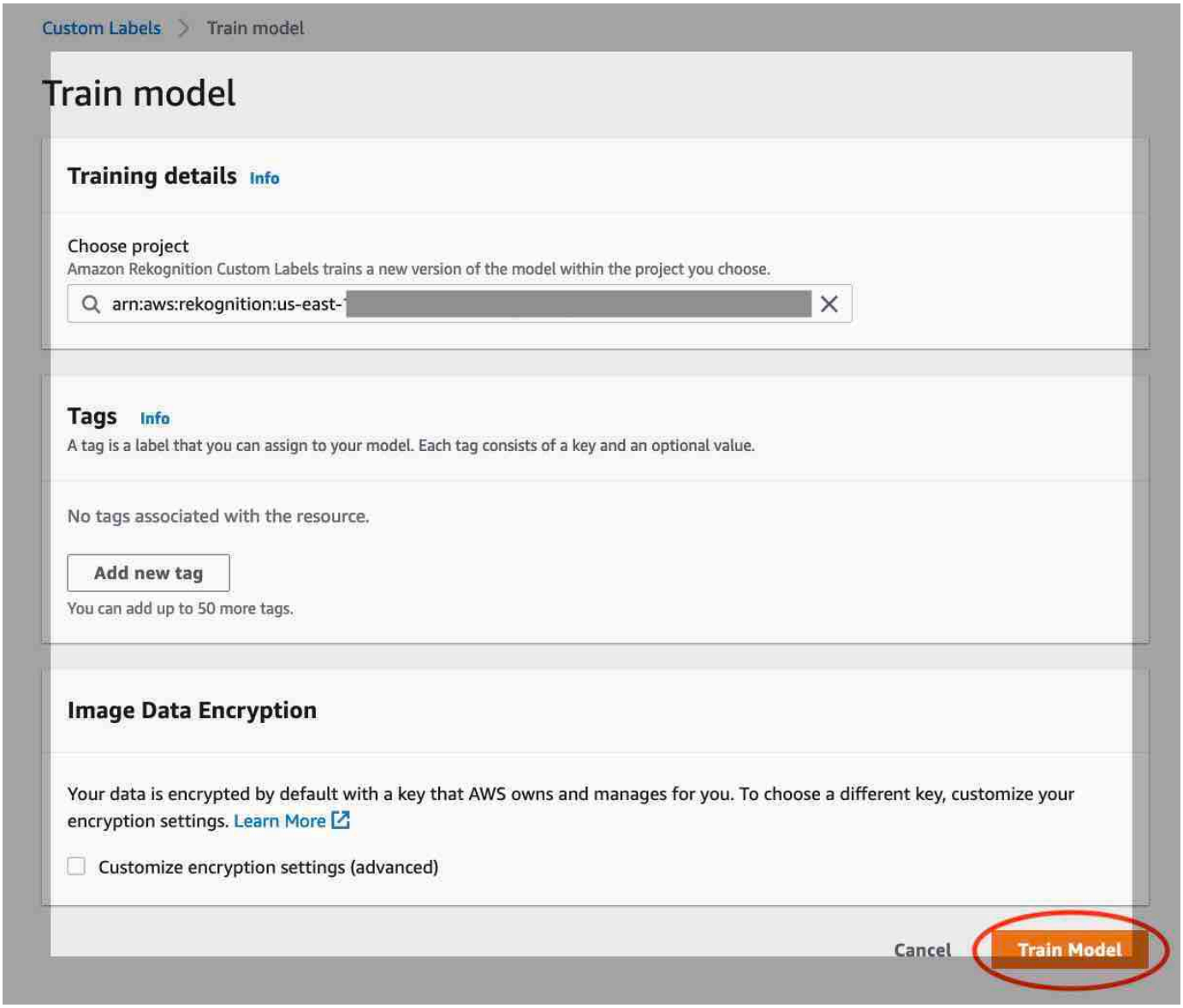
Gehen Sie wie folgt vor, um Ihr Modell zu trainieren. Weitere Informationen finden Sie unter Schulung eines Amazon-Rekognition-Custom-Labels-Modells.

Um dein Modell zu trainieren (Konsole)

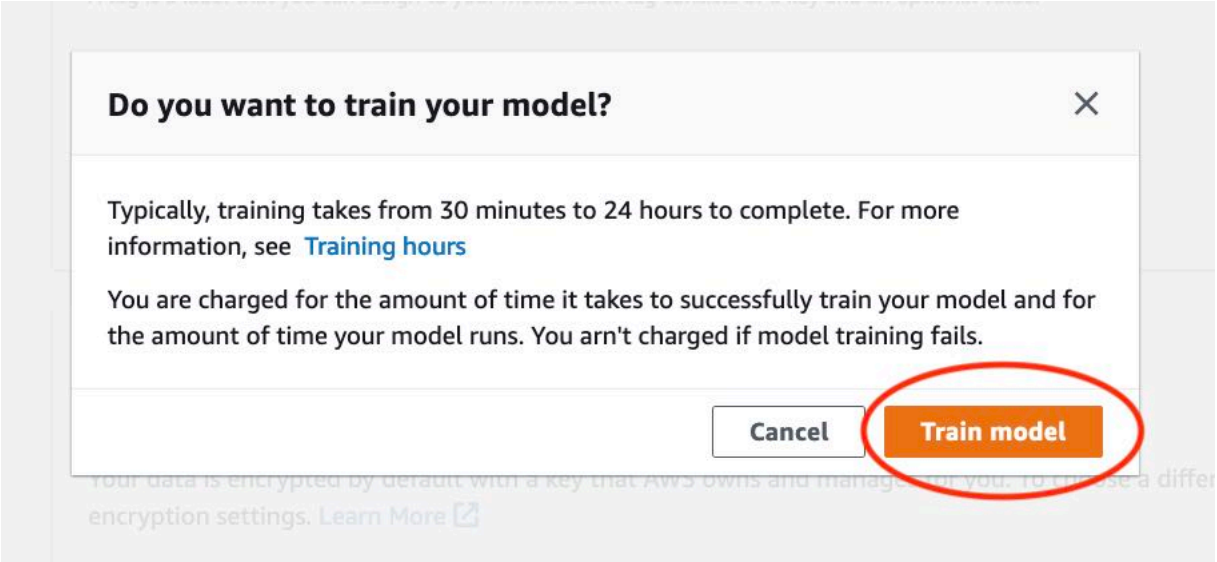
1. Auf derDatensatzSeite, wählenZugmodell.



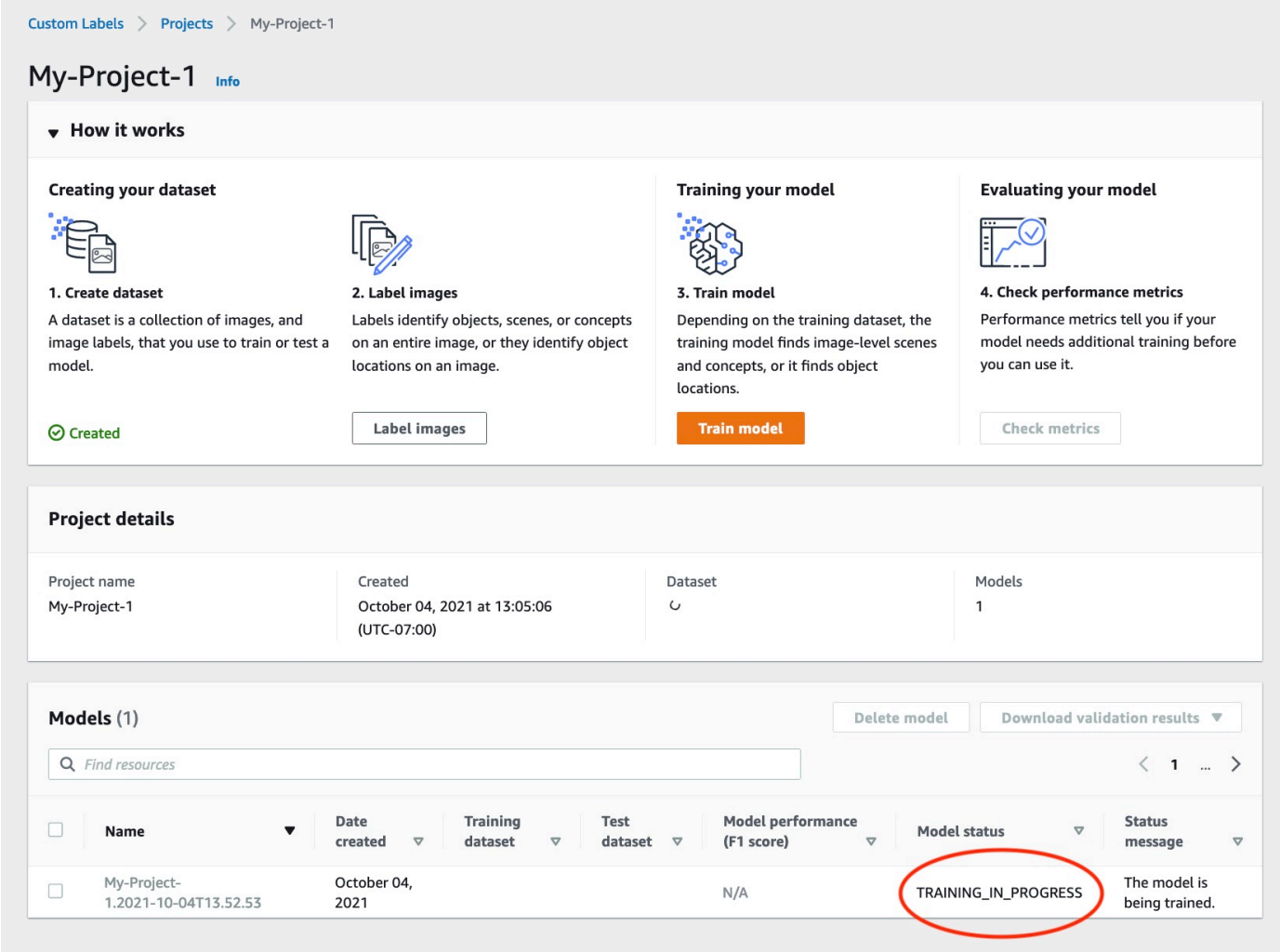
2. Auf derZugmodellSeite, wählenZugmodell. Der Amazon-Ressourcenname (ARN) für Ihr Projekt befindet sich imWählen Sie ein ProjektFeld bearbeiten.



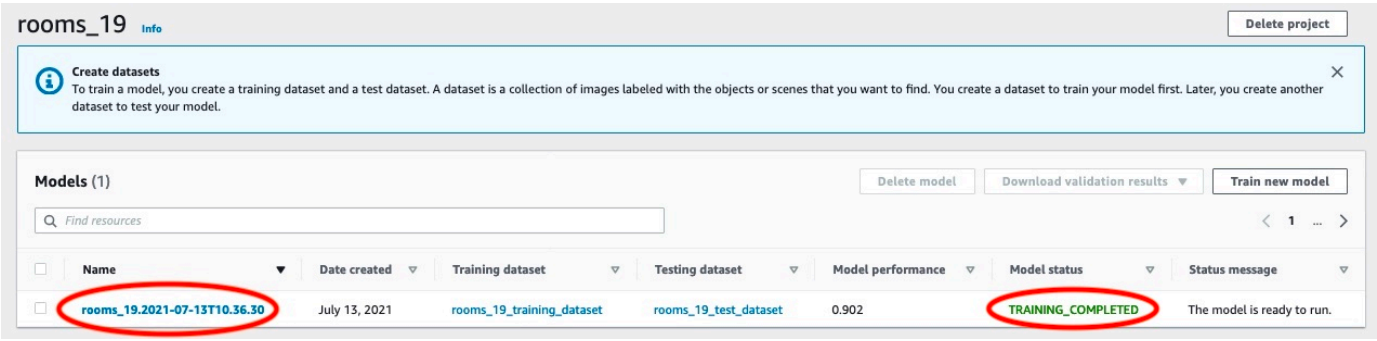
3. In derMöchtest du dein Modell trainieren?Dialogfenster, wählen SieZugmodell.



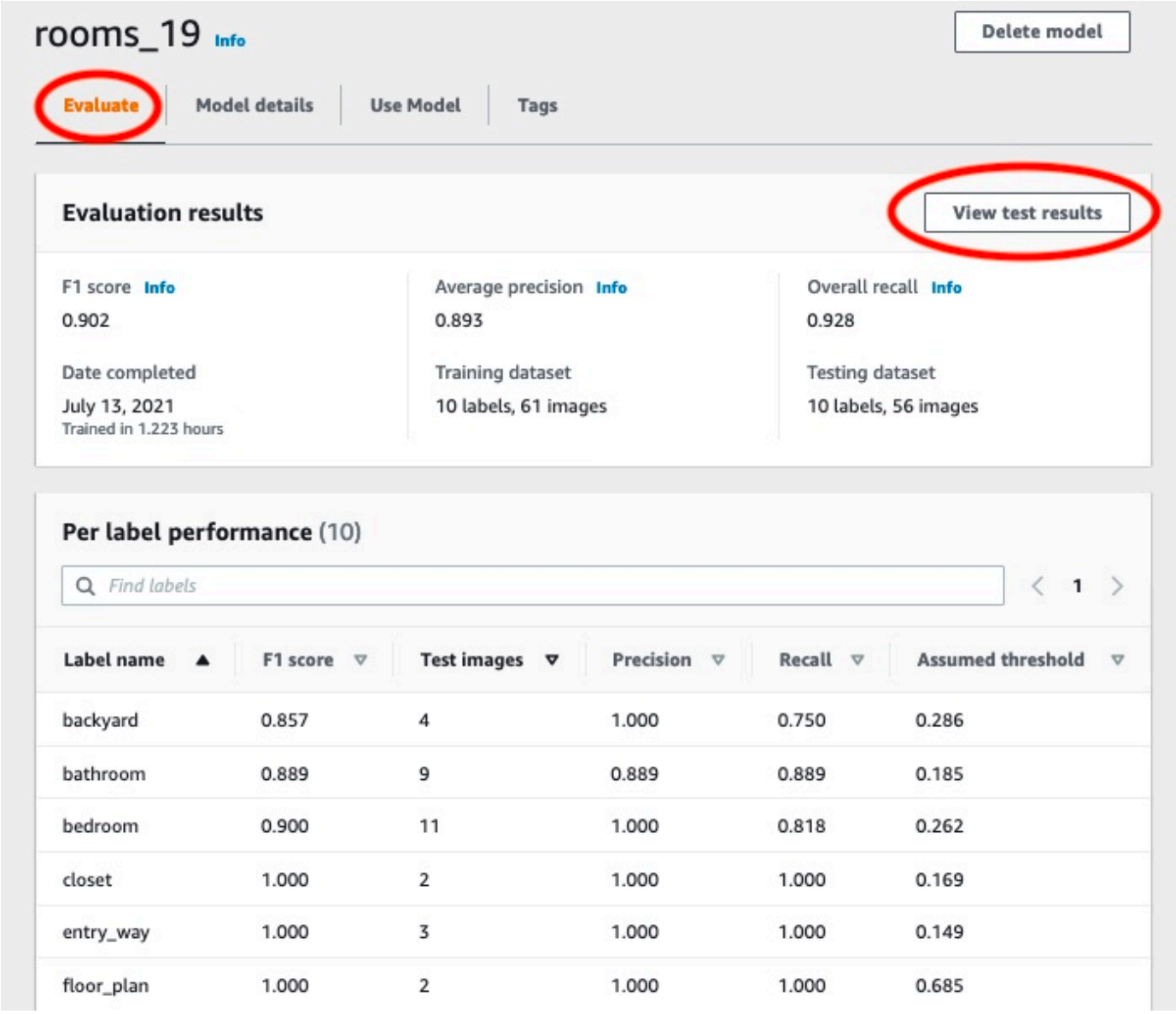
4. In derModelleIm Abschnitt der Projektseite können Sie sehen, dass die Schulung im Gange ist. Sie können den aktuellen Status überprüfen, indem Sie die`Model Status`Spalte für die Modellversion. Das Training eines Modells dauert eine Weile.



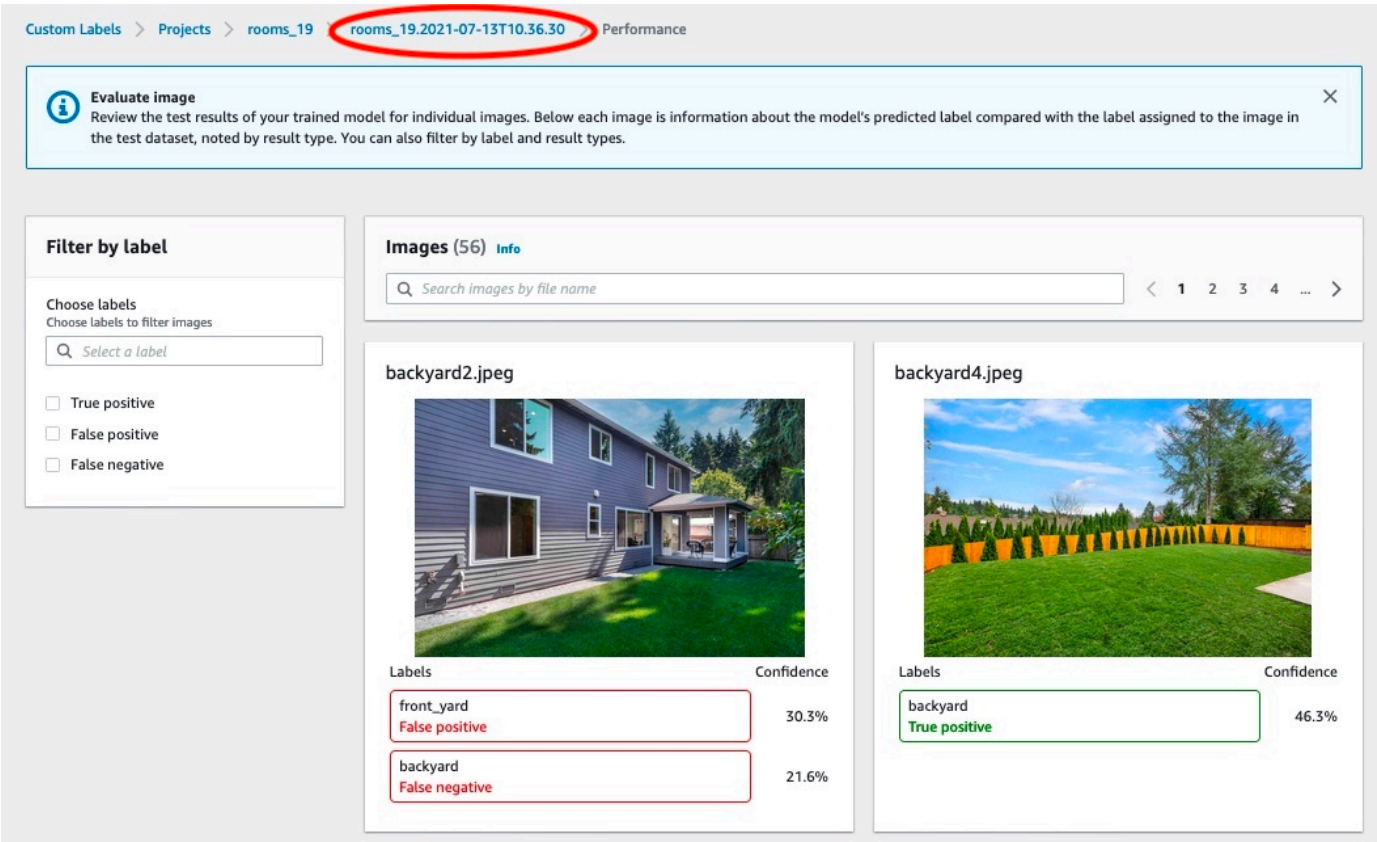
5. Wählen Sie nach Abschluss des Trainings den Modellnamen. Das Training ist beendet, wenn der Modellstatus lautetTRAINING_ABGESCHLOSSEN.



6. Wählen Sie dieEvaluierenSchaltfläche, um die Bewertungsergebnisse zu sehen. Hinweise zur Bewertung eines Modells finden Sie unterVerbessern eines geschulten Amazon Rekognition Custom Labels-Modells.
7. Wählen SieTestergebnisse ansehenum die Ergebnisse für einzelne Testbilder zu sehen. Weitere Informationen finden Sie unter Metriken für die Bewertung Ihres Modells.



8. Nachdem Sie sich die Testergebnisse angesehen haben, wählen Sie den Modellnamen aus, um zur Modellseite zurückzukehren.



Schritt 8: Starte dein Modell

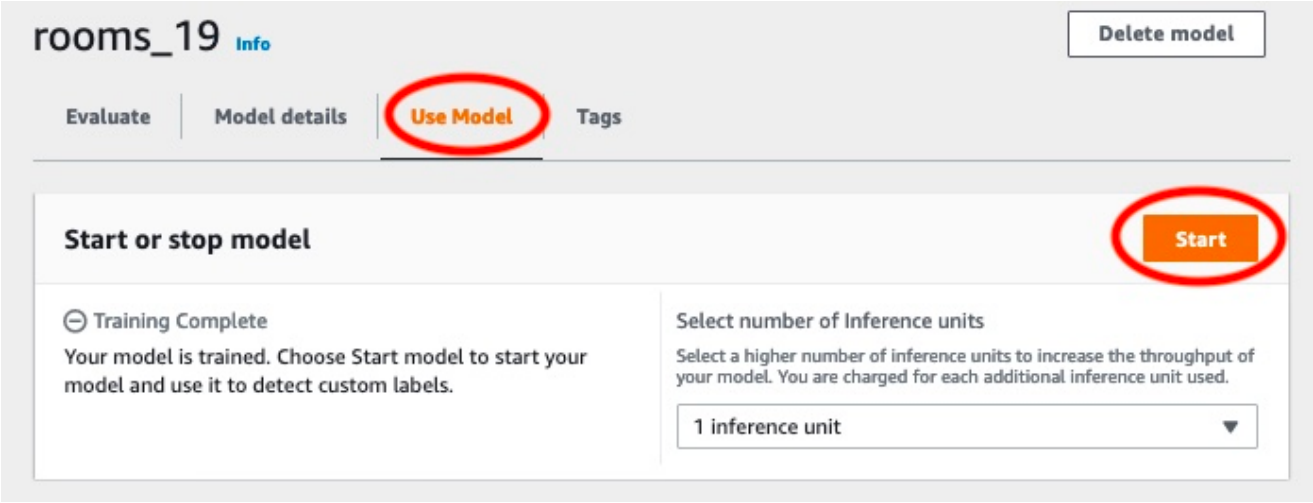
In diesem Schritt starten Sie Ihr Modell. Nachdem Ihr Modell gestartet ist, können Sie es verwenden, um Bilder zu analysieren.

Ihnen wird die Zeit in Rechnung gestellt, die Ihr Modell läuft. Stoppen Sie Ihr Modell, wenn Sie keine Bilder analysieren müssen. Sie können Ihr Modell zu einem späteren Zeitpunkt neu starten. Weitere Informationen finden Sie unter Ausführen eines trainierten Amazon Rekognition Custom Labels-Modells.

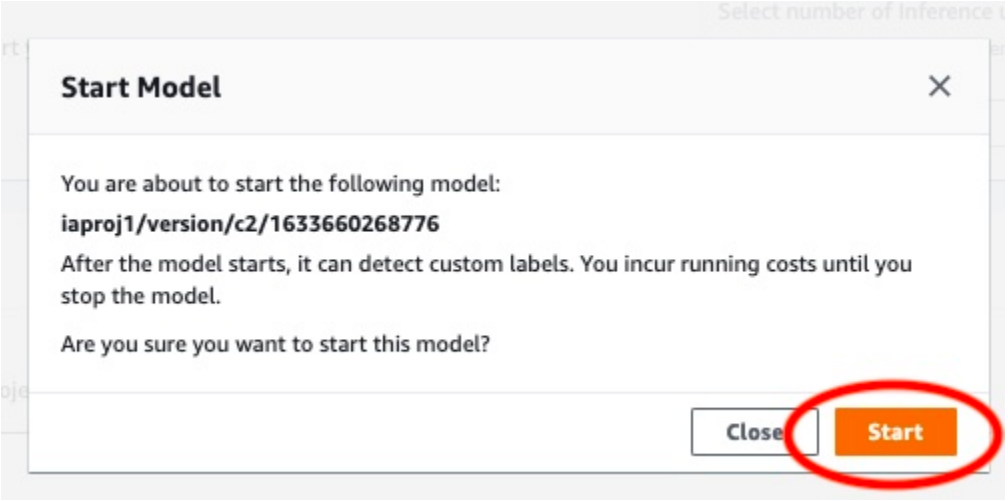
Um dein Modell zu starten

1. Wählen Sie dieModell verwendenReiter auf der Modellseite.
2. In derModell starten oder stoppengehen Sie im Abschnitt wie folgt vor:

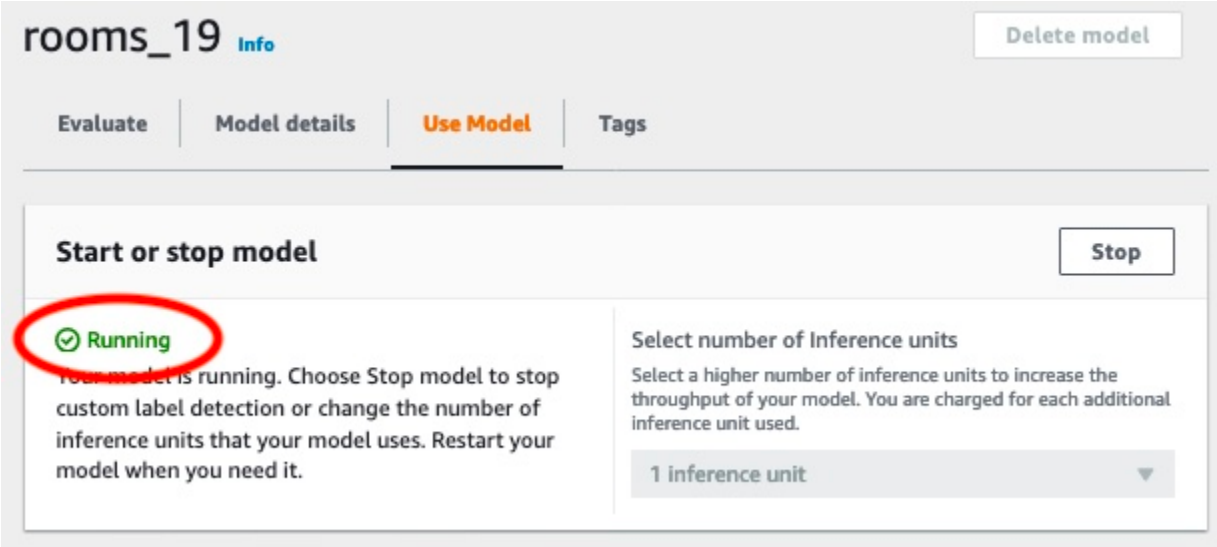
    a. Wählen Sie Starten.



b. In derModell startenDialogfenster, wählen SieBeginne.



3. Warten Sie, bis das Modell läuft. Das Modell läuft, wenn der Status in derModell starten oder stoppenAbschnitt istLaufen.



Schritt 9: Analysieren Sie ein Bild mit Ihrem Modell

Sie analysieren ein Bild, indem Sie denDetectCustomLabelsAPI. In diesem Schritt verwenden Sie den`detect-custom-labels` AWS Command Line Interface(AWS CLI) Befehl zur Analyse eines Beispielbilds. Du bekommst dieAWS CLIBefehl von der Amazon Rekognition Custom Labels-Konsole aus. Die Konsole konfiguriert dieAWS CLIBefehl, um Ihr Modell zu verwenden. Sie müssen nur ein Image bereitstellen, das in einem Amazon S3-Bucket gespeichert ist.

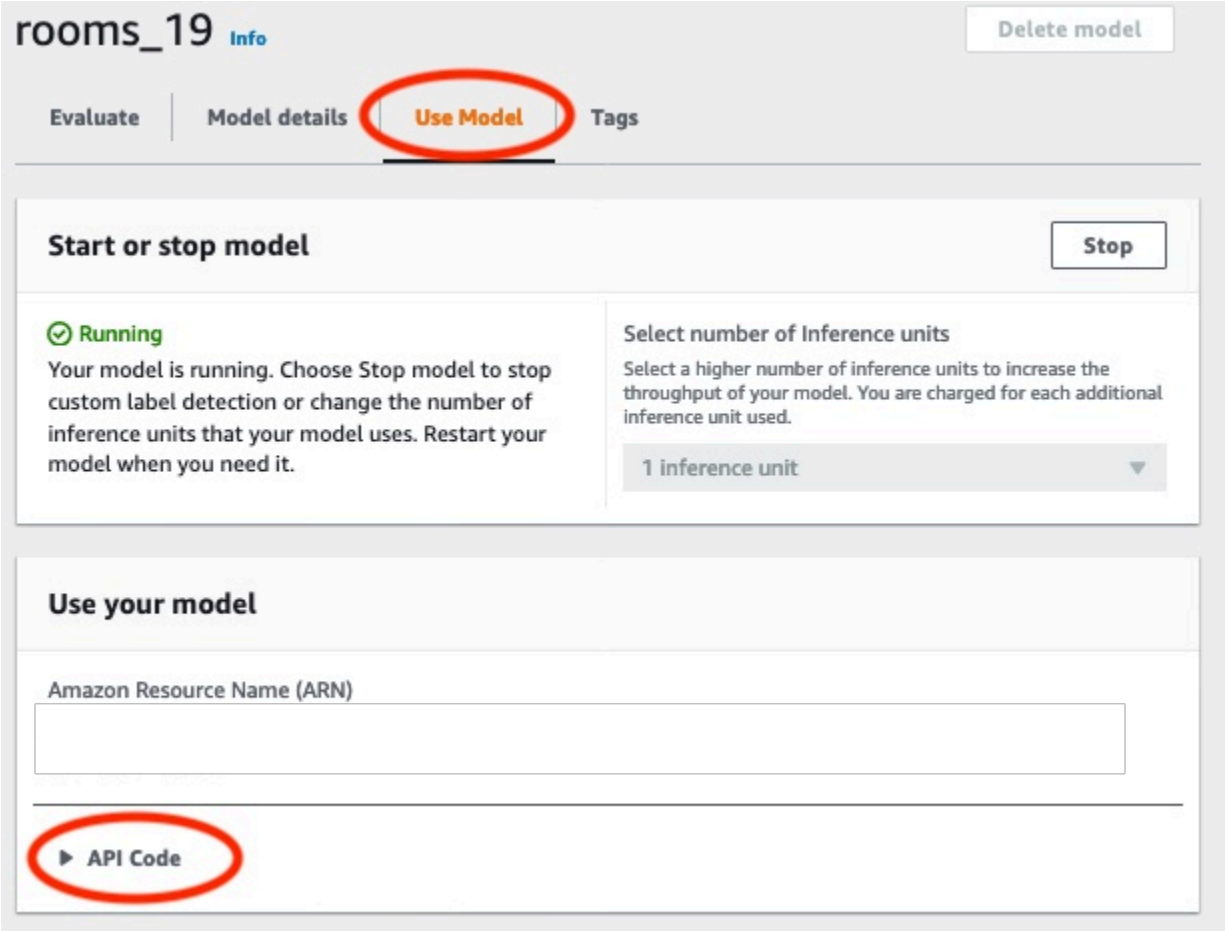
> **Note**
>
> Die Konsole bietet auch Python-Beispielcode.

Die Ausgabe von`detect-custom-labels`enthält eine Liste der im Bild gefundenen Beschriftungen, Begrenzungsrahmen (falls das Modell Objektpositionen findet) und das Vertrauen, das das Modell in die Genauigkeit der Vorhersagen hat.

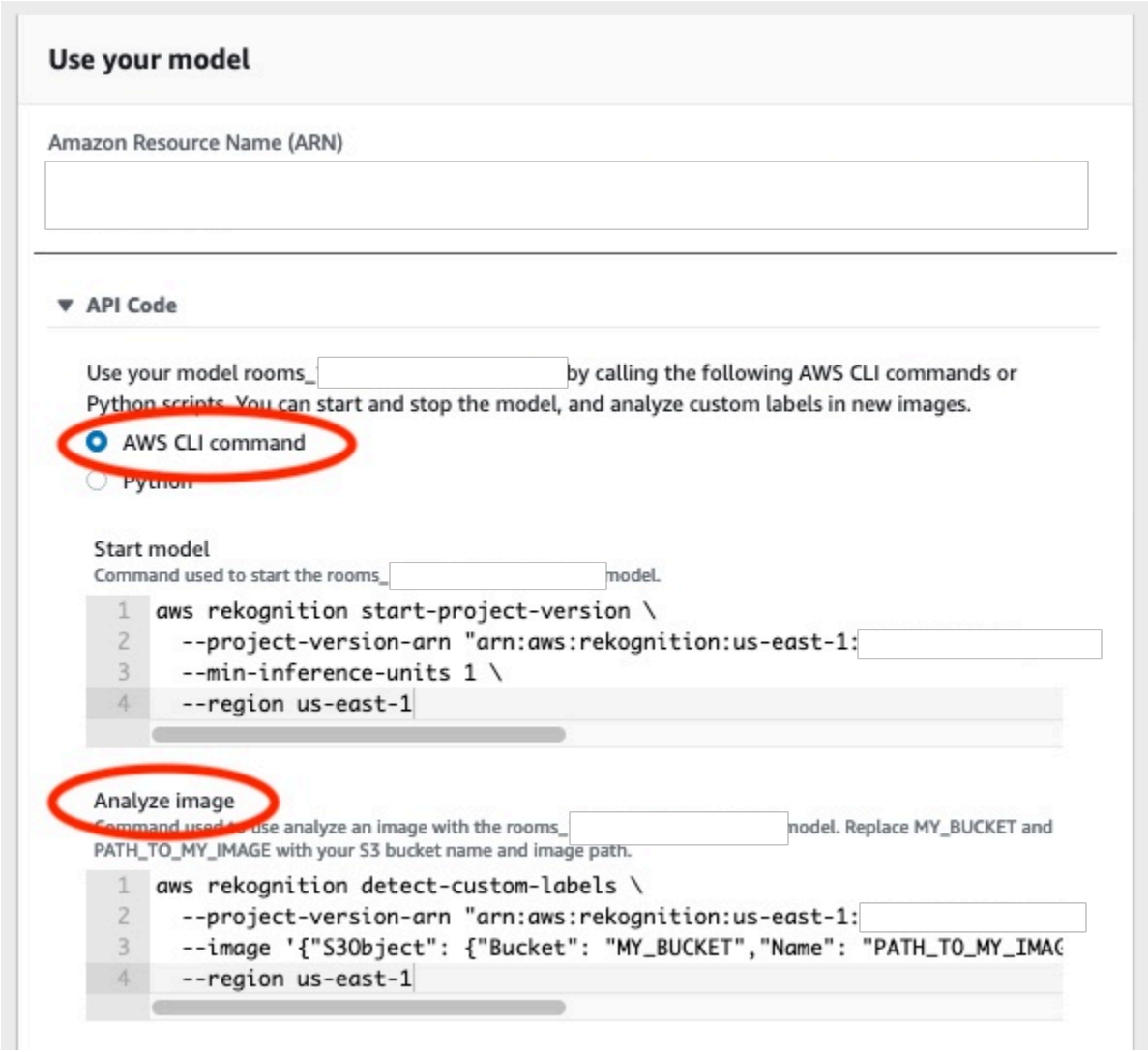
Weitere Informationen finden Sie unter Analysieren eines Bildes mit einem trainierten Modell.

Um ein Bild zu analysieren (Konsole)

1. Falls Sie dies noch nicht getan haben, richten Sie das einAWS CLI. Detaillierte Anweisungen finden Sie unter the section called “Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS”.
2. Wählen Sie dieModell verwendentippen Sie auf und wählen Sie dannAPI-Code.



3. Wählen SieAWS-CLI-Befehl.
4. In derBild analysierenAbschnitt, kopiere denAWS CLIBefehl, der aufruft`detect-custom-labels`.



5. Laden Sie ein Bild in einen Amazon S3-Bucket hoch. Anweisungen finden Sie unterObjekte in Amazon S3 hochladenin derAmazon Simple Storage Service-Benutzerhandbuch. Wenn Sie Bilder aus dem Rooms-Projekt verwenden, verwenden Sie eines der Bilder, die Sie in einen separaten Ordner verschoben habenSchritt 1: Sammle deine Bilder.

6. Geben Sie in der Befehlszeile denAWS CLIBefehl, den Sie im vorherigen Schritt kopiert haben. Es sollte wie das folgende Beispiel aussehen.

   Der Wert von`--project-version-arn`sollte der Amazon Resource Name (ARN) Ihres Modells sein. Der Wert von`--region`sollte der seinAWSRegion, in der Sie das Modell erstellt haben.

   Veränderung`MY_BUCKET`und`PATH_TO_MY_IMAGE`zu dem Amazon S3-Bucket und dem Image, das Sie im vorherigen Schritt verwendet haben.

Wenn Sie das verwendencustom-labels-accessProfil, um Anmeldeinformationen zu erhalten, fügen Sie das hinzu`--profile custom-labels-access`Parameter.

```
aws rekognition detect-custom-labels \
  --project-version-arn "model_arn" \
  --image '{"S3Object": {"Bucket": "MY_BUCKET","Name": "PATH_TO_MY_IMAGE"}}' \
  --region us-east-1 \
  --profile custom-labels-access
```

Die JSON-Ausgabe vonAWS CLIDer Befehl sollte wie folgt aussehen.`Name`ist der Name des Labels auf Bildebene, das das Modell gefunden hat.`Confidence`(0-100) ist das Vertrauen des Modells in die Genauigkeit der Vorhersage.

```
{
    "CustomLabels": [
        {
            "Name": "living_space",
            "Confidence": 83.41299819946289
        }
    ]
}
```

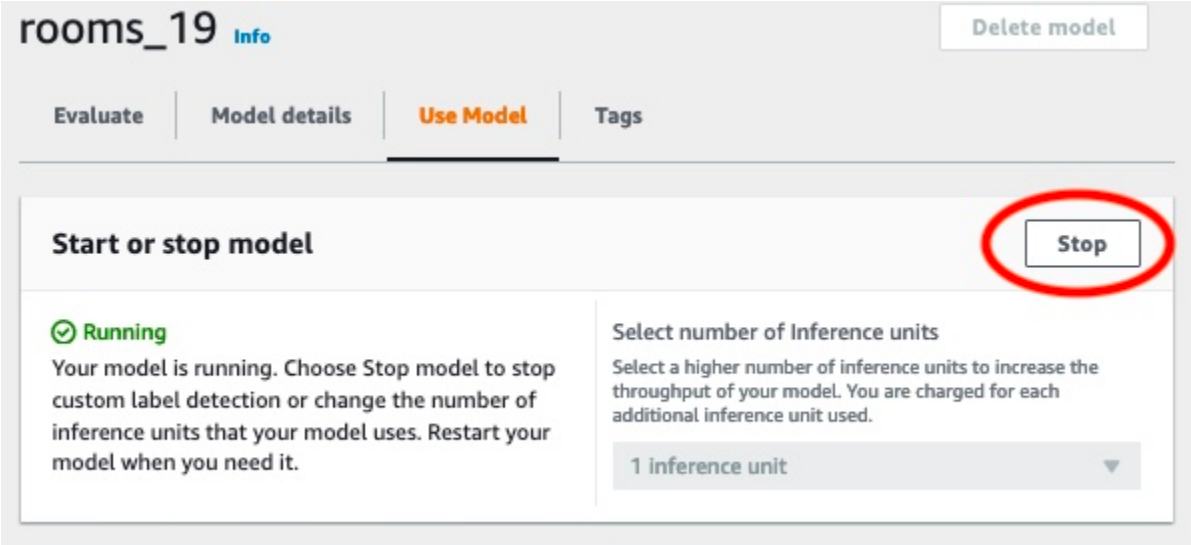
7. Verwenden Sie das Modell weiterhin, um andere Bilder zu analysieren. Stoppen Sie das Modell, wenn Sie es nicht mehr verwenden.

Schritt 10: Stoppen Sie Ihr Modell

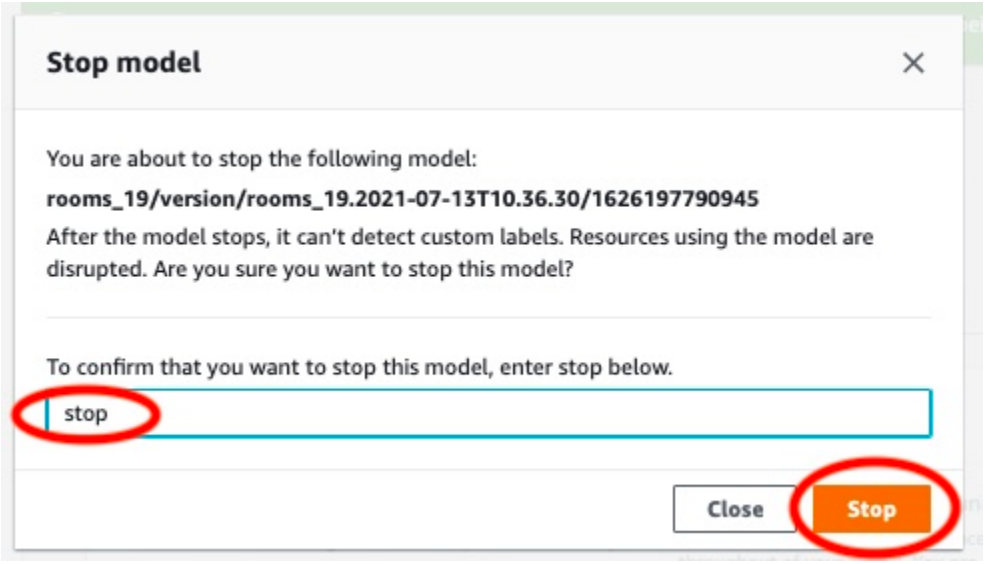
In diesem Schritt beenden Sie die Ausführung Ihres Modells. Ihnen wird die Zeit in Rechnung gestellt, in der Ihr Modell läuft. Wenn Sie das Modell nicht mehr verwenden, sollten Sie es beenden.

Um dein Modell zu stoppen

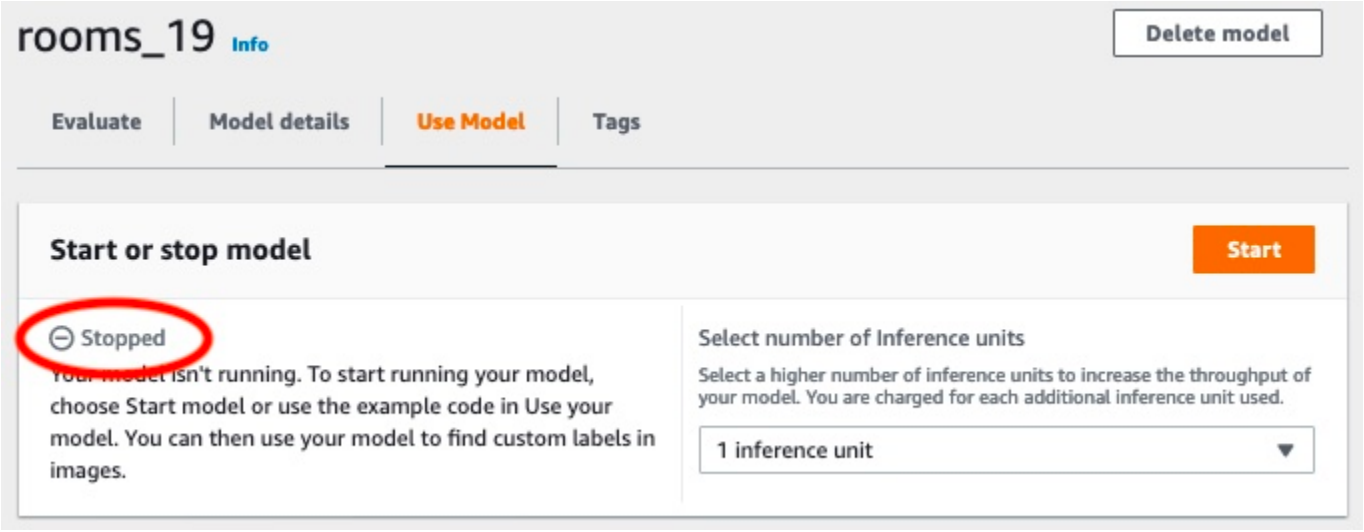
1. In derModell starten oder stoppenAbschnitt wählenStopp.



2. In derModell stoppenDialogfenster, geben Sie einhaltum zu bestätigen, dass Sie das Modell beenden möchten.



3. Wählen SieStoppum dein Modell zu stoppen. Das Modell wurde gestoppt, als der Status in derModell starten oder stoppenAbschnitt istGestopft.


rooms_19 Info
Delete model
Evaluate
Model details
Use Model
Tags
Start or stop model
Start
Stopped
isn't running. To start running your model, choose Start model or use the example code in Use your model. You can then use your model to find custom labels in images.
Select number of Inference units
Select a higher number of inference units to increase the throughput of your model. You are charged for each additional inference unit used.
1 inference unit

Erstellen eines Amazon Rekognition-Custom-Custom-Label-Erstellen eines neuen

Ein Modell ist die Software, die Sie trainieren, um die Konzepte, Szenen und Objekte zu finden, die für Ihr Unternehmen einzigartig sind. Sie können ein Modell mit der Amazon Rekognition Custom Labels-Konsole oder mit demAWS SDK erstellen. Bevor Sie ein Amazon Rekognition-Custom-Label-Modell Rekognition-Custom-Label-ModellGrundlegendes zu Amazon Rekognition Custom Labels Rekognition-Custom-Custom

Dieser Abschnitt enthält Konsolen- und SDK-Informationen zum Erstellen eines Projekts, zum Erstellen von Trainings- und Testdatensätzen für verschiedene Modelltypen und zum Trainieren eines Modells. In späteren Abschnitten erfahren Sie, wie Sie Ihr Modell verbessern und verwenden können. Ein Tutorial, das Ihnen veranschaulicht, wie Sie einen bestimmten Modell mit der Konsole erstellen und verwenden, finden Sie unterTutorial: Bilder klassifizieren.

Themen

- Erstellen eines eines Projekts
- Trainings- und Testdatensätze erstellen
- Schulung eines Amazon-Rekognition-Custom-Labels-Modells
- Debuggen eines fehlgeschlagenen Modelltrainings

Erstellen eines eines Projekts

Ein Projekt verwaltet die Modellversionen, den Trainingsdatensatz und den Testdatensatz für ein Modell. Sie können ein Projekt mit der Amazon Rekognition Custom Labels -Konsole oder mit der -API erstellen. Weitere Projektaufgaben, z. B. das Löschen eines Projekts, finden Sie unterVerwalten eines Amazon Rekognition Custom Labels-Projekts.

Erstellen eines Amazon-Rekognition-Custom-Labels-Projekts (-Konsole)

Sie können die Amazon Rekognition Custom Labels -Konsole zum Erstellen eines Projekts verwenden. Wenn Sie die Konsole zum ersten Mal in einer neuenAWS Region verwenden, fordert Amazon Rekognition Custom Labels Sie auf, in IhremAWS Konto einen Amazon S3 S3-Bucket (Konsolen-Bucket) zu erstellen. Der Bucket wird zum Speichern von Projektdateien verwendet. Sie

können die Amazon-Rekognition-Custom-Labels-Konsole nur verwenden, wenn der -Konsole erstellt wurde.

Sie können die Amazon Rekognition Custom Labels -Konsole zum Erstellen eines Projekts verwenden.

Um ein Projekt zu erstellen (Konsole)

1. Melden Sie sich bei der anAWS Management Console und öffnen Sie die Amazon Rekognition Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie im linken Bereich Use Custom Labels aus. Die Amazon Rekognition Custom Labels Landingpage wird angezeigt.
3. Wählen Sie auf der Landingpage von Amazon Rekognition Custom Labels die Option Erste Schritte aus.
4. Wählen Sie im linken Bereich die Option „Wählen Sie im linken Bereich aus“ aus.
5. Wählen Sie Create Project (Projekt anlegen) aus.
6. Geben Sie im Feld Project name (Projektname) einen Namen für Ihr Projekt an.
7. Wählen Sie Projekt erstellen, um Ihr Projekt zu erstellen.
8. Folgen Sie den Schritten unterTrainings- und Testdatensätze erstellen, um die Trainings- und Testdatensätze für Ihr Projekt zu erstellen.

Erstellen eines Amazon-Rekognition-Custom-Labels-Projekts (SDK)

Erstellen eines Amazon-Rekognition-Custom-Labels-Projekts, indem Sie aufrufen CreateProject. Die Antwort ist ein Amazon-Ressourcenname (ARN) zur Identifizierung des Projekts. Nachdem Sie ein Projekt erstellt haben, erstellen Sie Datensätze für das Training und Testen eines Modells. Weitere Informationen finden Sie unter Erstellen von Trainings- und Testdatensätzen mit Bildern.

Um ein Projekt zu erstellen (SDK)

1. Wenn Sie es noch nicht getan haben, installieren und konfigurieren Sie dieAWS CLI und dieAWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Code, um ein Projekt zu erstellen.

AWS CLI

Das folgende Beispiel erstellt ein Projekt und zeigt seinen ARN an.

Ändern des Projekts`project-name` in den Namen des Projekts, das Sie erstellen möchten.

```
aws rekognition create-project --project-name my_project \
 --profile custom-labels-access
```

Python

Das folgende Beispiel erstellt ein Projekt und zeigt seinen ARN an. Geben Sie die folgenden Befehlszeilenargumente an:

- `project_name`— der Name des Projekts, das Sie erstellen möchten.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_project(rek_client, project_name):
    """
    Creates an Amazon Rekognition Custom Labels project
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: A name for the new prooject.
    """

    try:
        #Create the project.
        logger.info("Creating project: %s",project_name)

        response=rek_client.create_project(ProjectName=project_name)

        logger.info("project ARN: %s",response['ProjectArn'])
```

```
        return response['ProjectArn']


    except ClientError as err:
        logger.exception("Couldn't create project - %s: %s", project_name,
 err.response['Error']['Message'])
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_name", help="A name for the new project."
    )


def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Creating project: {args.project_name}")

        # Create the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        project_arn=create_project(rekognition_client,
            args.project_name)

        print(f"Finished creating project: {args.project_name}")
        print(f"ARN: {project_arn}")

    except ClientError as err:
```

```
        logger.exception("Problem creating project: %s", err)
        print(f"Problem creating project: {err}")


if __name__ == "__main__":
    main()
```

Java V2

Das folgende Beispiel erstellt ein Projekt und zeigt seinen ARN an.

Geben Sie das folgende Befehlszeilenargument ein:

- `project_name`— der Name des Projekts, das Sie erstellen möchten.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateProjectRequest;
import software.amazon.awssdk.services.rekognition.model.CreateProjectResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateProject {

    public static final Logger logger =
 Logger.getLogger(CreateProject.class.getName());

    public static String createMyProject(RekognitionClient rekClient, String
 projectName) {

        try {
```

```
            logger.log(Level.INFO, "Creating project: {0}", projectName);
            CreateProjectRequest createProjectRequest =
 CreateProjectRequest.builder().projectName(projectName).build();

            CreateProjectResponse response =
 rekClient.createProject(createProjectRequest);

            logger.log(Level.INFO, "Project ARN: {0} ", response.projectArn());

            return response.projectArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create project: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        final String USAGE = "\n" + "Usage: " + "<project_name> <bucket> <image>
\n\n" + "Where:\n"
                + "   project_name - A name for the new project\n\n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectName = args[0];
        String projectArn = null;
        ;

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Create the project
```

```
            projectArn = createMyProject(rekClient, projectName);

            System.out.println(String.format("Created project: %s %nProject ARN:
 %s", projectName, projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

3. Notieren Sie sich den Namen des Projekt-ARN, der in der Antwort angezeigt wird. Sie benötigen sie zum Erstellen eines Modells.
4. Folgen Sie den Schritten unterErstellen von Trainings- und Testdatensätzen (SDK) , um die Trainings- und Testdatensätze für Ihr Projekt zu erstellen.

Trainings- und Testdatensätze erstellen

Ein Datensatz besteht aus einer Reihe von Bildern und Labels, die diese Bilder beschreiben. Ihr Projekt benötigt einen Trainingsdatensatz und einen Testdatensatz. Amazon Rekognition Custom Labels verwendet den Trainingsdatensatz, um Ihr Modell zu trainieren. Nach dem Training überprüft Amazon Rekognition Custom Labels anhand des Testdatensatzes, wie gut das trainierte Modell die richtigen Labels vorhersagt.

Sie können Datensätze mit der Amazon Rekognition Custom Labels-Konsole oder mit dem SDK erstellen. AWS Bevor Sie einen Datensatz erstellen, empfehlen wir Ihnen, Grundlegendes zu Amazon Rekognition Custom Labels zu lesen. Informationen zu anderen Datensatzaufgaben finden Sie unter Verwalten von Datensätzen.

Das Erstellen von Trainings- und Testdatensätzen für ein Projekt umfasst die folgenden Schritte:

So erstellen Sie Trainings- und Testdatensätze für Ihr Projekt

1. Ermitteln Sie, mit welchen Labels Sie Ihre Trainings- und Testdatensätze versehen müssen. Weitere Informationen finden Sie unter Datensätzen einen Zweck geben.
2. Sammeln Sie die Bilder für Ihre Trainings- und Testdatensätze. Weitere Informationen finden Sie unter the section called “Vorbereiten der Bilder”.
3. Erstellen Sie die Trainings- und Testdatensätze. Weitere Informationen finden Sie unter Erstellen von Trainings- und Testdatensätzen mit Bildern. Wenn Sie das AWS SDK verwenden, finden Sie weitere Informationen unter. Erstellen von Trainings- und Testdatensätzen (SDK)
4. Fügen Sie Ihren Datensatzbildern bei Bedarf Labels oder Begrenzungsrahmen auf Bildebene hinzu. Weitere Informationen finden Sie unter Labeling von Bildern.

Nachdem Sie die Datensätze erstellt haben, können Sie das Modell trainieren.

Themen

Datensätzen einen Zweck geben

Die Labels, mit denen Sie die Trainings- und Testdatensätze in Ihrem Projekt versehen, bestimmen den Typ des Modells, das Sie erstellen. Mit Amazon Rekognition Custom Labels können Sie Modelle erstellen, die Folgendes tun.

- Objekte, Szenen und Konzepte finden
- Nach Objektpositionen suchen
- Marken finden

Objekte, Szenen und Konzepte finden

Das Modell klassifiziert die Objekte, Szenen und Konzepte, die einem ganzen Bild zugeordnet sind.

Sie können zwei Arten von Klassifizierungsmodellen erstellen: die Bildklassifizierung und die Klassifizierung mit mehreren Labels. Für beide Typen von Klassifikationsmodellen findet das Modell ein oder mehrere passende Labels aus dem gesamten Satz von Labels, die für das Training verwendet wurden. Sowohl für die Trainings- als auch für die Testdatensätze sind mindestens zwei Labels erforderlich.

Bildklassifizierung

Das Modell klassifiziert Bilder als zu einem Satz vordefinierter Labels gehörend. Beispielsweise könnte ein Modell verwendet werden, das bestimmt, ob ein Bild einen Wohnraum enthält. Das folgende Bild könnte das Label Wohnraum auf Bildebene haben.

Fügen Sie für diesen Modelltyp jedem Bild des Trainings- und Testdatensatzes ein einzelnes Label auf Bildebene hinzu. Ein Beispielobjekt finden Sie unter Bildklassifizierung.

Klassifizierung mit mehreren Labels

Das Modell unterteilt Bilder in mehrere Kategorien, z. B. die Art der Blüte und ob sie Blätter hat oder nicht. Das folgende Bild könnte beispielsweise die Labels mediterranean_spurge und no_leaves auf Bildebene haben.

Weisen Sie für diesen Modelltyp den Bildern der Trainings- und Testdatensätze Labels auf Bildebene für jede Kategorie zu. Ein Beispielobjekt finden Sie unter Bildklassifizierung (mehrere Label).

Zuweisen von Labels auf Bildebene

Wenn Ihre Bilder in einem Amazon-S3-Bucket gespeichert sind, können Sie Ordnernamen verwenden, um automatisch Labels auf Bildebene hinzuzufügen. Weitere Informationen finden Sie unter Amazon-S3-Bucket. Sie können Bildern auch Labels auf Bildebene hinzufügen, nachdem Sie einen Datensatz erstellt haben. Weitere Informationen finden Sie unter the section called “Einem Bild Labels auf Bildebene zuweisen”. Sie können bei Bedarf neue Labels hinzufügen. Weitere Informationen finden Sie unter Labels verwalten.

Nach Objektpositionen suchen

Um ein Modell zu erstellen, das die Position von Objekten in Ihren Bildern vorhersagt, definieren Sie Begrenzungsrahmen und Labels für die Objektposition der Bilder in Ihren Trainings- und Testdatensätzen. Ein Begrenzungsrahmen ist ein Rahmen, der ein Objekt eng umgibt. Das folgende Bild zeigt beispielsweise Begrenzungsboxen rund um einen Amazon Echo und einen Amazon Echo Dot. Jedem Begrenzungsrahmen ist eine Bezeichnung zugewiesen (Amazon Echo oder Amazon Echo Dot).

Um die Positionen von Objekten zu finden, benötigen Ihre Datensätze mindestens ein Label. Beim Modelltraining wird automatisch ein weiteres Label erstellt, das den Bereich außerhalb der Begrenzungsrahmen auf einem Bild darstellt.

Begrenzungsrahmen zuweisen

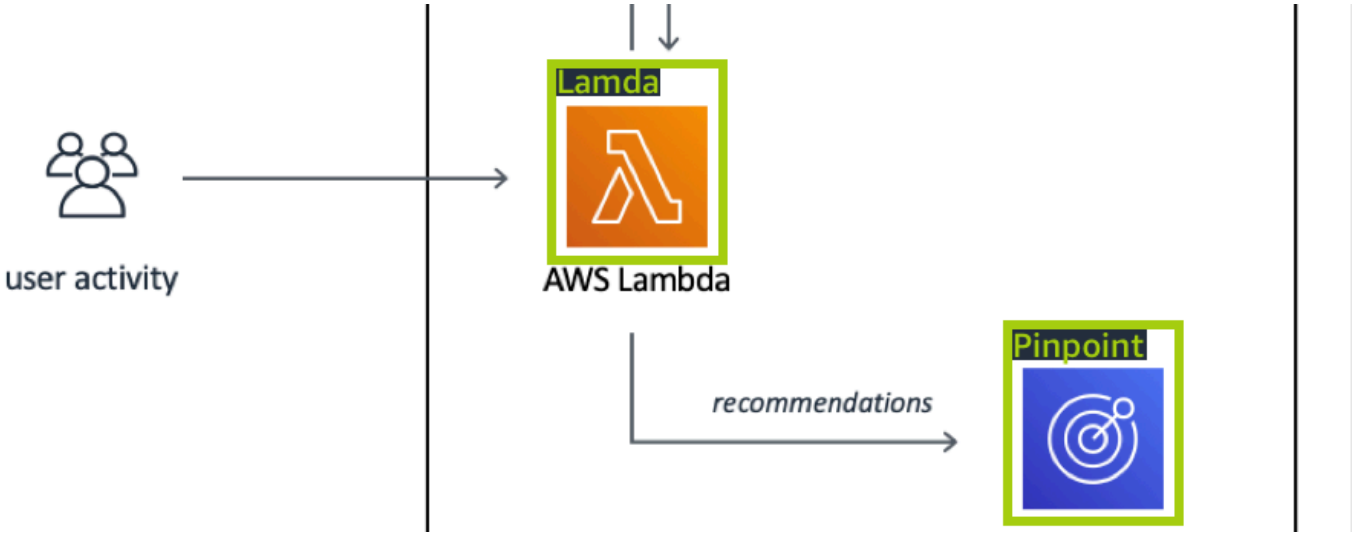
Wenn Sie Ihren Datensatz erstellen, können Sie Begrenzungsrahmeninformationen für Ihre Bilder hinzufügen. Sie können beispielsweise eine Manifestdatei SageMaker im Ground Truth Format importieren, die Begrenzungsrahmen enthält. Sie können Begrenzungsrahmen auch hinzufügen, nachdem Sie einen Datensatz erstellt haben. Weitere Informationen finden Sie unter Objekte mit Begrenzungsrahmen mit Labels versehen. Sie können bei Bedarf neue Labels hinzufügen. Weitere Informationen finden Sie unter Labels verwalten.

Marken finden

Wenn Sie Marken finden möchten, z. B. anhand ihrer Logos und animierter Charaktere, können Sie zwei verschiedene Bildtypen für Ihre Trainingsdatensätze verwenden.

- Nur Bilder, die das Logo zeigen. Jedes Bild benötigt ein einzelnes Label auf Bildebene, das den Namen des Logos darstellt. Die Bezeichnung auf Bildebene für das folgende Bild könnte beispielsweise Lambda lauten.

- Bilder, die das Logo an natürlichen Orten enthalten, z. B. bei einem Fußballspiel oder einem Architekturdiagramm. Jedes Trainingsbild benötigt Begrenzungsfelder, die jede Instance des Logos umgeben. Die folgende Abbildung zeigt beispielsweise ein Architekturdiagramm mit beschrifteten Begrenzungsrahmen, die die AWS Lambda- und Amazon Pinpoint Pinpoint-Logos umgeben.



Wir empfehlen, dass Sie in Ihren Trainingsbildern keine Labels auf Bildebene und Begrenzungsrahmen mischen.

Die Testbilder müssen mit Begrenzungsrahmen versehen sein, die Instances der Marke, nach der Sie suchen, umgeben. Sie können den Trainingsdatensatz nur dann aufteilen, um den Testdatensatz zu erstellen, wenn die Trainingsbilder beschriftete Begrenzungsrahmen enthalten. Wenn die Trainingsbilder nur Labels auf Bildebene haben, müssen Sie einen Testdatensatz erstellen, der Bilder mit Begrenzungsrahmen, die mit Labels versehen sind, enthält. Wenn Sie ein Modell darauf trainieren, Markenstandorte zu finden, führen Sie Objekte mit Begrenzungsrahmen mit Labels versehen und Einem Bild Labels auf Bildebene zuweisen aus, je nachdem, mit welchen Labels Sie Ihre Bilder versehen.

Das Erkennung von Marken-Beispielprojekt zeigt, wie Amazon Rekognition Custom Labels mit Labeln versehene Begrenzungsrahmen verwendet, um ein Modell zu trainieren, das Objektpositionen findet.

Label-Anforderungen für Modelltypen

Bestimmen Sie anhand der folgenden Tabelle, wie Sie Ihre Bilder mit Labels versehen.

Sie können Labels auf Bildebene und Bilder mit Begrenzungsrahmen in einem einzigen Datensatz kombinieren. In diesem Fall entscheidet Amazon Rekognition Custom Labels, ob ein Modell auf Bildebene oder ein Objektlokalisierungsmodell erstellt werden soll.

Beispiel	Trainingsbilder	Testbilder
Bildklassifizierung	1 Label auf Bildebene pro Bild	1 Label auf Bildebene pro Bild
Klassifizierung mit mehreren Labels	Mehrere Labels auf Bildebene pro Bild	Mehrere Labels auf Bildebene pro Bild
Marken finden	Labels auf Bildebene (Sie können auch mit Labeln versehene Begrenzun gsrahmen verwenden)	Mit Labeln versehene Begrenzungsrahmen
Nach Objektpositionen suchen	Mit Labeln versehene Begrenzungsrahmen	Mit Labeln versehene Begrenzungsrahmen

Vorbereiten der Bilder

Die Bilder in Ihrem Trainings- und Testdatensatz enthalten die Objekte, Szenen oder Konzepte, die Ihr Modell finden soll.

Der Inhalt der Bilder sollte eine Vielzahl von Hintergründen und Lichtverhältnissen haben, die den Bildern entsprechen, die das trainierte Modell identifizieren soll.

Dieser Abschnitt enthält Informationen zu den Bildern in Ihrem Trainings- und Testdatensatz.

Bildformat

Sie können Amazon Rekognition Custom Labels-Modelle mit Bildern im PNG- und JPEG-Format trainieren. Ebenso benötigen Sie Bilder im PNG- und JPEG-Format, um benutzerdefinierte Labels mit `DetectCustomLabels` zu erkennen.

Empfehlungen für Eingabebilder

Amazon Rekognition Custom Labels benötigt Bilder, um Ihr Modell zu trainieren und zu testen. Beachten Sie bei der Vorbereitung Ihrer Bilder Folgendes:

- Wählen Sie eine bestimmte Domain für das Modell, das Sie erstellen möchten. Sie könnten beispielsweise ein Modell für malerische Landschaften und ein anderes Modell für Objekte wie Maschinenteile auswählen. Amazon Rekognition Custom Labels funktioniert am besten, wenn sich Ihre Bilder in der ausgewählten Domain befinden.
- Verwenden Sie mindestens 10 Bilder, um Ihr Modell zu trainieren.
- Bilder müssen im PNG- oder JPEG-Format vorliegen.
- Verwenden Sie Bilder, die das Objekt in einer Vielzahl von Lichtverhältnissen, Hintergründen und Auflösungen zeigen.
- Die Trainings- und Testbilder sollten den Bildern ähneln, mit denen Sie das Modell verwenden möchten.
- Legen Sie fest, welche Labels Sie den Bildern zuweisen.
- Stellen Sie sicher, dass die Bilder in Bezug auf die Auflösung ausreichend groß sind. Weitere Informationen finden Sie unter Richtlinien und Kontingente in Amazon Rekognition Custom Labels.
- Stellen Sie sicher, dass Okklusionen keine Objekte verdecken, die Sie erkennen möchten.
- Es sollte ein ausreichender Kontrast zum Hintergrund bestehen.
- Die Aufnahmen sollten hell und scharf genug sein. Vermeiden Sie so weit wie möglich Bilder, die aufgrund von Motiv- und Kamerabewegungen verschwommen sein könnten.
- Verwenden Sie ein Bild, bei dem das Objekt einen großen Teil des Bildes einnimmt.
- Bilder in Ihrem Testdatensatz sollten keine Bilder sein, die sich im Trainingsdatensatz befinden. Sie sollten die Objekte, Szenen und Konzepte enthalten, für deren Analyse das Modell trainiert wurde.

Größe des Bildsatzes

Amazon Rekognition Custom Labels verwendet einen Bildsatz, um ein Modell zu trainieren. Sie sollten mindestens 10 Bilder für das Training verwenden. Amazon Rekognition Custom Labels speichert Trainings- und Testbilder in Datensätzen. Weitere Informationen finden Sie unter Erstellen von Trainings- und Testdatensätzen mit Bildern.

Erstellen von Trainings- und Testdatensätzen mit Bildern

Sie können mit einem Projekt beginnen, das einen einzelnen Datensatz enthält, oder mit einem Projekt, das separate Trainings- und Testdatensätze enthält. Wenn Sie mit einem einzelnen Datensatz beginnen, teilt Amazon Rekognition Custom Labels Ihren Datensatz während des Trainings auf, um einen Trainingsdatensatz (80 %) und einen Testdatensatz (20 %) für Ihr Projekt zu erstellen. Beginnen Sie mit einem einzigen Datensatz, wenn Amazon Rekognition Custom Labels entscheiden soll, wo Bilder zum Trainieren und Testen verwendet werden. Um die vollständige Kontrolle über Trainings, Tests und Leistungsoptimierungen zu haben, empfehlen wir, dass Sie Ihr Projekt mit separaten Trainings- und Testdatensätzen beginnen.

Sie können Trainings- und Testdatensätze für ein Projekt erstellen, indem Sie Bilder von einem der folgenden Speicherorte importieren:

- Amazon-S3-Bucket
- Lokaler Computer
- Manifestdatei
- Bestehender Datensatz

Wenn Sie Ihr Projekt mit separaten Trainings- und Testdatensätzen beginnen, können Sie für jeden Datensatz unterschiedliche Quellverzeichnisse verwenden.

Je nachdem, von wo Sie Ihre Bilder importieren, haben Ihre Bilder möglicherweise keine Labels. Beispielsweise haben Bilder, die von einem lokalen Computer importiert wurden, keine Label. Bilder, die aus einer Amazon SageMaker Ground Truth Manifest-Datei importiert wurden, sind beschriftet. Sie können die Amazon Rekognition Custom Labels-Konsole verwenden, um Labels hinzuzufügen, zu ändern und zuzuweisen. Weitere Informationen finden Sie unter Labeling von Bildern.

Wenn Bilder fehlerhaft hochgeladen werden, Bilder fehlen oder Labels in Bildern fehlen, lesen Sie Debuggen eines fehlgeschlagenen Modelltrainings.

Weitere Informationen zu Datensätzen finden Sie unter Verwalten von Datensätzen.

Erstellen von Trainings- und Testdatensätzen (SDK)

Sie können das AWS SDK verwenden, um Trainings- und Testdatensätze zu erstellen.

Trainingsdatensatz

Sie können das AWS SDK auf folgende Weise verwenden, um einen Trainingsdatensatz zu erstellen.

- Verwenden Sie es CreateDatasetmit einer von Ihnen bereitgestellten Manifestdatei im Amazon Sagemaker-Format. Weitere Informationen finden Sie unter the section called “Erstellen einer Manifestdatei”. Beispielcode finden Sie unter Einen Datensatz mit einer SageMaker Ground Truth Manifest-Datei (SDK) erstellen.
- Verwenden Sie `CreateDataset`, um einen bestehenden Amazon Rekognition Custom Labels-Datensatz zu kopieren. Beispielcode finden Sie unter Erstellen eines Datensatzes unter Verwendung eines vorhandenen Datensatzes (SDK).
- Erstellen Sie einen leeren Datensatz mit `CreateDataset` und fügen Sie zu einem späteren Zeitpunkt Datensatzeinträge mit hinzu. UpdateDatasetEntries Informationen zum Erstellen eines leeren Datensatzes finden Sie unter Hinzufügen eines Datensatzes zu einem Projekt. Informationen zum Hinzufügen von Bildern zu einem Datensatz finden Sie unter Weitere Bilder hinzufügen (SDK). Sie müssen die Datensatzeinträge hinzufügen, bevor Sie ein Modell trainieren können.

Testdatensatz

Sie können das AWS SDK auf folgende Weise verwenden, um einen Testdatensatz zu erstellen:

- Verwenden Sie es CreateDatasetmit einer von Ihnen bereitgestellten Manifestdatei im Amazon Sagemaker-Format. Weitere Informationen finden Sie unter the section called “Erstellen einer Manifestdatei”. Beispielcode finden Sie unter Einen Datensatz mit einer SageMaker Ground Truth Manifest-Datei (SDK) erstellen.
- Verwenden Sie `CreateDataset`, um einen bestehenden Amazon Rekognition Custom Labels-Datensatz zu kopieren. Beispielcode finden Sie unter Erstellen eines Datensatzes unter Verwendung eines vorhandenen Datensatzes (SDK).
- Erstellen Sie einen leeren Datensatz mit `CreateDataset` und fügen Sie zu einem späteren Zeitpunkt Datensatzeinträge mit `UpdateDatasetEntries` hinzu. Informationen zum Erstellen eines leeren Datensatzes finden Sie unter Hinzufügen eines Datensatzes zu einem Projekt. Informationen zum Hinzufügen von Bildern zu einem Datensatz finden Sie unter Weitere Bilder

hinzufügen (SDK). Sie müssen die Datensatzeinträge hinzufügen, bevor Sie ein Modell trainieren können.

- Teilen Sie den Trainingsdatensatz in separate Trainings- und Testdatensätze auf. Erstellen Sie zunächst einen leeren Testdatensatz mit `CreateDataset`. Verschieben Sie dann 20% der Trainingsdatensatzeinträge in den Testdatensatz, indem Sie aufrufen. DistributeDatasetEntries Informationen zum Erstellen eines leeren Datensatzes finden Sie unter Hinzufügen eines Datensatzes zu einem Projekt (SDK). Informationen zum Aufteilen des Trainingsdatensatzes finden Sie unter Verteilen eines Trainingsdatensatzes (SDK).

Amazon-S3-Bucket

Die Bilder werden von einem Amazon-S3-Bucket importiert. Sie können den Konsolen-Bucket oder einen anderen Amazon S3 S3-Bucket in Ihrem AWS Konto verwenden. Wenn Sie den Konsolen-Bucket verwenden, sind die erforderlichen Berechtigungen bereits eingerichtet. Wenn Sie den Konsolen-Bucket nicht verwenden, siehe Zugreifen auf externe Amazon-S3-Buckets.

> **Note**
>
> Sie können das AWS SDK nicht verwenden, um einen Datensatz direkt aus Bildern in einem Amazon S3 S3-Bucket zu erstellen. Erstellen Sie stattdessen eine Manifestdatei, die auf die Quellverzeichnisse der Bilder verweist. Weitere Informationen finden Sie unter Manifestdatei.

Während der Datensatzerstellung können Sie festlegen, dass Bildern Labelnamen zugewiesen werden, die auf dem Namen des Ordners basieren, der die Bilder enthält. Der/Die Ordner müssen ein untergeordnetes Element des Amazon S3-Ordnerpfads sein, den Sie bei der Datensatzerstellung im Ordner S3 angegeben haben. Informationen zum Erstellen eines Datensatzes finden Sie unter Erstellen eines Datensatzes durch Importieren von Bildern aus einem S3-Bucket.

Angenommen, ein Amazon-S3-Bucket hat die folgende Ordnerstruktur. Wenn Sie den Speicherort des Amazon S3-Ordners als S3-Bucket/Alexa-Geräte angeben, wird den Bildern im Ordner Echo das Label Echo zugewiesen. In ähnlicher Weise wird Bildern im Ordner Echo-Dot das Label Echo-Dot zugewiesen. Die Namen untergeordneter Ordner werden nicht für Bilder-Labels verwendet. Stattdessen wird der entsprechende untergeordnete Ordner des Amazon S3-Ordnerspeicherorts verwendet. Bildern im Ordner white-echo-dotswird beispielsweise das Label echo-dot zugewiesen. Bildern auf der Ebene des Speicherorts des S3-Ordners (Alexa-Geräte) sind keine Labels zugewiesen.

Ordner, die sich weiter unten in der Ordnerstruktur befinden, können für Bilder-Label verwendet werden, indem ein untergeordneter Speicherort für den S3-Ordner angegeben wird. Wenn Sie beispielsweise S3-Bucket/Alexa-Devices/Echo-Dot angeben, werden die Bilder im Ordner beschriftet. white-echo-dotwhite-echo-dot Bilder, die sich außerhalb des angegebenen Speicherorts des S3-Ordners befinden, wie z. B. Echo, werden nicht importiert.

```
S3-bucket
### alexa-devices
    ### echo
    #   ### echo-image-1.png
    #   ### echo-image-2.png
    #   ### .
    #   ### .
    ### echo-dot
        ### white-echo-dot
        #   ### white-echo-dot-image-1.png
        #   ### white-echo-dot-image-2.png
        #
        ### echo-dot-image-1.png
        ### echo-dot-image-2.png
        ### .
        ### .
```

Wir empfehlen Ihnen, den Amazon S3 S3-Bucket (Konsolen-Bucket) zu verwenden, der von Amazon Rekognition für Sie erstellt wurde, als Sie die Konsole zum ersten Mal in der aktuellen AWS Region geöffnet haben. Wenn sich der Amazon-S3-Bucket, den Sie verwenden, vom Konsolen-Bucket unterscheidet (extern), werden Sie von der Konsole während der Datensatzerstellung aufgefordert, die entsprechenden Berechtigungen einzurichten. Weitere Informationen finden Sie unter the section called “Schritt 2: Einrichten der Konsolen-Berechtigungen”.

Erstellen eines Datensatzes durch Importieren von Bildern aus einem S3-Bucket

Im folgenden Verfahren wird gezeigt, wie Sie einen Datensatz mithilfe von Bildern erstellen, die im S3-Bucket der Konsole gespeichert sind. Die Bilder erhalten als Label automatisch den Namen des Ordners, in dem sie gespeichert sind.

Nachdem Sie Ihre Bilder importiert haben, können Sie auf der Galerie-Seite eines Datensatzes weitere Bilder hinzufügen, Labels zuweisen und Begrenzungsrahmen hinzufügen. Weitere Informationen finden Sie unter Labeling von Bildern.

Laden Sie Ihre Bilder in einen Amazon Simple Storage Service-Bucket hoch.

1. Erstellen Sie auf Ihrem lokalen Computer einen Ordner. Verwenden Sie einen Ordnernamen wie Alexa-Geräte.
2. Erstellen Sie in dem Ordner, den Sie gerade erstellt haben, Ordner, die nach jedem Label benannt sind, das Sie verwenden möchten. Zum Beispiel Echo und Echo-Dot. Die Ordnerstruktur sollte in etwa folgendermaßen aussehen.

```
alexa-devices
### echo
#   ### echo-image-1.png
#   ### echo-image-2.png
#   ### .
#   ### .
### echo-dot
    ### echo-dot-image-1.png
    ### echo-dot-image-2.png
    ### .
    ### .
```

3. Platzieren Sie die Bilder, die einem Label entsprechen, in den Ordner mit demselben Labelnamen.
4. Melden Sie sich bei der Amazon S3 S3-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/s3/.
5. Fügen Sie den Ordner, den Sie in Schritt 1 erstellt haben, dem Amazon-S3-Bucket (Konsolen-Bucket) hinzu, den Amazon Rekognition Custom Labels bei der ersten Einrichtung für Sie erstellt hat. Weitere Informationen finden Sie unter Verwalten eines Amazon Rekognition Custom Labels-Projekts.
6. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
7. Wählen Sie Benutzerdefinierte Labels verwenden.
8. Wählen Sie Erste Schritte.
9. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
10. Wählen Sie auf der Seite Projekte das Projekt aus, dem Sie den Datensatz hinzufügen möchten. Die Detailseite für Ihr Projekt wird angezeigt.
11. Wählen Sie Datensatz erstellen. Die Seite Datensatz erstellen wird angezeigt.

12. Wählen Sie in der Startkonfiguration entweder Mit einem einzelnen Datensatz beginnen oder Mit einem Trainingsdatensatz beginnen aus. Um ein qualitativ hochwertigeres Modell zu erstellen, empfehlen wir, mit separaten Trainings- und Testdatensätzen zu beginnen.

    Single dataset

    a. Wählen Sie im Abschnitt Details zum Trainingsdatensatz die Option Bilder aus dem S3-Bucket importieren aus.
    b. Geben Sie im Abschnitt Details zum Trainingsdatensatz die Informationen für die Schritte 13-15 im Abschnitt Konfiguration der Bildquelle ein.

    Separate training and test datasets

    a. Wählen Sie im Abschnitt Details zum Trainingsdatensatz die Option Bilder aus dem S3-Bucket importieren aus.
    b. Geben Sie im Abschnitt Details zum Trainingsdatensatz die Informationen für die Schritte 13-15 im Abschnitt Konfiguration der Bildquelle ein.
    c. Wählen Sie im Abschnitt Details zum Testdatensatz die Option Bilder aus dem S3-Bucket importieren aus.
    d. Geben Sie im Abschnitt Details zum Testdatensatz die Informationen für die Schritte 13-15 im Abschnitt Konfiguration der Bildquelle ein.
13. Wählen Sie Bilder aus Amazon-S3-Bucket importieren aus.
14. Geben Sie unter S3-URI den Speicherort und den Ordnerpfad des Amazon-S3-Buckets ein.
15. Wählen Sie Je nach Ordner automatisch Labels an Bilder anhängen aus.
16. Wählen Sie Datensätze erstellen aus. Die Datensatzseite für Ihr Projekt wird geöffnet.
17. Wenn Sie Labels hinzufügen oder ändern müssen, führen Sie Labeling von Bildern aus.
18. Folgen Sie den Anweisungen unter Ein Model trainieren (Konsole), um Ihr Modell zu trainieren.

Lokaler Computer

Die Bilder werden direkt von Ihrem Computer geladen. Sie können bis zu 30 Bilder gleichzeitig hochladen.

Den Bildern, die Sie hochladen, sind keine Labels zugeordnet. Weitere Informationen finden Sie unter Labeling von Bildern. Wenn Sie viele Bilder hochladen möchten, sollten Sie einen Amazon-S3-Bucket verwenden. Weitere Informationen finden Sie unter Amazon-S3-Bucket.

> **Note**
>
> Sie können das AWS SDK nicht verwenden, um einen Datensatz mit lokalen Bildern zu erstellen. Erstellen Sie stattdessen eine Manifestdatei und laden Sie die Bilder in einen Amazon-S3-Bucket hoch. Weitere Informationen finden Sie unter Manifestdatei.

So erstellen Sie einen Datensatz mit Bildern auf einem lokalen Computer (Konsole)

1. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Benutzerdefinierte Labels verwenden.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie auf der Seite Projekte das Projekt aus, dem Sie den Datensatz hinzufügen möchten. Die Detailseite für Ihr Projekt wird angezeigt.
6. Wählen Sie Datensatz erstellen. Die Seite Datensatz erstellen wird angezeigt.
7. Wählen Sie in der Startkonfiguration entweder Mit einem einzelnen Datensatz beginnen oder Mit einem Trainingsdatensatz beginnen aus. Um ein qualitativ hochwertigeres Modell zu erstellen, empfehlen wir, mit separaten Trainings- und Testdatensätzen zu beginnen.

   Single dataset

   a. Wählen Sie im Abschnitt Details zum Trainingsdatensatz die Option Bilder von Ihrem Computer hochladen aus.
   b. Wählen Sie Datensatz erstellen.
   c. Wählen Sie auf der Datensatzseite des Projekts die Option Bilder hinzufügen aus.
   d. Wählen Sie die Bilder aus Ihren Computerdateien aus, die Sie in den Datensatz hochladen möchten. Sie können die Bilder ziehen oder die Bilder auswählen, die Sie von Ihrem lokalen Computer hochladen möchten.
   e. Wählen Sie Bilder hochladen.

   Separate training and test datasets

   a. Wählen Sie im Abschnitt Details zum Trainingsdatensatz die Option Bilder von Ihrem Computer hochladen aus.

b. Wählen Sie im Abschnitt Details zum Testdatensatz die Option Bilder von Ihrem Computer hochladen aus.

> **Note**
>
> Ihre Trainings- und Testdatensätze können unterschiedliche Bildquellen haben.

c. Wählen Sie Datensätze erstellen aus. Die Datensatzseite Ihres Projekts wird mit den Registerkarten Training und Test für die jeweiligen Datensätze angezeigt.

d. Wählen Sie Aktionen und anschließend Bilder zum Trainingsdatensatz hinzufügen aus.

e. Wählen Sie die Bilder aus, die Sie in den Datensatz hochladen möchten. Sie können die Bilder ziehen oder die Bilder auswählen, die Sie von Ihrem lokalen Computer hochladen möchten.

f. Wählen Sie Bilder hochladen.

g. Wiederholen Sie die Schritte 5e-5g. Wählen Sie für Schritt 5e Aktionen und dann Bilder zum Testdatensatz hinzufügen aus.

8. Folgen Sie den Schritten unter Labeling von Bildern, um Ihre Bilder zu beschriften.
9. Folgen Sie den Anweisungen unter Ein Model trainieren (Konsole), um Ihr Modell zu trainieren.

Manifestdatei

Sie können einen Datensatz mithilfe einer Manifestdatei SageMaker im Amazon Ground Truth Format erstellen. Sie können die Manifestdatei aus einem Amazon SageMaker Ground Truth Job verwenden. Wenn Ihre Bilder und Beschriftungen nicht das Format einer SageMaker Ground Truth-Manifestdatei haben, können Sie eine Manifestdatei SageMaker im Format erstellen und diese zum Importieren Ihrer beschrifteten Bilder verwenden.

Themen

- Einen Datensatz mit einer SageMaker Ground Truth-Manifestdatei erstellen (Konsole)
- Einen Datensatz mit einer SageMaker Ground Truth Manifest-Datei (SDK) erstellen
- Stellenangebot bei Amazon SageMaker Ground Truth
- Erstellen einer Manifestdatei
- Konvertierung anderer Datensatzformate in eine Manifestdatei

Einen Datensatz mit einer SageMaker Ground Truth-Manifestdatei erstellen (Konsole)

Das folgende Verfahren zeigt Ihnen, wie Sie einen Datensatz mithilfe einer Manifestdatei SageMaker im Ground Truth Format erstellen.

1. Erstellen Sie eine Manifestdatei für Ihren Trainingsdatensatz, indem Sie einen der folgenden Schritte ausführen:

   - Erstellen Sie eine Manifestdatei mit einem SageMaker GroundTruth Job, indem Sie den Anweisungen unter folgenStellenangebot bei Amazon SageMaker Ground Truth.
   - Erstellen Sie Ihre eigene Manifestdatei, indem Sie den Anweisungen unter Erstellen einer Manifestdatei folgen.

   Wenn Sie einen Testdatensatz erstellen möchten, wiederholen Sie Schritt 1, um den Testdatensatz zu erstellen.
2. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
3. Wählen Sie Benutzerdefinierte Labels verwenden.
4. Wählen Sie Erste Schritte.
5. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
6. Wählen Sie auf der Seite Projekte das Projekt aus, dem Sie den Datensatz hinzufügen möchten. Die Detailseite für Ihr Projekt wird angezeigt.
7. Wählen Sie Datensatz erstellen. Die Seite Datensatz erstellen wird angezeigt.
8. Wählen Sie in der Startkonfiguration entweder Mit einem einzelnen Datensatz beginnen oder Mit einem Trainingsdatensatz beginnen aus. Um ein qualitativ hochwertigeres Modell zu erstellen, empfehlen wir, mit separaten Trainings- und Testdatensätzen zu beginnen.

   Single dataset

   a. Wählen Sie im Abschnitt Details zum Trainingsdatensatz die Option Mit SageMaker Ground Truth beschriftete Bilder importieren aus.

   b. Geben Sie im Feld Speicherort der Manifestdatei den Speicherort der Manifestdatei ein, den Sie in Schritt 1 erstellt haben.

   c. Wählen Sie Datensatz erstellen. Die Datensatzseite für Ihr Projekt wird geöffnet.

Separate training and test datasets

a. Wählen Sie im Abschnitt Details zum Trainingsdatensatz die Option Mit SageMaker Ground Truth beschriftete Bilder importieren aus.

b. Geben Sie im Feld Speicherort der Manifestdatei den Speicherort der Trainingsdatensatz-Manifestdatei ein, den Sie in Schritt 1 erstellt haben.

c. Wählen Sie im Abschnitt Details zum Testdatensatz die Option Bilder importieren mit der Bezeichnung SageMaker Ground Truth aus.

> **Note**
>
> Ihre Trainings- und Testdatensätze können unterschiedliche Bildquellen haben.

d. Geben Sie im Feld Speicherort der Manifestdatei den Speicherort der Testdatensatz-Manifestdatei ein, den Sie in Schritt 1 erstellt haben.

e. Wählen Sie Datensätze erstellen aus. Die Datensatzseite für Ihr Projekt wird geöffnet.

9. Wenn Sie Labels hinzufügen oder ändern müssen, führen Sie Labeling von Bildern aus.
10. Folgen Sie den Anweisungen unter Ein Model trainieren (Konsole), um Ihr Modell zu trainieren.

Einen Datensatz mit einer SageMaker Ground Truth Manifest-Datei (SDK) erstellen

Das folgende Verfahren zeigt Ihnen, wie Sie mithilfe der CreateDatasetAPI Trainings- oder Testdatensätze aus einer Manifestdatei erstellen.

Sie können eine vorhandene Manifestdatei verwenden, z. B. die Ausgabe eines SageMaker Ground Truth Jobs, oder Ihre eigene Manifestdatei erstellen.

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Erstellen Sie eine Manifestdatei für Ihren Trainingsdatensatz, indem Sie einen der folgenden Schritte ausführen:

   - Erstellen Sie eine Manifestdatei mit einem SageMaker GroundTruth Job, indem Sie den Anweisungen unter folgenStellenangebot bei Amazon SageMaker Ground Truth.

- Erstellen Sie Ihre eigene Manifestdatei, indem Sie den Anweisungen unter Erstellen einer Manifestdatei folgen.

Wenn Sie einen Testdatensatz erstellen möchten, wiederholen Sie Schritt 2, um den Testdatensatz zu erstellen.

3. Verwenden Sie den folgenden Beispielcode, um den Trainings- und Testdatensatz zu erstellen.

AWS CLI

Erstellen Sie einen Datensatz mit dem folgenden Code. Ersetzen Sie Folgendes:

- `project_arn`— den ARN des Projekts, dem Sie den Testdatensatz hinzufügen möchten.
- `type` — den Typ des Datensatzes, den Sie erstellen möchten (TRAINIEREN oder TESTEN)
- `bucket` — den Bucket, der die Manifestdatei für den Datensatz enthält.
- `manifest_file` — den Pfad und Namen der Manifestdatei

```
aws rekognition create-dataset --project-arn project_arn \
  --dataset-type type \
  --dataset-source '{ "GroundTruthManifest": { "S3Object": { "Bucket": "bucket",
 "Name": "manifest_file" } } }' \
  --profile custom-labels-access
```

Python

Erstellen Sie einen Datensatz mit den folgenden Werten. Geben Sie die folgenden Befehlszeilenparameter an:

- `project_arn` — den ARN des Projekts, dem Sie den Testdatensatz hinzufügen möchten.
- `dataset_type` — den Typ des Datensatzes, den Sie erstellen möchten (`train` oder `test`).
- `bucket` — den Bucket, der die Manifestdatei für den Datensatz enthält.
- `manifest_file` — den Pfad und Namen der Manifestdatei

```
#Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)


import argparse
import logging
import time
import json
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_dataset(rek_client, project_arn, dataset_type, bucket,
 manifest_file):
    """
    Creates an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
 dataset.
    :param dataset_type: The type of the dataset that you want to create (train
 or test).
    :param bucket: The S3 bucket that contains the manifest file.
    :param manifest_file: The path and filename of the manifest file.
    """

    try:
        #Create the project
        logger.info("Creating %s dataset for project %s",dataset_type,
 project_arn)

        dataset_type = dataset_type.upper()

        dataset_source = json.loads(
            '{ "GroundTruthManifest": { "S3Object": { "Bucket": "'
            + bucket
            + '", "Name": "'
            + manifest_file
            + '" } } }'
        )

        response = rek_client.create_dataset(
```

```
            ProjectArn=project_arn, DatasetType=dataset_type,
 DatasetSource=dataset_source
        )

        dataset_arn=response['DatasetArn']

        logger.info("dataset ARN: %s",dataset_arn)

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']

            if status == "CREATE_IN_PROGRESS":
                logger.info("Creating dataset: %s ",dataset_arn)
                time.sleep(5)
                continue

            if status == "CREATE_COMPLETE":
                logger.info("Dataset created: %s", dataset_arn)
                finished=True
                continue

            if status == "CREATE_FAILED":
                error_message = f"Dataset creation failed: {status} :
 {dataset_arn}"
                logger.exception(error_message)
                raise Exception (error_message)

            error_message = f"Failed. Unexpected state for dataset creation:
 {status} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        return dataset_arn


    except ClientError as err:
        logger.exception("Couldn't create dataset: %s",err.response['Error']
['Message'])
        raise
```

```
def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
 the dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the dataset that you want to create
 (train or test)."
    )

    parser.add_argument(
        "bucket", help="The S3 bucket that contains the manifest file."
    )

    parser.add_argument(
        "manifest_file", help="The path and filename of the manifest file."
    )


def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        #Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Creating {args.dataset_type} dataset for project
 {args.project_arn}")

        #Create the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn=create_dataset(rekognition_client,
```

```
            args.project_arn,
            args.dataset_type,
            args.bucket,
            args.manifest_file)

        print(f"Finished creating dataset: {dataset_arn}")


    except ClientError as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")



if __name__ == "__main__":
    main()
```

Java V2

Erstellen Sie einen Datensatz mit den folgenden Werten. Geben Sie die folgenden Befehlszeilenparameter an:

- `project_arn` — den ARN des Projekts, dem Sie den Testdatensatz hinzufügen möchten.
- `dataset_type` — den Typ des Datensatzes, den Sie erstellen möchten (`train` oder `test`).
- `bucket` — den Bucket, der die Manifestdatei für den Datensatz enthält.
- `manifest_file` — den Pfad und Namen der Manifestdatei

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
```

```
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetSource;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.S3Object;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateDatasetManifestFiles {

    public static final Logger logger =
 Logger.getLogger(CreateDatasetManifestFiles.class.getName());

    public static String createMyDataset(RekognitionClient rekClient, String
 projectArn, String datasetType,
            String bucket, String name) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating {0} dataset for project : {1} from
 s3://{2}/{3} ",
                    new Object[] { datasetType, projectArn, bucket, name });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Could not create dataset. Unrecognized
 dataset type: {0}", datasetType);
                throw new Exception("Could not create dataset. Unrecognized
 dataset type: " + datasetType);
```

```
            }

            GroundTruthManifest groundTruthManifest =
GroundTruthManifest.builder()

.s3Object(S3Object.builder().bucket(bucket).name(name).build()).build();

            DatasetSource datasetSource =
DatasetSource.builder().groundTruthManifest(groundTruthManifest).build();

            CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)

.datasetType(requestDatasetType).datasetSource(datasetSource).build();

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(response.datasetArn()).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

                switch (status) {

                case CREATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset created");
                    created = true;
                    break;

                case CREATE_IN_PROGRESS:
```

```
                    Thread.sleep(5000);
                    break;

                case CREATE_FAILED:
                    String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (created == false);

            return response.datasetArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String datasetType = null;
        String bucket = null;
        String name = null;
        String projectArn = null;
        String datasetArn = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
<dataset_arn>\n\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
copy the datast to.\n\n"
```

```
                + "   dataset_type - the type of the dataset that you want to
 create (train or test).\n\n"
                + "   bucket - the S3 bucket that contains the manifest file.\n
\n"
                + "   name - the location and name of the manifest file within
 the bucket.\n\n";

        if (args.length != 4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        datasetType = args[1];
        bucket = args[2];
        name = args[3];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();


             // Create the dataset
            datasetArn = createMyDataset(rekClient, projectArn, datasetType,
 bucket, name);

            System.out.println(String.format("Created dataset: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }
```

```
    }

}
```

4. Informationen zum Hinzufügen oder Ändern von Labels finden Sie unter Labels verwalten (SDK).
5. Folgen Sie den Anweisungen unter Ein Modell trainieren (SDK), um Ihr Modell zu trainieren.

Stellenangebot bei Amazon SageMaker Ground Truth

Mit Amazon SageMaker Ground Truth können Sie Mitarbeiter von Amazon Mechanical Turk, einem von Ihnen ausgewählten Anbieter, oder interne, private Mitarbeiter zusammen mit maschinellem Lernen einsetzen, mit dem Sie beschriftete Bilder erstellen können. Amazon Rekognition Custom Labels importiert SageMaker Ground Truth Truth-Manifestdateien aus einem Amazon S3 S3-Bucket, den Sie angeben.

Amazon Rekognition Custom Labels unterstützt die folgenden SageMaker Ground Truth Truth-Aufgaben.

- Bildklassifizierung
- Begrenzungsrahmen

Bei den Dateien, die Sie importieren, handelt es sich um die Bilder und eine Manifestdatei. Die Manifestdatei enthält Label- und Begrenzungsrahmen-Informationen für die Bilder, die Sie importieren.

Amazon Rekognition benötigt Berechtigungen für den Zugriff auf den Amazon-S3-Bucket, in dem Ihre Bilder gespeichert sind. Wenn Sie den Konsolen-Bucket verwenden, der von Amazon Rekognition Custom Labels für Sie eingerichtet wurde, sind die erforderlichen Berechtigungen bereits eingerichtet. Wenn Sie den Konsolen-Bucket nicht verwenden, siehe Zugreifen auf externe Amazon-S3-Buckets.

Erstellen einer Manifestdatei mit einem SageMaker Ground Truth Job (Konsole)

Das folgende Verfahren zeigt Ihnen, wie Sie einen Datensatz mithilfe von Bildern erstellen, die mit einem SageMaker Ground-Truth-Job beschriftet wurden. Die Job-Ausgabedateien werden in Ihrem Amazon Rekognition Custom Labels-Konsolen-Bucket gespeichert.

So erstellen Sie einen Datensatz mit Bildern, die mit einem SageMaker Ground Truth Job gekennzeichnet sind (Konsole)

1. Melden Sie sich bei der Amazon S3 S3-Konsole an AWS Management Console und öffnen Sie sie unter https://console.aws.amazon.com/s3/.
2. Erstellen Sie im Konsolen-Bucket einen Ordner für Ihre Trainingsbilder.

   > **Note**
   >
   > Der Konsolen-Bucket wird erstellt, wenn Sie die Amazon Rekognition Custom Labels-Konsole zum ersten Mal in einer AWS Region öffnen. Weitere Informationen finden Sie unter Verwalten eines Amazon Rekognition Custom Labels-Projekts.

3. Laden Sie Ihre Bilder in den Ordner hoch, den Sie gerade erstellt haben.
4. Erstellen Sie im Konsolen-Bucket einen Ordner für die Ausgabe des Ground Truth-Jobs.
5. Öffnen Sie die SageMaker Konsole unter https://console.aws.amazon.com/sagemaker/.
6. Erstellen Sie einen Ground Truth Labeling-Job. Sie benötigen die Amazon S3-URLs für die Ordner, die Sie in Schritt 2 und Schritt 4 erstellt haben. Weitere Informationen finden Sie unter Verwenden von Amazon SageMaker Ground Truth für die Datenkennzeichnung.
7. Notieren Sie sich den Speicherort der `output.manifest`-Datei in dem Ordner, den Sie in Schritt 4 erstellt haben. Sie sollte sich im Unterordner *`Ground-Truth-Job-Name`*`/manifests/output` befinden.
8. Folgen Sie den Anweisungen unter Einen Datensatz mit einer SageMaker Ground Truth-Manifestdatei erstellen (Konsole), um einen Datensatz mit der hochgeladenen Manifestdatei zu erstellen. Geben Sie für Schritt 8 unter Speicherort der Manifestdatei die Amazon S3-URL für den Speicherort ein, den Sie im vorherigen Schritt notiert haben. Wenn Sie das AWS SDK verwenden, tun Sie diesEinen Datensatz mit einer SageMaker Ground Truth Manifest-Datei (SDK) erstellen.
9. Wiederholen Sie die Schritte 1 bis 6, um einen SageMaker Ground Truth Job für Ihren Testdatensatz zu erstellen.

Erstellen einer Manifestdatei

Sie können einen Test- oder Trainingsdatensatz erstellen, indem Sie eine Manifestdatei SageMaker im Ground Truth Format importieren. Wenn Ihre Bilder in einem Format beschriftet sind, das keine

SageMaker Ground-Truth-Manifestdatei ist, verwenden Sie die folgenden Informationen, um eine Manifestdatei SageMaker im Ground-Truth-Format zu erstellen.

Manifestdateien haben das JSON-Zeilen-Format, wobei jede Zeile ein vollständiges JSON-Objekt ist, das die Label-Informationen für ein Bild darstellt. Amazon Rekognition Custom Labels unterstützt SageMaker Ground-Truth-Manifeste mit JSON-Zeilen in den folgenden Formaten:

- Klassifizierungsjob-Output — Wird verwendet, um einem Bild Labels auf Bildebene hinzuzufügen. Ein Label auf Bildebene definiert die Klasse der Szene, des Konzepts oder des Objekts (falls keine Informationen für die Objektposition benötigt werden), die sich auf einem Bild befindet. Ein Bild kann mehr als eine Bezeichnung auf Bildebene haben. Weitere Informationen finden Sie unter Labels auf Bildebene in Manifestdateien.
- Begrenzungsrahmenjob-Output — Wird verwendet, um ein Label für Klasse und Position eines oder mehrerer Objekte auf einem Bild zu erstellen. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien.

JSON-Zeilen auf Bildebene und Lokalisierungszeilen (Begrenzungsrahmen) können in derselben Manifestdatei miteinander verkettet werden.

> **Note**
>
> Die Beispiele für JSON-Zeilen in diesem Abschnitt sind aus Gründen der Lesbarkeit formatiert.

Wenn Sie eine Manifestdatei importieren, wendet Amazon Rekognition Custom Labels Validierungsregeln für Grenzwerte, Syntax und Semantik an. Weitere Informationen finden Sie unter Validierungsregeln für Manifestdateien.

Die Bilder, auf die eine Manifestdatei verweist, müssen sich in demselben Amazon-S3-Bucket befinden. Die Manifestdatei kann sich in einem anderen Amazon-S3-Bucket befinden als der Amazon-S3-Bucket, in dem die Bilder gespeichert sind. Sie geben die Position eines Bildes im `source-ref`-Feld einer JSON-Zeile an.

Amazon Rekognition benötigt Berechtigungen für den Zugriff auf den Amazon-S3-Bucket, in dem Ihre Bilder gespeichert sind. Wenn Sie den Konsolen-Bucket verwenden, der von Amazon Rekognition Custom Labels für Sie eingerichtet wurde, sind die erforderlichen Berechtigungen bereits eingerichtet. Wenn Sie den Konsolen-Bucket nicht verwenden, siehe Zugreifen auf externe Amazon-S3-Buckets.

Themen

Erstellen einer Manifestdatei

Mit dem folgenden Verfahren wird ein Projekt mit einem Trainings- und Testdatensatz erstellt. Die Datensätze werden aus den von Ihnen erstellten Trainings- und Testmanifestdateien erstellt.

So erstellen Sie einen Datensatz mit einer Manifestdatei SageMaker im Ground Truth Format (Konsole)

1. Erstellen Sie im Konsolen-Bucket einen Ordner für Ihre Manifestdateien.
2. Erstellen Sie in dem Konsolen-Bucket einen Ordner, um Ihre Bilder zu speichern.
3. Laden Sie Ihre Bilder in den nun erstellten Ordner hoch.
4. Erstellen Sie eine Manifestdatei SageMaker im Ground Truth Format für Ihren Trainingsdatensatz. Weitere Informationen finden Sie unter Labels auf Bildebene in Manifestdateien und Objektlokalisierung in Manifestdateien.

   > **Important**
   >
   > Der `source-ref`-Feldwert in jeder JSON-Zeile muss einem Bild zugeordnet sein, das Sie hochgeladen haben.

5. Erstellen Sie eine Manifestdatei SageMaker im Ground Truth Format für Ihren Testdatensatz.
6. Laden Sie Ihre Manifestdateien in den Ordner hoch, den Sie gerade erstellt haben.
7. Notieren Sie sich den Namen der Manifestdatei.
8. Folgen Sie den Anweisungen unter Einen Datensatz mit einer SageMaker Ground Truth-Manifestdatei erstellen (Konsole), um einen Datensatz mit der hochgeladenen Manifestdatei zu erstellen. Geben Sie für Schritt 8 unter Speicherort der Manifestdatei die Amazon S3-URL für den Speicherort ein, den Sie im vorherigen Schritt notiert haben. Wenn Sie das AWS SDK verwenden, tun Sie diesEinen Datensatz mit einer SageMaker Ground Truth Manifest-Datei (SDK) erstellen.

Labels auf Bildebene in Manifestdateien

Um Beschriftungen auf Bildebene zu importieren (Bilder, die mit Szenen, Konzepten oder Objekten beschriftet sind, für die keine Lokalisierungsinformationen erforderlich sind), fügen Sie einer Manifestdatei JSON-Zeilen SageMaker im Ground Truth Classification Job Output-Format hinzu. Eine Manifestdatei besteht aus einer oder mehreren JSON-Zeilen, eine für jedes Bild, das Sie importieren möchten.

> **Tip**
>
> Um die Erstellung einer Manifestdatei zu vereinfachen, stellen wir ein Python-Skript zur Verfügung, das eine Manifestdatei aus einer CSV-Datei erstellt. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei aus einer CSV-Datei.

So erstellen Sie eine Manifestdatei für Labels auf Bildebene.

1. Erstellen Sie eine leere Textdatei.
2. Fügen Sie eine JSON-Zeile für jedes Bild hinzu, das Sie importieren möchten. Jede JSON-Zeile sollte nun etwa folgendermaßen aussehen.

```
{"source-ref":"s3://custom-labels-console-us-east-1-nnnnnnnnnn/gt-job/
manifest/IMG_1133.png","TestCLConsoleBucket":0,"TestCLConsoleBucket-
metadata":{"confidence":0.95,"job-name":"labeling-job/
testclconsolebucket","class-name":"Echo Dot","human-annotated":"yes","creation-
date":"2020-04-15T20:17:23.433061","type":"groundtruth/image-classification"}}
```

3. Speichern Sie die Datei. Sie können die Erweiterung `.manifest` verwenden, sie ist jedoch nicht erforderlich.
4. Erstellen Sie einen Datensatz mit der von Ihnen erstellten Manifestdatei. Weitere Informationen finden Sie unter So erstellen Sie einen Datensatz mit einer Manifestdatei SageMaker im Ground Truth Format (Konsole).

JSON-Zeilen auf Bildebene

In diesem Abschnitt zeigen wir Ihnen, wie Sie eine JSON-Zeile für ein einzelnes Bild erstellen. Betrachten Sie das folgende Bild: Eine Szene für das folgende Bild könnte Sonnenaufgang heißen.

Die JSON-Zeile für das vorherige Bild mit der Szene Sonnenaufgang könnte wie folgt aussehen.

```
{
    "source-ref": "s3://bucket/images/sunrise.png",
    "testdataset-classification_Sunrise": 1,
    "testdataset-classification_Sunrise-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/testdataset-classification_Sunrise",
        "class-name": "Sunrise",
        "human-annotated": "yes",
        "creation-date": "2020-03-06T17:46:39.176",
        "type": "groundtruth/image-classification"
    }
}
```

Notieren Sie die folgenden Informationen:

Quellennachweis

(Erforderlich) Der Amazon S3-Speicherort des Bildes. Das Format ist "s3://*BUCKET*/*OBJECT_PATH*". Bilder in einem importierten Datensatz müssen im gleichen Amazon-S3-Bucket gespeichert werden.

Testdatensatz-Klassifizierung_Sonnenaufgang

(Erforderlich) das Label-Attribut. Sie wählen den Feldnamen. Der Feldwert (1 im vorherigen Beispiel) ist ein Bezeichner für ein Labelattribut. Er wird von Amazon Rekognition Custom Labels nicht verwendet und kann eine beliebige Ganzzahl sein. Es müssen entsprechende Metadaten vorhanden sein, die durch den Feldnamen mit angehängtem -Metadaten identifiziert werden. z. B. "testdataset-classification_Sunrise-metadata".

Testdatensatz-Klassifizierung_Sonnenaufgang-Metadaten

(Erforderlich) Metadaten zum Label-Attribut. Der Feldname muss mit dem Label-Attribut identisch sein, wobei -Metadaten angehängt ist.

Konfidenz

(Erforderlich) Wird derzeit nicht von Amazon Rekognition Custom Labels verwendet, aber es muss ein Wert zwischen 0 und 1 angegeben werden.

Jobname

(Optional) Ein Name, den Sie für den Job wählen, der das Bild verarbeitet.

Klassenname

(Erforderlich) Ein Klassenname, den Sie für die Szene oder das Konzept wählen, das auf das Bild zutrifft. z. B. "Sunrise".

mit menschlichen Anmerkungen versehen

(Erforderlich) Geben Sie "yes" an, wenn die Anmerkung von einem Menschen ausgefüllt wurde. Andernfalls "no".

Erstellungsdatum

(Erforderlich) Das Datum und die Uhrzeit in koordinierter Weltzeit (UTC), zu der das Label erstellt wurde.

Typ

(Erforderlich) Die Art der Verarbeitung, die auf das Bild angewendet werden soll. Für Labels auf Bildebene ist der Wert `"groundtruth/image-classification"`.

Hinzufügen mehrerer Labels auf Bildebene zu einem Bild

Sie können einem Bild mehrere Labels hinzufügen. Mit dem folgenden JSON-Code werden beispielsweise zwei Labels, Fußball und Ball, zu einem einzigen Bild hinzugefügt.

```
{
    "source-ref": "S3 bucket location",
    "sport0":0, # FIRST label
    "sport0-metadata": {
        "class-name": "football",
        "confidence": 0.8,
        "type":"groundtruth/image-classification",
        "job-name": "identify-sport",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256"
    },
    "sport1":1, # SECOND label
    "sport1-metadata": {
        "class-name": "ball",
        "confidence": 0.8,
        "type":"groundtruth/image-classification",
        "job-name": "identify-sport",
        "human-annotated": "yes",
        "creation-date": "2018-10-18T22:18:13.527256"
    }
}  # end of annotations for 1 image
```

Objektlokalisierung in Manifestdateien

Sie können Bilder importieren, die mit Informationen zur Objektlokalisierung beschriftet sind, indem Sie JSON-Zeilen SageMaker im Ground Truth Bounding Box Job Output-Format zu einer Manifestdatei hinzufügen.

Lokalisierungsinformationen stellen die Position eines Objekts auf einem Bild dar. Die Position wird durch einen Begrenzungsrahmen dargestellt, der das Objekt umgibt. Die Begrenzungsrahmenstruktur enthält die oberen linken Koordinaten des Begrenzungsrahmens sowie die Breite und Höhe des

Begrenzungsrahmens. Eine JSON-Zeile im Begrenzungsrahmen-Format enthält Begrenzungsrahmen für die Positionen eines oder mehrerer Objekte auf einem Bild und die Klasse jedes Objekts auf dem Bild.

Eine Manifestdatei besteht aus einer oder mehreren JSON-Zeilen, wobei jede Zeile die Informationen für ein einzelnes Bild enthält.

So erstellen Sie eine Manifestdatei zur Objektlokalisierung

1. Erstellen Sie eine leere Textdatei.
2. Fügen Sie eine JSON-Zeile für jedes Bild hinzu, das Sie importieren möchten. Jede JSON-Zeile sollte nun etwa folgendermaßen aussehen.

```
{"source-ref": "s3://bucket/images/IMG_1186.png", "bounding-box": {"image_size":
 [{"width": 640, "height": 480, "depth": 3}], "annotations": [{ "class_id":
 1, "top": 251, "left": 399, "width": 155, "height": 101}, {"class_id": 0,
 "top": 65, "left": 86, "width": 220, "height": 334}]}, "bounding-box-metadata":
 {"objects": [{ "confidence": 1}, {"confidence": 1}], "class-map": {"0": "Echo",
 "1": "Echo Dot"}, "type": "groundtruth/object-detection", "human-annotated":
 "yes", "creation-date": "2013-11-18T02:53:27", "job-name": "my job"}}
```

3. Speichern Sie die Datei. Sie können die Erweiterung `.manifest` verwenden, sie ist jedoch nicht erforderlich.
4. Erstellen Sie mit der Datei, die Sie gerade erstellt haben, einen Datensatz. Weitere Informationen finden Sie unter So erstellen Sie einen Datensatz mit einer Manifestdatei SageMaker im Ground Truth Format (Konsole).

JSON-Zeilen, die das Objekt begrenzen

In diesem Abschnitt zeigen wir Ihnen, wie Sie eine JSON-Zeile für ein einzelnes Bild erstellen. Das folgende Bild zeigt Begrenzungsrahmen rund um Amazon Echo- und Amazon Echo Dot-Geräte.

Das Folgende ist die JSON-Begrenzungsrahmen-Zeile für das vorherige Bild.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
```

```
    "height": 101
   }, {
    "class_id": 0,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
   }]
  },
  "bounding-box-metadata": {
   "objects": [{
    "confidence": 1
   }, {
    "confidence": 1
   }],
   "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2013-11-18T02:53:27",
   "job-name": "my job"
  }
}
```

Notieren Sie die folgenden Informationen:

Quellennachweis

(Erforderlich) Der Amazon S3-Speicherort des Bildes. Das Format ist "s3://*BUCKET*/*OBJECT_PATH*". Bilder in einem importierten Datensatz müssen im gleichen Amazon-S3-Bucket gespeichert werden.

Begrenzungsrahmen

(Erforderlich) Das Label-Attribut. Sie wählen den Feldnamen. Enthält die Bildgröße und die Begrenzungsrahmen für jedes Objekt, das im Bild erkannt wurde. Es müssen entsprechende Metadaten vorhanden sein, die durch den Feldnamen mit angehängtem -Metadaten identifiziert werden. z. B. "bounding-box-metadata".

Bildgröße

(Erforderlich) Ein Array mit einem einzelnen Element, das die Größe des Bildes in Pixeln enthält.

- Höhe - (Erforderlich) Die Höhe, in Pixeln, des Bildes.
- Breite — (Erforderlich) Die Tiefe des Bildes in Pixeln.
- Tiefe — (Erforderlich) Die Anzahl der Kanäle im Bild. Die Wert für RGB-Bilder ist 3. Wird derzeit nicht von Amazon Rekognition Custom Labels verwendet, aber ein Wert ist erforderlich.

Anmerkungen

(Erforderlich) Eine Reihe von Bounding-Box-Informationen für jedes Objekt, das im Bild erkannt wurde.

- Klassen_ID — (Erforderlich) Ordnet dem Label in Klassenzuordnung zu. Im vorherigen Beispiel ist das Objekt mit der Klassen_ID von 1 der Echo Dot im Bild.
- oben — (Erforderlich) Der Abstand zwischen dem oberen Bildrand und dem oberen Rand des Begrenzungsrahmens in Pixeln.
- links — (Erforderlich) Der Abstand von der linken Seite des Bilds zur linken Seite des Begrenzungsrahmens in Pixeln.
- Breite — (Erforderlich) Die Breite des Begrenzungsrahmens in Pixeln.
- Höhe - (Erforderlich) Die Höhe des Begrenzungsrahmens in Pixeln.

***Begrenzungsrahmen*-Metadaten**

(Erforderlich) Metadaten zum Label-Attribut. Der Feldname muss mit dem Label-Attribut identisch sein, wobei -Metadaten angehängt ist. Eine Reihe von Begrenzungsrahmen-Informationen für jedes Objekt, das im Bild erkannt wurde.

Objekte

(Erforderlich) Ein Array von Objekten im Bild. Ordnet dem Array Anmerkungen nach Index zu. Das Vertrauensattribut wird von Amazon Rekognition Custom Labels nicht verwendet.

Klassenzuordnung

(Erforderlich) Eine Zuordnung der Klassen, die für die im Bild erkannten Objekte gelten.

Typ

(Erforderlich) Der Typ des Klassifizierungsauftrags. `"groundtruth/object-detection"` identifiziert den Job als Objekterkennung.

Erstellungsdatum

(Erforderlich) Das Datum und die Uhrzeit in koordinierter Weltzeit (UTC), zu der das Label erstellt wurde.

Mit menschlichen Anmerkungen versehen

(Erforderlich) Geben Sie `"yes"` an, wenn die Anmerkung von einem Menschen ausgefüllt wurde. Andernfalls `"no"`.

Jobname

(Optional) Der Name des Jobs, der das Bild verarbeitet.

Validierungsregeln für Manifestdateien

Wenn Sie eine Manifestdatei importieren, wendet Amazon Rekognition Custom Labels Validierungsregeln für Grenzwerte, Syntax und Semantik an. Das SageMaker Ground Truth Schema erzwingt die Syntaxvalidierung. Weitere Informationen finden Sie unter Ausgaben: Im Folgenden finden Sie die Validierungsregeln für Grenzwerte und Semantik.

> **Note**
>
> - Die Ungültigkeitsregeln von 20 % gelten kumulativ für alle Validierungsregeln. Wenn der Import die 20 % -Grenze aufgrund einer beliebigen Kombination überschreitet, z. B. aufgrund von 15 % ungültigem JSON und 15 % ungültigen Bildern, schlägt der Import fehl.
> - Jedes Datensatz-Objekt ist eine Zeile im Manifest. Leere/ungültige Zeilen werden ebenfalls als Datensatzobjekte gezählt.
> - Überlappungen sind (gemeinsame Labels zwischen Test und Training)/(Trainingslabels).

Themen

- Einschränkungen
- Semantik

Einschränkungen

Validierung	Limit	Es wurde ein Fehler gemeldet
Größe der Manifestdatei	Maximal 1 GB	Fehler
Maximale Zeilenanzahl für eine Manifestdatei	Maximal 250 000 Datensatz -Objekte als Zeilen in einem Manifest.	Fehler
Untere Grenze für die Gesamtzahl gültiger Datensatz -Objekte pro Label	>=1	Fehler
Untere Grenze auf Labels	>=2	Fehler
Obere Grenze auf Labels	<= 250	Fehler
Minimale Anzahl von Begrenzungsrahmen pro Bild	0	None
Maximale Anzahl von Begrenzungsrahmen pro Bild	50	None

Semantik

Validierung	Limit	Es wurde ein Fehler gemeldet
Leeres Manifest		Fehler
Fehlendes oder unzugängl iches Quellennachweisobjekt	Anzahl der Objekte weniger als 20 %	Warnung
Fehlendes oder unzugängl iches Quellennachweisobjekt	Anzahl der Objekte > 20 %	Fehler
Test-Labels sind im Trainings datensatz nicht vorhanden	Die Labels überlappen sich mindestens zu 50 %	Fehler

Validierung	Limit	Es wurde ein Fehler gemeldet
Mischung aus Label-Bei spielen und Objektbeispielen für dasselbe Label in einem Datensatz. Klassifizierung und Erkennung für dieselbe Klasse in einem Datensatzobjekt.		Kein Fehler oder keine Warnung
Überlappende Ressourcen zwischen Test und Training	Es sollte keine Überschne idung zwischen Test- und Trainingsdatensätzen geben.	
Die Bilder in einem Datensatz müssen aus demselben Bucket stammen	Fehler, wenn sich die Objekte in einem anderen Bucket befinden	Fehler

Konvertierung anderer Datensatzformate in eine Manifestdatei

Sie können die folgenden Informationen verwenden, um Manifestdateien SageMaker im Amazon-Format aus einer Vielzahl von Quelldatensatzformaten zu erstellen. Nachdem Sie die Manifestdatei erstellt haben, verwenden Sie sie, um einen Datensatz zu erstellen. Weitere Informationen finden Sie unter Manifestdatei.

Themen

- Formatieren eines COCO-Datensatzes
- Transformation von SageMaker Ground Truth-Manifestdateien mit mehreren Labels
- Erstellen einer Manifestdatei aus einer CSV-Datei

Formatieren eines COCO-Datensatzes

COCO ist ein Format zur Spezifizierung umfangreicher Datensätze zur Objekterkennung, Segmentierung und Untertitelung. Dieses Python-Beispiel zeigt Ihnen, wie Sie einen Datensatz im COCO-Objekterkennungsformat in eine Manifestdatei im Begrenzungsrahmen-Format Amazon Rekognition Custom Labels umwandeln. Dieser Abschnitt enthält auch Informationen, mit denen Sie Ihren eigenen Code schreiben können.

Eine JSON-Datei im COCO-Format besteht aus fünf Abschnitten, die Informationen für einen gesamten Datensatz enthalten. Weitere Informationen finden Sie unter COCO-Format.

- `info` — allgemeine Informationen über den Datensatz.
- `licenses` — Lizenzinformationen für die Bilder im Datensatz.
- `images` — eine Liste der Bilder im Datensatz.
- `annotations` — eine Liste von Anmerkungen (einschließlich Begrenzungsrahmen), die in allen Bildern des Datensatzes vorhanden sind.
- `categories` — eine Liste von Label-Kategorien.

Sie benötigen Informationen aus den Listen `images`, `annotations` und `categories`, um eine Amazon Rekognition Custom Labels-Manifestdatei zu erstellen.

Eine Amazon Rekognition Custom Labels-Manifestdatei hat das JSON-Zeilenformat, wobei jede Zeile den Begrenzungsrahmen und die Labelinformationen für ein oder mehrere Objekte auf einem Bild enthält. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien.

Zuordnung von COCO-Objekten zu einer JSON-Zeile mit benutzerdefinierten Labels

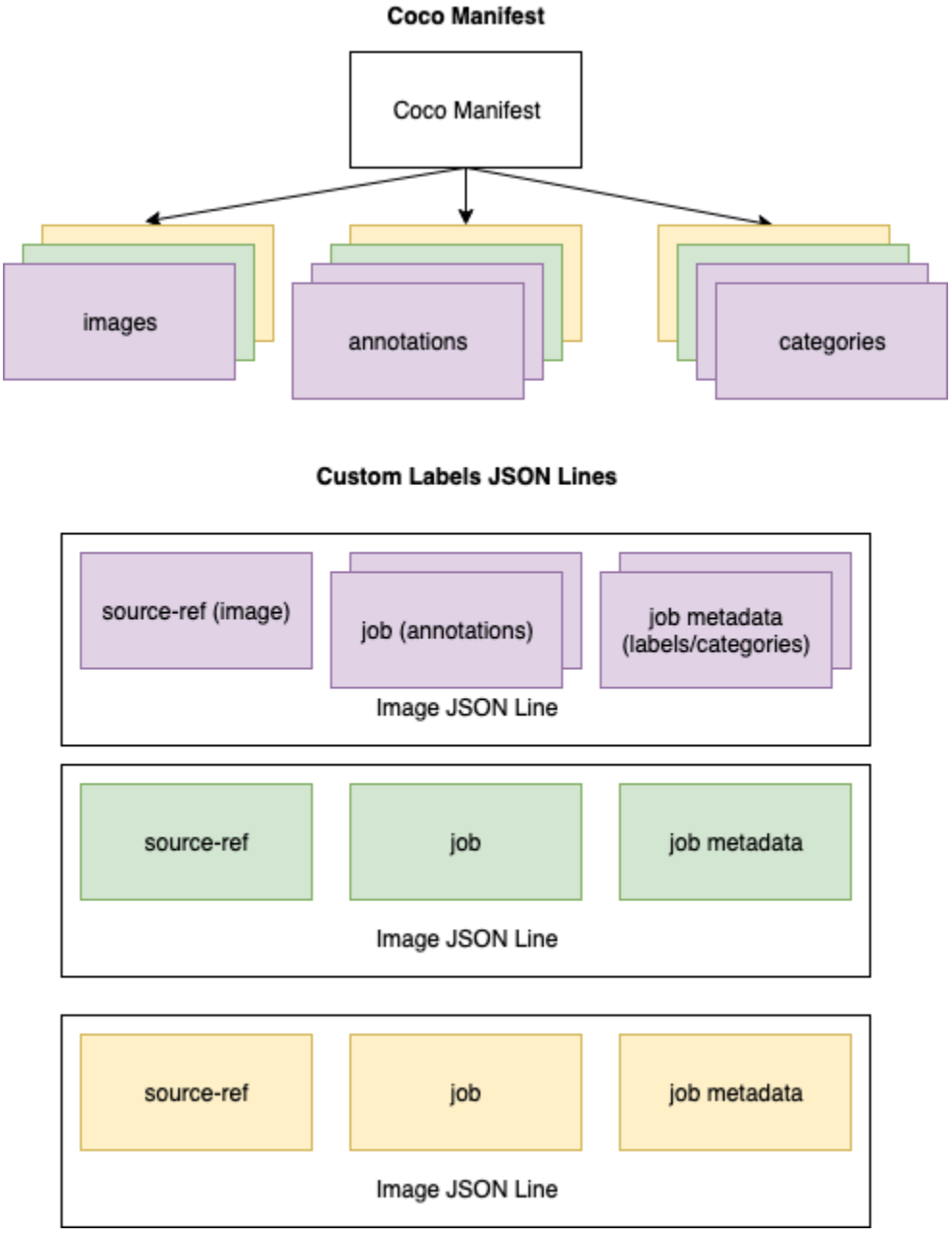
Um einen Datensatz im COCO-Format zu transformieren, ordnen Sie den COCO-Datensatz einer Amazon Rekognition Custom Labels-Manifestdatei für die Objektlokalisierung zu. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien. Um eine JSON-Zeile für jedes Bild zu erstellen, muss die Manifestdatei den COCO-Datensatz `image`, `annotation`und `category`-Objektfeld-IDs zuordnen.

Folgendes ist ein Beispiel für den Inhalt einer COCO-Manifestdatei: Weitere Informationen finden Sie unter COCO-Format.

```
{
    "info": {
        "description": "COCO 2017 Dataset","url": "http://cocodataset.org","version":
 "1.0","year": 2017,"contributor": "COCO Consortium","date_created": "2017/09/01"
    },
    "licenses": [
        {"url": "http://creativecommons.org/licenses/by/2.0/","id": 4,"name":
 "Attribution License"}
    ],
    "images": [
```

```
        {"id": 242287, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/xxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
xxxxxxxxxxxx.jpg", "width": 426, "height": 640, "file_name": "xxxxxxxxx.jpg",
 "date_captured": "2013-11-15 02:41:42"},
        {"id": 245915, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/nnnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnn.jpg",
 "date_captured": "2013-11-18 02:53:27"}
    ],
    "annotations": [
        {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
 417.51,......167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
 [19.23, 383.18, 314.5, 244.46]},
        {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
 238.8,........382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
 "bbox": [399, 251, 155, 101]},
        {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
 239.01,..........495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
 "bbox": [86, 65, 220, 334]}
    ],
    "categories": [
        {"supercategory": "speaker","id": 0,"name": "echo"},
        {"supercategory": "speaker","id": 1,"name": "echo dot"}
    ]
}
```

Das folgende Diagramm zeigt, wie die COCO-Datensatzlisten für einen Datensatz den JSON-Zeilen von Amazon Rekognition Custom Labels für ein Bild zugeordnet werden. Übereinstimmende Farben geben Informationen für ein einzelnes Bild an.



So rufen Sie die COCO-Objekte für eine einzelne JSON-Zeile ab

1. Rufen Sie für jedes Bild in der Bilderliste die Anmerkung aus der Anmerkungsliste ab, bei der der Wert des Anmerkungsfeldes `image_id` mit dem `id`-Bildfeld übereinstimmt.
2. Lesen Sie für jede Anmerkung, die in Schritt 1 gefunden wurde, die `categories`-Liste durch und ermitteln Sie jede `category`, bei der der Wert des `category`-Feldes `id` mit dem `annotation` `category_id`-Objektfeld übereinstimmt.

3. Erstellen Sie mithilfe der übereinstimmenden Objekte `image`, `annotation` und `category` eine JSON-Zeile für das Bild. Informationen zur Zuordnung der Felder finden Sie unter Zuordnen von COCO-Objektfeldern zu JSON-Zeilenobjektfeldern mit benutzerdefinierten Labels.
4. Wiederholen Sie die Schritte 1-3, bis Sie für jedes `image`-Objekt in der `images`-Liste JSON-Zeilen erstellt haben.

Beispielcode finden Sie unter Transformieren eines COCO-Datensatzes.

Zuordnen von COCO-Objektfeldern zu JSON-Zeilenobjektfeldern mit benutzerdefinierten Labels

Nachdem Sie die COCO-Objekte für eine Amazon Rekognition Custom Labels JSON-Zeile identifiziert haben, müssen Sie die COCO-Objektfelder den jeweiligen JSON-Zeilenobjektfeldern von Amazon Rekognition Custom Labels zuordnen. Die folgende JSON-Beispielzeile für Amazon Rekognition Custom Labels ordnet ein Bild (`id=000000245915`) dem vorherigen COCO-JSON-Beispiel zu. Notieren Sie die folgenden Informationen:

- `source-ref` ist der Speicherort des Images in einem Amazon-S3-Bucket. Wenn Ihre COCO-Bilder nicht in einem Amazon-S3-Bucket gespeichert sind, müssen Sie sie in einen Amazon-S3-Bucket verschieben.
- Die `annotations`-Liste enthält ein `annotation`-Objekt für jedes Objekt auf dem Bild. Ein `annotation`-Objekt enthält Informationen zum Begrenzungsrahmen (`top`,`left`,`width`,`height`) und eine Label-ID (`class_id`).
- Die Label-ID (`class_id`) ist der `class-map`-Liste in den Metadaten zugeordnet. Sie listet die auf dem Bild verwendeten Labels auf.

```
{
 "source-ref": "s3://custom-labels-bucket/images/000000245915.jpg",
 "bounding-box": {
  "image_size": {
   "width": 640,
   "height": 480,
   "depth": 3
  },
  "annotations": [{
   "class_id": 0,
   "top": 251,
   "left": 399,
   "width": 155,
```

```
    "height": 101
   }, {
    "class_id": 1,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
   }]
  },
  "bounding-box-metadata": {
   "objects": [{
    "confidence": 1
   }, {
    "confidence": 1
   }],
   "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2018-10-18T22:18:13.527256",
   "job-name": "my job"
  }
}
```

Verwenden Sie die folgenden Informationen, um die Felder der Amazon Rekognition Custom Labels-Manifestdatei den JSON-Feldern des COCO-Datensatzes zuzuordnen.

Quellennachweis

Die URL im S3-Format für den Speicherort des Bildes. Das Video muss in einem S3-Bucket gespeichert sein. Weitere Informationen finden Sie unter Quellennachweis. Wenn das `coco_url`-COCO-Feld auf eine S3-Bucket-Position verweist, können Sie den Wert von `coco_url` für den Wert von `source-ref` verwenden. Alternativ können Sie das Feld `source-ref` dem Feld `file_name` (COCO) zuordnen und in Ihrem Transformationscode den erforderlichen S3-Pfad zu dem Speicherort des Bilds hinzufügen.

Begrenzungsrahmen

Ein Label-Attributname Ihrer Wahl. Weitere Informationen finden Sie unter *Begrenzungsrahmen*.

Bildgröße

Die Größe des Bildes in MB Ordnet einem `image`-Objekt in der Bilder-Liste zu.

- `height` -> `image.height`
- `width` -> `image.width`
- `depth` -> Wird nicht von Amazon Rekognition Custom Labels verwendet, aber es muss ein Wert angegeben werden.

Anmerkungen

Eine Liste von `annotation`-Objekten. Für jedes Objekt auf dem Bild gibt es einen `annotation`.

Anmerkung

Enthält Begrenzungsrahmen für eine Instance eines Objekts auf dem Bild.

- `class_id` -> Zuordnung einer numerischen ID zur `class-map`-Liste von Custom Label.
- `top` -> `bbox[1]`
- `left` -> `bbox[0]`
- `width` -> `bbox[2]`
- `height` -> `bbox[3]`

***Begrenzungsrahmen*-Metadaten**

Metadaten für das Label-Attribut. Beinhaltet die Labels und Label-IDs. Weitere Informationen finden Sie unter *Begrenzungsrahmen*-Metadaten.

Objekte

Ein Array von Objekten im Bild. Ordnet der `annotations`-Liste nach Index zu.

Object

- `confidence` ->Wird nicht von Amazon Rekognition Custom Labels verwendet, aber ein Wert (1) ist erforderlich.

Klassenzuordnung

Eine Übersicht der Labels (Klassen), die für die im Bild erkannten Objekte gelten. Ordnet Kategorieobjekten in der Kategorien-Liste zu.

- `id` -> `category`.`id`
- `id value` -> `category`.`name`

Typ

Muss `groundtruth/object-detection` sein.

mit menschlichen Anmerkungen versehen

Geben Sie `yes` oder `no` an. Weitere Informationen finden Sie unter *`Begrenzungsrahmen`*-Metadaten.

Erstellungsdatum -> Bild.Aufnahmedatum

Das Erstellungsdatum und der Erstellungszeitpunkt des Bildes. Ordnet dem Feld Bild.Aufnahmedatum eines Bildes in der COCO-Bilderliste zu. Amazon Rekognition Custom Labels erwartet, dass das Format von `creation-date` Y-M-DTH:M:S lautet.

Jobname

Ein Jobname Ihrer Wahl.

COCO-Format

Ein COCO-Datensatz besteht aus fünf Informationsabschnitten, die Informationen für den gesamten Datensatz enthalten. Das Format für einen Datensatz zur COCO-Objekterkennung ist im COCO-Datenformat dokumentiert.

- Info – allgemeine Informationen über den Datensatz.
- licenses – Lizenzinformationen für die Bilder im Datensatz.
- images – eine Liste der Bilder im Datensatz.
- annotations – eine Liste von Anmerkungen (einschließlich Begrenzungsrahmen), die in allen Bildern im Datensatz vorhanden sind.

- categories – eine Liste von Label-Kategorien.

Um ein Manifest für benutzerdefinierte Labels zu erstellen, verwenden Sie die Listen `images`, `annotations` und `categories` aus der COCO-Manifestdatei. Die anderen Abschnitte (`info`,`licences`) sind nicht erforderlich. Folgendes ist ein Beispiel für den Inhalt einer COCO-Manifestdatei.

```
{
    "info": {
        "description": "COCO 2017 Dataset","url": "http://cocodataset.org","version":
 "1.0","year": 2017,"contributor": "COCO Consortium","date_created": "2017/09/01"
    },
    "licenses": [
        {"url": "http://creativecommons.org/licenses/by/2.0/","id": 4,"name":
 "Attribution License"}
    ],
    "images": [
        {"id": 242287, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/xxxxxxxxxxxx.jpg", "flickr_url": "http://farm3.staticflickr.com/2626/
xxxxxxxxxxxx.jpg", "width": 426, "height": 640, "file_name": "xxxxxxxxx.jpg",
 "date_captured": "2013-11-15 02:41:42"},
        {"id": 245915, "license": 4, "coco_url": "http://images.cocodataset.org/
val2017/nnnnnnnnnnnn.jpg", "flickr_url": "http://farm1.staticflickr.com/88/
xxxxxxxxxxxx.jpg", "width": 640, "height": 480, "file_name": "nnnnnnnnnn.jpg",
 "date_captured": "2013-11-18 02:53:27"}
    ],
    "annotations": [
        {"id": 125686, "category_id": 0, "iscrowd": 0, "segmentation": [[164.81,
 417.51,......167.55, 410.64]], "image_id": 242287, "area": 42061.80340000001, "bbox":
 [19.23, 383.18, 314.5, 244.46]},
        {"id": 1409619, "category_id": 0, "iscrowd": 0, "segmentation": [[376.81,
 238.8,........382.74, 241.17]], "image_id": 245915, "area": 3556.2197000000015,
 "bbox": [399, 251, 155, 101]},
        {"id": 1410165, "category_id": 1, "iscrowd": 0, "segmentation": [[486.34,
 239.01,..........495.95, 244.39]], "image_id": 245915, "area": 1775.8932499999994,
 "bbox": [86, 65, 220, 334]}
    ],
    "categories": [
        {"supercategory": "speaker","id": 0,"name": "echo"},
        {"supercategory": "speaker","id": 1,"name": "echo dot"}
    ]
}
```

Liste der Bilder

Die Bilder, auf die in einem COCO-Datensatz verwiesen wird, sind im Bilderarray aufgeführt. Jedes Bildobjekt enthält Informationen über das Bild, z. B. den Namen der Bilddatei. Notieren Sie sich im folgenden Beispiel-Bildobjekt die folgenden Informationen und welche Felder erforderlich sind, um eine Amazon Rekognition Custom Labels-Manifestdatei zu erstellen.

- `id` – (Erforderlich) Eine eindeutige Kennung für das Bild. Das `id`-Feld ist dem `id`-Feld im Annotationsarray zugeordnet (in dem Begrenzungsrahmen-Informationen gespeichert werden).
- `license` – (Nicht erforderlich) Entspricht dem Lizenz-Array.
- `coco_url` – (Optional) Den Speicherort des Bildes.
- `flickr_url` – (Nicht erforderlich) Der Speicherort des Bildes auf Flickr.
- `width` – (Erforderlich) Die Breite des Bildes.
- `height` – (Erforderlich) Die Höhe des Bildes.
- `file_name` – (Erforderlich) Der Name der Bilddatei. In diesem Beispiel stimmen `file_name` und `id` überein, aber das ist keine Voraussetzung für COCO-Datensätze.
- `date_captured` – (Erforderlich) Datum und Uhrzeit der Aufnahme des Bildes.

```
{
    "id": 245915,
    "license": 4,
    "coco_url": "http://images.cocodataset.org/val2017/nnnnnnnnnnnn.jpg",
    "flickr_url": "http://farm1.staticflickr.com/88/nnnnnnnnnnnnnnnnnn.jpg",
    "width": 640,
    "height": 480,
    "file_name": "000000245915.jpg",
    "date_captured": "2013-11-18 02:53:27"
}
```

Liste der Anmerkungen (Begrenzungsfelder)

Die Begrenzungsrahmeninformationen für alle Objekte auf allen Bildern werden in der Liste der Anmerkungen gespeichert. Ein einzelnes Anmerkungsobjekt enthält Begrenzungsrahmeninformationen für ein einzelnes Objekt und die Bezeichnung des Objekts auf einem Bild. Für jede Instance eines Objekts auf einem Bild gibt es ein Anmerkungsobjekt.

Notieren Sie sich im folgenden Beispiel die folgenden Informationen und welche Felder erforderlich sind, um eine Amazon Rekognition Custom Labels-Manifestdatei zu erstellen.

- `id` – (Nicht erforderlich) Die Kennung für die Anmerkung.
- `image_id` – (Erforderlich) Entspricht dem Bild `id` im Bildarray.
- `category_id` – (Erforderlich) Der Bezeichner für das Label, das das Objekt innerhalb eines Begrenzungsrahmens identifiziert. Er wird dem `id`-Feld des Kategorien-Arrays zugeordnet.
- `iscrowd` – (Nicht erforderlich) Gibt an, ob das Bild eine Menge von Objekten enthält.
- `segmentation` – (Nicht erforderlich) Segmentierungsinformationen für Objekte auf einem Bild. Amazon Rekognition Custom Labels unterstützt keine Segmentierung.
- `area` – (Nicht erforderlich) Der Bereich der Anmerkung.
- `bbox` – (Erforderlich) Enthält die Koordinaten eines Begrenzungsrahmens, der ein Objekt auf dem Bild umgibt, in Pixeln.

```
{
    "id": 1409619,
    "category_id": 1,
    "iscrowd": 0,
    "segmentation": [
        [86.0, 238.8,.........382.74, 241.17]
    ],
    "image_id": 245915,
    "area": 3556.2197000000015,
    "bbox": [86, 65, 220, 334]
}
```

Liste der Kategorien

Labelinformationen werden im Kategorien-Array gespeichert. Notieren Sie sich im folgenden Beispiel für ein Kategorie-Objekt die folgenden Informationen und welche Felder erforderlich sind, um eine Amazon Rekognition Custom Labels-Manifestdatei zu erstellen.

- `supercategory` – (Nicht erforderlich) Die übergeordnete Kategorie für ein Label.
- `id` – (Erforderlich) Die Label-ID. Das `id`-Feld ist dem `category_id`-Feld in einem `annotation`-Objekt zugeordnet. Im folgenden Beispiel ist der Identifier für einen Echo Dot 2.
- `name` – (Erforderlich) Der Labelname.

```
        {"supercategory": "speaker","id": 2,"name": "echo dot"}
```

Transformieren eines COCO-Datensatzes

Verwenden Sie das folgende Python-Beispiel, um Begrenzungsrahmen-Informationen aus einem Datensatz im COCO-Format in eine Amazon Rekognition Custom Labels-Manifestdatei umzuwandeln. Der Code lädt die erstellte Manifestdatei in Ihren Amazon-S3-Bucket hoch. Der Code stellt auch einen AWS CLI-Befehl bereit, mit dem Sie Ihre Bilder hochladen können.

So transformieren Sie einen COCO-Datensatz (SDK)

1. Wenn Sie dies noch nicht getan haben:

   a. Stellen Sie sicher, dass Sie die folgenden `AmazonS3FullAccess`-Berechtigungen haben. Weitere Informationen finden Sie unter Einrichten von SDK-Berechtigungen.

   b. Installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.

2. Verwenden Sie den folgenden Python-Code, um einen COCO-Datensatz zu transformieren. Legen Sie die folgenden Werte fest.

   - `s3_bucket` – Der Name des S3-Buckets, in dem Sie die Bilder und die Amazon Rekognition Custom Labels-Manifestdatei speichern möchten.
   - `s3_key_path_images` – Der Pfad zu dem Ort, an dem Sie die Bilder innerhalb des S3-Buckets platzieren möchten (`s3_bucket`).
   - `s3_key_path_manifest_file` – Der Pfad zu der Stelle, an der Sie die Manifestdatei für benutzerdefinierte Labels im S3-Bucket platzieren möchten (`s3_bucket`).
   - `local_path` – Der lokale Pfad, in dem das Beispiel den COCO-Eingabedatensatz öffnet und auch die neue Custom Labels-Manifestdatei speichert.
   - `local_images_path` – Der lokale Pfad zu den Bildern, die Sie für das Training verwenden möchten.
   - `coco_manifest` – Der Dateiname des COCO-Eingabedatensatzes.
   - `cl_manifest_file` – Ein Name für die Manifestdatei, die durch das Beispiel erstellt wurde. Die Datei wird an dem von `local_path` angegebenen Speicherort gespeichert. Konventionell hat die Datei die Erweiterung `.manifest`, dies ist jedoch nicht erforderlich.
   - `job_name` – Ein Name für den Custom Labels-Job.

```
import json
import os
```

```
import random
import shutil
import datetime
import botocore
import boto3
import PIL.Image as Image
import io

#S3 location for images
s3_bucket = 'bucket'
s3_key_path_manifest_file = 'path to custom labels manifest file/'
s3_key_path_images = 'path to images/'
s3_path='s3://' + s3_bucket  + '/' + s3_key_path_images
s3 = boto3.resource('s3')

#Local file information
local_path='path to input COCO dataset and output Custom Labels manifest/'
local_images_path='path to COCO images/'
coco_manifest = 'COCO dataset JSON file name'
coco_json_file = local_path + coco_manifest
job_name='Custom Labels job name'
cl_manifest_file = 'custom_labels.manifest'

label_attribute ='bounding-box'

open(local_path + cl_manifest_file, 'w').close()

# class representing a Custom Label JSON line for an image
class cl_json_line:
    def __init__(self,job, img):

        #Get image info. Annotations are dealt with seperately
        sizes=[]
        image_size={}
        image_size["width"] = img["width"]
        image_size["depth"] = 3
        image_size["height"] = img["height"]
        sizes.append(image_size)

        bounding_box={}
        bounding_box["annotations"] = []
        bounding_box["image_size"] = sizes

        self.__dict__["source-ref"] = s3_path + img['file_name']
```

```
        self.__dict__[job] = bounding_box

        #get metadata
        metadata = {}
        metadata['job-name'] = job_name
        metadata['class-map'] = {}
        metadata['human-annotated']='yes'
        metadata['objects'] = []
        date_time_obj = datetime.datetime.strptime(img['date_captured'], '%Y-%m-%d
 %H:%M:%S')
        metadata['creation-date']= date_time_obj.strftime('%Y-%m-%dT%H:%M:%S')
        metadata['type']='groundtruth/object-detection'

        self.__dict__[job + '-metadata'] = metadata


print("Getting image, annotations, and categories from COCO file...")

with open(coco_json_file) as f:

    #Get custom label compatible info
    js = json.load(f)
    images = js['images']
    categories = js['categories']
    annotations = js['annotations']

    print('Images: ' + str(len(images)))
    print('annotations: ' + str(len(annotations)))
    print('categories: ' + str(len (categories)))


print("Creating CL JSON lines...")

images_dict = {image['id']: cl_json_line(label_attribute, image) for image in
 images}

print('Parsing annotations...')
for annotation in annotations:

    image=images_dict[annotation['image_id']]

    cl_annotation = {}
    cl_class_map={}
```

```
    # get bounding box information
    cl_bounding_box={}
    cl_bounding_box['left'] = annotation['bbox'][0]
    cl_bounding_box['top'] = annotation['bbox'][1]

    cl_bounding_box['width'] = annotation['bbox'][2]
    cl_bounding_box['height'] = annotation['bbox'][3]
    cl_bounding_box['class_id'] = annotation['category_id']

    getattr(image, label_attribute)['annotations'].append(cl_bounding_box)


    for category in categories:
         if annotation['category_id'] == category['id']:
            getattr(image, label_attribute + '-metadata')['class-map']
[category['id']]=category['name']



    cl_object={}
    cl_object['confidence'] = int(1)  #not currently used by Custom Labels
    getattr(image, label_attribute + '-metadata')['objects'].append(cl_object)

print('Done parsing annotations')

# Create manifest file.
print('Writing Custom Labels manifest...')

for im in images_dict.values():

    with open(local_path+cl_manifest_file, 'a+') as outfile:
            json.dump(im.__dict__,outfile)
            outfile.write('\n')
            outfile.close()

# Upload manifest file to S3 bucket.
print ('Uploading Custom Labels manifest file to S3 bucket')
print('Uploading'  + local_path + cl_manifest_file + ' to ' +
 s3_key_path_manifest_file)
print(s3_bucket)
s3 = boto3.resource('s3')
s3.Bucket(s3_bucket).upload_file(local_path + cl_manifest_file,
 s3_key_path_manifest_file + cl_manifest_file)

# Print S3 URL to manifest file,
```

```
print ('S3 URL Path to manifest file. ')
print('\033[1m s3://' + s3_bucket + '/' + s3_key_path_manifest_file +
 cl_manifest_file + '\033[0m')

# Display aws s3 sync command.
print ('\nAWS CLI s3 sync command to upload your images to S3 bucket. ')
print ('\033[1m aws s3 sync ' + local_images_path + ' ' + s3_path + '\033[0m')
```

3. Führen Sie den Code aus.
4. Notieren Sie den `s3 sync`-Befehl in der Programmausgabe. Sie benötigen ihn im nächsten Schritt.
5. Führen Sie über die Eingabeaufforderung den folgenden `s3 sync`-Befehl aus. Ihre Bilder werden zu dem S3-Bucket hochgeladen. Wenn der Befehl beim Hochladen fehlschlägt, führen Sie ihn erneut aus, bis Ihre lokalen Bilder mit dem S3-Bucket synchronisiert sind.
6. Notieren Sie sich in der Programmausgabe den S3-URL-Pfad zur Manifestdatei. Sie benötigen ihn im nächsten Schritt.
7. Folgen Sie den Anweisungen unter Einen Datensatz mit einer SageMaker Ground Truth-Manifestdatei erstellen (Konsole), um einen Datensatz mit der hochgeladenen Manifestdatei zu erstellen. Geben Sie für Schritt 8 im Manifestdatei-Speicherort die Amazon S3-URL ein, die Sie sich im vorherigen Schritt notiert haben. Wenn Sie das AWS SDK verwenden, tun Sie diesEinen Datensatz mit einer SageMaker Ground Truth Manifest-Datei (SDK) erstellen.

Transformation von SageMaker Ground Truth-Manifestdateien mit mehreren Labels

In diesem Thema erfahren Sie, wie Sie eine Amazon SageMaker Ground Truth-Manifestdatei mit mehreren Labels in eine Manifestdatei im Amazon Rekognition Custom Labels-Format umwandeln.

SageMaker Ground Truth Truth-Manifestdateien für Aufträge mit mehreren Labels sind anders formatiert als Manifestdateien im Amazon Rekognition Custom Labels-Format. Bei der Klassifizierung mit mehreren Labels wird ein Bild in eine Gruppe von Klassen eingeteilt, kann aber gleichzeitig mehreren Klassen angehören. In diesem Fall kann das Bild möglicherweise mehrere Labels haben, z. B. Fußball und Ball.

Informationen zu SageMaker Ground-Truth-Aufträgen mit mehreren Labels finden Sie unter Bildklassifizierung (mehrere Labels). Informationen zu Amazon Rekognition Custom Labels-Manifestdateien im Format mit mehreren Labels finden Sie unter the section called “Hinzufügen mehrerer Labels auf Bildebene zu einem Bild”.

Die Manifestdatei für einen SageMaker Ground Truth Job abrufen

Das folgende Verfahren zeigt Ihnen, wie Sie die Ausgabe-Manifestdatei (`output.manifest`) für einen Amazon SageMaker Ground Truth Job abrufen. Sie verwenden `output.manifest` als Eingabe für das nächste Verfahren.

So laden Sie eine SageMaker Ground Truth Job-Manifest-Datei herunter

1. Öffnen Sie https://console.aws.amazon.com/sagemaker/.
2. Wählen Sie im Navigationsbereich Ground Truth und dann Labeling-Jobs aus.
3. Wählen Sie den Labeling-Job, der die Manifestdatei enthält, die Sie verwenden möchten.
4. Wählen Sie auf der Detailseite den Link unter Speicherort des Ausgabe-Datensatzes aus. Die Amazon S3-Konsole wird am Speicherort des Datensatzes geöffnet.
5. Wählen Sie `Manifests`, `output` und anschließend `output.manifest` aus.
6. Um die Manifestdatei herunterzuladen, wählen Sie Objektanmerkungen und dann Herunterladen aus.

Transformieren einer SageMaker Manifestdatei mit mehreren Bezeichnungen

Das folgende Verfahren erstellt eine Amazon Rekognition Custom Labels-Manifestdatei im Multi-Label-Format aus einer vorhandenen Manifestdatei im Multi-Label-Format. SageMaker GroundTruth

> **Note**
>
> Um den Code auszuführen, benötigen Sie Python Version 3 oder höher.

Um eine Manifestdatei mit mehreren Bezeichnungen zu transformieren SageMaker

1. Führen Sie den folgenden Python-Code aus. Geben Sie den Namen der Manifestdatei, die Sie in Die Manifestdatei für einen SageMaker Ground Truth Job abrufen als Befehlszeilenargument erstellt haben, an.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier:  Apache-2.0
"""
Purpose
Shows how to create and Amazon Rekognition Custom Labels format
manifest file from an Amazon SageMaker Ground Truth Image
```

```
Classification (Multi-label) format manifest file.
"""
import json
import logging
import argparse
import os.path

logger = logging.getLogger(__name__)

def create_manifest_file(ground_truth_manifest_file):
    """
    Creates an Amazon Rekognition Custom Labels format manifest file from
    an Amazon SageMaker Ground Truth Image Classification (Multi-label) format
    manifest file.
    :param: ground_truth_manifest_file: The name of the Ground Truth manifest file,
    including the relative path.
    :return: The name of the new Custom Labels manifest file.
    """

    logger.info('Creating manifest file from %s', ground_truth_manifest_file)
    new_manifest_file =
 f'custom_labels_{os.path.basename(ground_truth_manifest_file)}'

    # Read the SageMaker Ground Truth manifest file into memory.
    with open(ground_truth_manifest_file) as gt_file:
        lines = gt_file.readlines()

    #Iterate through the lines one at a time to generate the
    #new lines for the Custom Labels manifest file.
    with open(new_manifest_file, 'w') as the_new_file:
        for line in lines:
            #job_name - The of the Amazon Sagemaker Ground Truth job.
            job_name = ''
            # Load in the old json item from the Ground Truth manifest file
            old_json = json.loads(line)

            # Get the job name
            keys = old_json.keys()
            for key in keys:
                if 'source-ref' not in key and '-metadata' not in key:
                    job_name = key

            new_json = {}
            # Set the location of the image
```

```
            new_json['source-ref'] = old_json['source-ref']

            # Temporarily store the list of labels
            labels = old_json[job_name]

            # Iterate through the labels and reformat to Custom Labels format
            for index, label in enumerate(labels):
                new_json[f'{job_name}{index}'] = index
                metadata = {}
                metadata['class-name'] = old_json[f'{job_name}-metadata']['class-
map'][str(label)]
                metadata['confidence'] = old_json[f'{job_name}-metadata']
['confidence-map'][str(label)]
                metadata['type'] = 'groundtruth/image-classification'
                metadata['job-name'] = old_json[f'{job_name}-metadata']['job-name']
                metadata['human-annotated'] = old_json[f'{job_name}-metadata']
['human-annotated']
                metadata['creation-date'] = old_json[f'{job_name}-metadata']
['creation-date']
                # Add the metadata to new json line
                new_json[f'{job_name}{index}-metadata'] = metadata
            # Write the current line to the json file
            the_new_file.write(json.dumps(new_json))
            the_new_file.write('\n')

    logger.info('Created %s', new_manifest_file)
    return  new_manifest_file

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "manifest_file", help="The Amazon SageMaker Ground Truth manifest file"
        "that you want to use."
    )


def main():
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")
    try:
```

```
        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()
        # Create the manifest file
        manifest_file = create_manifest_file(args.manifest_file)
        print(f'Manifest file created: {manifest_file}')
    except FileNotFoundError as err:
        logger.exception('File not found: %s', err)
        print(f'File not found: {err}. Check your manifest file.')

if __name__ == "__main__":
    main()
```

2. Notieren Sie sich den Namen der neuen Manifestdatei, die das Skript anzeigt. Sie werden ihn im nächsten Schritt verwenden.
3. Laden Sie Ihre Manifestdateien in den Amazon-S3-Bucket hoch, den Sie zum Speichern der Manifestdatei verwenden möchten.

> **Note**
>
> Stellen Sie sicher, dass Amazon Rekognition Custom Labels Zugriff auf den Amazon-S3-Bucket hat, auf den im `source-ref`-Feld der JSON-Zeilen der Manifestdatei verwiesen wird. Weitere Informationen finden Sie unter Zugreifen auf externe Amazon-S3-Buckets. Wenn Ihr Ground Truth-Job Bilder im Amazon Rekognition Custom Labels-Konsolen-Bucket speichert, müssen Sie keine Berechtigungen hinzufügen.

4. Folgen Sie den Anweisungen unter Einen Datensatz mit einer SageMaker Ground Truth-Manifestdatei erstellen (Konsole), um einen Datensatz mit der hochgeladenen Manifestdatei zu erstellen. Geben Sie für Schritt 8 unter Speicherort der Manifestdatei die Amazon S3-URL für die Manifestdatei ein. Wenn Sie das AWS SDK verwenden, tun Sie dasEinen Datensatz mit einer SageMaker Ground Truth Manifest-Datei (SDK) erstellen.

Erstellen einer Manifestdatei aus einer CSV-Datei

Dieses Python-Beispielskript vereinfacht die Erstellung einer Manifestdatei, indem es eine CSV-Datei (Comma Separated Values) verwendet, um Bilder zu beschriften. So erstellen Sie die CSV-Datei Die Manifestdatei eignet sich für die Klassifizierung von Bildern mit mehreren Labels oder

Bildklassifizierung (mehrere Label). Weitere Informationen finden Sie unter Objekte, Szenen und Konzepte finden.

> **Note**
>
> Dieses Skript erstellt keine Manifestdatei, die für die Suche nach Objektpositionen oder Markenpositionen geeignet ist.

Eine Manifestdatei beschreibt die Bilder, die zum Trainieren eines Modells verwendet werden. Zum Beispiel Bildpositionen und Labels, die Bildern zugewiesen wurden. Eine Manifestdatei besteht aus einer oder mehreren JSON-Zeilen. Jede JSON-Zeile beschreibt ein einzelnes Bild. Weitere Informationen finden Sie unter the section called “Labels auf Bildebene in Manifestdateien”.

Eine CSV-Datei stellt tabellarische Daten über mehrere Zeilen in einer Textdatei dar. Felder in einer Zeile werden durch ein Komma getrennt. Weitere Informationen finden Sie unter Kommagetrennte Werte. Bei diesem Skript steht jede Zeile in Ihrer CSV-Datei für ein einzelnes Bild und ist einer JSON-Zeile in der Manifestdatei zugeordnet. Um eine CSV-Datei für eine Manifestdatei zu erstellen, die die Bildklassifizierung mit mehreren Labels unterstützt, fügen Sie jeder Zeile ein oder mehrere Labels auf Bildebene hinzu. Um eine geeignete Manifestdatei für Bildklassifizierung zu erstellen, fügen Sie jeder Zeile ein einzelnes Label auf Bildebene hinzu.

In der folgenden CSV-Datei werden beispielsweise die Bilder im Projekt Bildklassifizierung (mehrere Label) (Blumen) Erste Schritte beschrieben.

```
camellia1.jpg,camellia,with_leaves
camellia2.jpg,camellia,with_leaves
camellia3.jpg,camellia,without_leaves
helleborus1.jpg,helleborus,without_leaves,not_fully_grown
helleborus2.jpg,helleborus,with_leaves,fully_grown
helleborus3.jpg,helleborus,with_leaves,fully_grown
jonquil1.jpg,jonquil,with_leaves
jonquil2.jpg,jonquil,with_leaves
jonquil3.jpg,jonquil,with_leaves
jonquil4.jpg,jonquil,without_leaves
mauve_honey_myrtle1.jpg,mauve_honey_myrtle,without_leaves
mauve_honey_myrtle2.jpg,mauve_honey_myrtle,with_leaves
mauve_honey_myrtle3.jpg,mauve_honey_myrtle,with_leaves
mediterranean_spurge1.jpg,mediterranean_spurge,with_leaves
mediterranean_spurge2.jpg,mediterranean_spurge,without_leaves
```

Das Skript generiert JSON-Zeilen für jede Zeile. Das Folgende ist beispielsweise die JSON-Zeile für die erste Zeile (`camellia1.jpg,camellia,with_leaves`).

```
{"source-ref": "s3://bucket/flowers/train/camellia1.jpg","camellia": 1,"camellia-
metadata":{"confidence": 1,"job-name": "labeling-job/camellia","class-name":
 "camellia","human-annotated": "yes","creation-date": "2022-01-21T14:21:05","type":
 "groundtruth/image-classification"},"with_leaves": 1,"with_leaves-metadata":
{"confidence": 1,"job-name": "labeling-job/with_leaves","class-name":
 "with_leaves","human-annotated": "yes","creation-date": "2022-01-21T14:21:05","type":
 "groundtruth/image-classification"}}
```

In der Beispiel-CSV ist der Amazon S3-Pfad zum Bild nicht vorhanden. Wenn Ihre CSV-Datei den Amazon S3-Pfad für die Bilder nicht enthält, verwenden Sie das `--s3_path`-Befehlszeilenargument, um den Amazon S3-Pfad zu dem Bild anzugeben.

Das Skript zeichnet den ersten Eintrag für jedes Bild in einer deduplizierten Bild-CSV-Datei auf. Die CSV-Datei mit dedupliziertem Bild enthält eine einzelne Instance jedes Bildes, das in der CSV-Eingabedatei gefunden wurde. Weitere Vorkommen eines Bilds in der CSV-Eingabedatei werden in einer doppelten CSV-Bilddatei aufgezeichnet. Wenn das Skript doppelte Bilder findet, überprüfen Sie die CSV-Datei mit dem doppelten Bild und aktualisieren Sie die CSV-Datei mit dem deduplizierten Bild nach Bedarf. Führen Sie das Skript mit der deduplizierten Datei erneut aus. Wenn in der CSV-Eingabedatei keine Duplikate gefunden werden, löscht das Skript die deduplizierte CSV-Datei mit dem Bild und die CSV-Datei mit dem doppelten Bild, da sie leer sind.

In diesem Verfahren erstellen Sie die CSV-Datei und führen das Python-Skript aus, um die Manifestdatei zu erstellen.

So erstellen Sie eine Manifestdatei aus einer CSV-Datei

1. Erstellen Sie eine CSV-Datei mit den folgenden Feldern in jeder Zeile (eine Zeile pro Bild). Fügen Sie der CSV-Datei keine Kopfzeile hinzu.

Feld 1	Feld 2	Feld n
Der Bildname oder der Amazon S3-Pfad des Bildes. z. B. `s3://my-bucket/flowers/train/camellia1.jpg` . Sie	Das erste Label auf Bildebene für das Bild.	Eine oder mehrere zusätzlic he Labels auf Bildebene, getrennt durch Kommas. Fügen Sie es nur hinzu, wenn Sie eine Manifestd

Feld 1	Feld 2	Feld n
können keine Mischung aus Bildern mit dem Amazon S3-Pfad und Bildern ohne ihn verwenden.		atei erstellen möchten, die die Bildklassifizierung mit mehreren Labels unterstützt.

Zum Beispiel `camellia1.jpg,camellia,with_leaves` oder `s3://my-bucket/flowers/train/camellia1.jpg,camellia,with_leaves`

2. Speichern Sie die CSV-Datei.
3. Führen Sie das folgende Python-Skript aus. Stellen Sie die folgenden Argumente bereit:
   - `csv_file` – Die CSV-Datei, die Sie in Schritt 1 erstellt haben.
   - `manifest_file` – Der Name der Manifestdatei, die Sie erstellen möchten.
   - (Optional) `--s3_path` *`s3://path_to_folder/`* – Der Amazon S3-Pfad, der den Bilddateinamen hinzugefügt werden soll (Feld 1). Verwenden Sie `--s3_path`, wenn die Bilder in Feld 1 noch keinen S3-Pfad enthalten.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier:  Apache-2.0

from datetime import datetime, timezone
import argparse
import logging
import csv
import os
import json

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service documentation.
Shows how to create an image-level (classification) manifest file from a CSV file.
You can specify multiple image level labels per image.
CSV file format is
image,label,label,..
If necessary, use the bucket argument to specify the S3 bucket folder for the
 images.
```

```
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-gt-cl-
transform.html
"""

logger = logging.getLogger(__name__)


def check_duplicates(csv_file, deduplicated_file, duplicates_file):
    """
    Checks for duplicate images in a CSV file. If duplicate images
    are found, deduplicated_file is the deduplicated CSV file - only the first
    occurence of a duplicate is recorded. Other duplicates are recorded in
 duplicates_file.
    :param csv_file: The source CSV file.
    :param deduplicated_file: The deduplicated CSV file to create. If no duplicates
 are found
    this file is removed.
    :param duplicates_file: The duplicate images CSV file to create. If no
 duplicates are found
    this file is removed.
    :return: True if duplicates are found, otherwise false.
    """

    logger.info("Deduplicating %s", csv_file)

    duplicates_found = False

    # Find duplicates.
    with open(csv_file, 'r', newline='', encoding="UTF-8") as f,\
            open(deduplicated_file, 'w', encoding="UTF-8") as dedup,\
            open(duplicates_file, 'w', encoding="UTF-8") as duplicates:

        reader = csv.reader(f, delimiter=',')
        dedup_writer = csv.writer(dedup)
        duplicates_writer = csv.writer(duplicates)

        entries = set()
        for row in reader:
            # Skip empty lines.
            if not ''.join(row).strip():
                continue

            key = row[0]
            if key not in entries:
```

```
                dedup_writer.writerow(row)
                entries.add(key)
            else:
                duplicates_writer.writerow(row)
                duplicates_found = True

    if duplicates_found:
        logger.info("Duplicates found check %s", duplicates_file)

    else:
        os.remove(duplicates_file)
        os.remove(deduplicated_file)

    return duplicates_found


def create_manifest_file(csv_file, manifest_file, s3_path):
    """
    Reads a CSV file and creates a Custom Labels classification manifest file.
    :param csv_file: The source CSV file.
    :param manifest_file: The name of the manifest file to create.
    :param s3_path: The S3 path to the folder that contains the images.
    """
    logger.info("Processing CSV file %s", csv_file)

    image_count = 0
    label_count = 0

    with open(csv_file, newline='', encoding="UTF-8") as csvfile,\
            open(manifest_file, "w", encoding="UTF-8") as output_file:

        image_classifications = csv.reader(
            csvfile, delimiter=',', quotechar='|')

        # Process each row (image) in CSV file.
        for row in image_classifications:
            source_ref = str(s3_path)+row[0]

            image_count += 1

            # Create JSON for image source ref.
            json_line = {}
            json_line['source-ref'] = source_ref
```

```
            # Process each image level label.
            for index in range(1, len(row)):
                image_level_label = row[index]

                # Skip empty columns.
                if image_level_label == '':
                    continue
                label_count += 1

               # Create the JSON line metadata.
                json_line[image_level_label] = 1
                metadata = {}
                metadata['confidence'] = 1
                metadata['job-name'] = 'labeling-job/' + image_level_label
                metadata['class-name'] = image_level_label
                metadata['human-annotated'] = "yes"
                metadata['creation-date'] = \
                    datetime.now(timezone.utc).strftime('%Y-%m-%dT%H:%M:%S.%f')
                metadata['type'] = "groundtruth/image-classification"

                json_line[f'{image_level_label}-metadata'] = metadata

                # Write the image JSON Line.
            output_file.write(json.dumps(json_line))
            output_file.write('\n')

    output_file.close()
    logger.info("Finished creating manifest file %s\nImages: %s\nLabels: %s",
                manifest_file, image_count, label_count)

    return image_count, label_count


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "csv_file", help="The CSV file that you want to process."
    )

    parser.add_argument(
```

```
        "--s3_path", help="The S3 bucket and folder path for the images."
        " If not supplied, column 1 is assumed to include the S3 path.",
 required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        s3_path = args.s3_path
        if s3_path is None:
            s3_path = ''

        # Create file names.
        csv_file = args.csv_file
        file_name = os.path.splitext(csv_file)[0]
        manifest_file = f'{file_name}.manifest'
        duplicates_file = f'{file_name}-duplicates.csv'
        deduplicated_file = f'{file_name}-deduplicated.csv'

        # Create manifest file, if there are no duplicate images.
        if check_duplicates(csv_file, deduplicated_file, duplicates_file):
            print(f"Duplicates found. Use {duplicates_file} to view duplicates "
                  f"and then update {deduplicated_file}. ")
            print(f"{deduplicated_file} contains the first occurence of a
 duplicate. "
                  "Update as necessary with the correct label information.")
            print(f"Re-run the script with {deduplicated_file}")
        else:
            print("No duplicates found. Creating manifest file.")

            image_count, label_count = create_manifest_file(csv_file,
                                                            manifest_file,
                                                            s3_path)
```

```
            print(f"Finished creating manifest file: {manifest_file} \n"
                  f"Images: {image_count}\nLabels: {label_count}")

    except FileNotFoundError as err:
        logger.exception("File not found: %s", err)
        print(f"File not found: {err}. Check your input CSV file.")


if __name__ == "__main__":
    main()
```

4. Wenn Sie einen Testdatensatz verwenden möchten, wiederholen Sie die Schritte 1-3, um eine Manifestdatei für Ihren Testdatensatz zu erstellen.
5. Kopieren Sie die Bilder bei Bedarf in den Amazon-S3-Bucket-Pfad, den Sie in Spalte 1 der CSV-Datei (oder in der `--s3_path`-Befehlszeile) angegeben haben. Sie können den folgenden AWS S3-Befehl verwenden.

```
aws s3 cp --recursive your-local-folder s3://your-target-S3-location
```

6. Laden Sie Ihre Manifestdateien in den Amazon-S3-Bucket hoch, den Sie zum Speichern der Manifestdatei verwenden möchten.

> **Note**
>
> Stellen Sie sicher, dass Amazon Rekognition Custom Labels Zugriff auf den Amazon-S3-Bucket hat, auf den im `source-ref`-Feld der JSON-Zeilen der Manifestdatei verwiesen wird. Weitere Informationen finden Sie unter Zugreifen auf externe Amazon-S3-Buckets. Wenn Ihr Ground Truth-Job Bilder im Amazon Rekognition Custom Labels-Konsolen-Bucket speichert, müssen Sie keine Berechtigungen hinzufügen.

7. Folgen Sie den Anweisungen unter Einen Datensatz mit einer SageMaker Ground Truth-Manifestdatei erstellen (Konsole), um einen Datensatz mit der hochgeladenen Manifestdatei zu erstellen. Geben Sie für Schritt 8 unter Speicherort der Manifestdatei die Amazon S3-URL für die Manifestdatei ein. Wenn Sie das AWS SDK verwenden, tun Sie diesEinen Datensatz mit einer SageMaker Ground Truth Manifest-Datei (SDK) erstellen.

Bestehender Datensatz

Wenn Sie zuvor einen Datensatz erstellt haben, können Sie seinen Inhalt in einen neuen Datensatz kopieren. Informationen zum Erstellen eines Datensatzes aus einem vorhandenen Datensatz mit dem AWS SDK finden Sie unterErstellen eines Datensatzes unter Verwendung eines vorhandenen Datensatzes (SDK).

So erstellen Sie einen Datensatz mit einem bestehenden Amazon Rekognition Custom Labels-Datensatz (Konsole)

1. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Benutzerdefinierte Labels verwenden.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie auf der Seite Projekte das Projekt aus, dem Sie den Datensatz hinzufügen möchten. Die Detailseite für Ihr Projekt wird angezeigt.
6. Wählen Sie Datensatz erstellen. Die Seite Datensatz erstellen wird angezeigt.
7. Wählen Sie in der Startkonfiguration entweder Mit einem einzelnen Datensatz beginnen oder Mit einem Trainingsdatensatz beginnen aus. Um ein qualitativ hochwertigeres Modell zu erstellen, empfehlen wir, mit separaten Trainings- und Testdatensätzen zu beginnen.

   Single dataset

   a. Wählen Sie im Abschnitt Details zum Trainingsdatensatz die Option Einen vorhandenen Amazon Rekognition Custom Labels-Datensatz kopieren aus.
   b. Geben Sie im Abschnitt Details zum Trainingsdatensatz im Bearbeitungsfeld Datensatz den Namen des Datensatzes ein, den Sie kopieren möchten, oder wählen Sie ihn aus.
   c. Wählen Sie Datensatz erstellen. Die Datensatzseite für Ihr Projekt wird geöffnet.

   Separate training and test datasets

   a. Wählen Sie im Abschnitt Details zum Trainingsdatensatz die Option Einen vorhandenen Amazon Rekognition Custom Labels-Datensatz kopieren aus.
   b. Geben Sie im Abschnitt Details zum Trainingsdatensatz im Bearbeitungsfeld Datensatz den Namen des Datensatzes ein, den Sie kopieren möchten, oder wählen Sie ihn aus.

c. Wählen Sie im Abschnitt Details zum Testdatensatz die Option Einen vorhandenen Amazon Rekognition Custom Labels-Datensatz kopieren aus.

d. Geben Sie im Abschnitt Testdatensatzdetails im Bearbeitungsfeld Datensatz den Namen des Datensatzes ein, den Sie kopieren möchten, oder wählen Sie ihn aus.

> **Note**
>
> Ihre Trainings- und Testdatensätze können unterschiedliche Bildquellen haben.

e. Wählen Sie Datensätze erstellen aus. Die Datensatzseite für Ihr Projekt wird geöffnet.

8. Wenn Sie Labels hinzufügen oder ändern müssen, führen Sie Labeling von Bildern aus.
9. Folgen Sie den Anweisungen unter Ein Model trainieren (Konsole), um Ihr Modell zu trainieren.

Labeling von Bildern

Ein Label identifiziert ein Objekt, eine Szene, ein Konzept oder einen Begrenzungsrahmen, der ein Objekt in einem Bild umgibt. Wenn Ihr Datensatz beispielsweise Bilder von Hunden enthält, können Sie Labels für Hunderassen hinzufügen.

Nachdem Sie Ihre Bilder in einen Datensatz importiert haben, müssen Sie den Bildern möglicherweise Labels hinzufügen oder falsch beschriftete Bilder korrigieren. Bilder werden beispielsweise nicht mit Labels versehen, wenn sie von einem lokalen Computer importiert werden. Sie verwenden die Datensatz-Galerie, um dem Datensatz neue Labels hinzuzufügen und Bildern im Datensatz Labels und Begrenzungsrahmen zuzuweisen.

Wie Sie die Bilder in Ihren Datensätzen mit Labels versehen, bestimmt den Modelltyp, den Amazon Rekognition Custom Labels trainiert. Weitere Informationen finden Sie unter Datensätzen einen Zweck geben.

Themen

- Labels verwalten
- Einem Bild Labels auf Bildebene zuweisen
- Objekte mit Begrenzungsrahmen mit Labels versehen

Labels verwalten

Sie können Labels mithilfe der Amazon Rekognition Custom Labels-Konsole verwalten. Es gibt keine spezielle API für die Verwaltung von Labels – Labels werden dem Datensatz hinzugefügt, wenn Sie den Datensatz mit `CreateDataset` erstellen oder wenn Sie dem Datensatz mit `UpdateDatasetEntries` weitere Bilder hinzufügen.

Themen

Labels verwalten (Konsole)

Sie können die Amazon Rekognition Custom Labels-Konsole verwenden, um Labels zu einem Datensatz hinzuzufügen, zu ändern oder zu entfernen. Um einem Datensatz ein Label hinzuzufügen, können Sie ein neues Label hinzufügen, das Sie erstellen, oder Labels aus einem vorhandenen Datensatz in Rekognition importieren.

Themen

Neue Labels hinzufügen (Konsole)

Sie können neue Labels angeben, die Sie Ihrem Datensatz hinzufügen möchten.

Fügen Sie Labels mithilfe des Bearbeitungsfensters hinzu

So fügen Sie ein neues Label hinzu (Konsole)

1. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Benutzerdefinierte Labels verwenden.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie auf der Seite Projekte das Projekt aus, das Sie verwenden möchten. Die Detailseite für Ihr Projekt wird angezeigt.

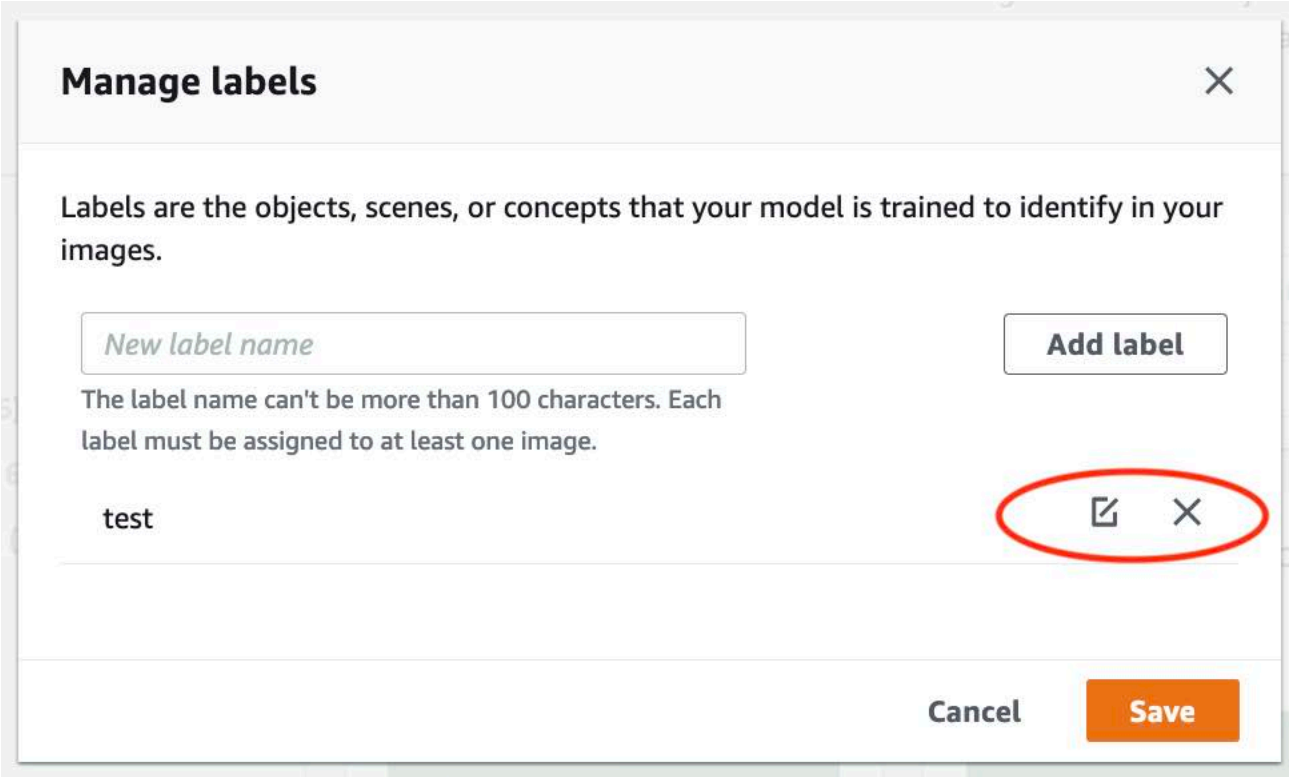
6. Wenn Sie Ihrem Trainingsdatensatz Labels hinzufügen möchten, wählen Sie den Tab Training. Wählen Sie andernfalls den Tab Test, um dem Testdatensatz Labels hinzuzufügen.
7. Wählen Sie Labeling starten, um in den Labeling-Modus zu wechseln.
8. Wählen Sie in der Datensatz-Galerie im Bereich Labels die Option Labels verwalten aus, um das Dialogfeld Labels verwalten zu öffnen.
9. Geben Sie im Bearbeitungsfeld einen neuen Labelnamen ein.
10. Klicken Sie auf Label hinzufügen.
11. Wiederholen Sie die Schritte 9 und 10, bis Sie alle benötigten Labels erstellt haben.
12. Wählen Sie Speichern, um die hinzugefügten Labels zu speichern.

Labels ändern und entfernen (Konsole)

Sie können Labels umbenennen oder entfernen, nachdem Sie sie einem Datensatz hinzugefügt haben. Sie können nur Labels entfernen, die keinem Bild zugewiesen sind.

So benennen Sie ein vorhandenes Label um oder entfernen es (Konsole)

1. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Benutzerdefinierte Labels verwenden.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie auf der Seite Projekte das Projekt aus, das Sie verwenden möchten. Die Detailseite für Ihr Projekt wird angezeigt.
6. Wenn Sie Labels in Ihrem Trainingsdatensatz ändern oder löschen möchten, wählen Sie den Tab Training. Wählen Sie andernfalls den Tab Test, um Labels für den Testdatensatz zu ändern oder zu löschen.
7. Wählen Sie Labeling starten, um in den Labeling-Modus zu wechseln.
8. Wählen Sie in der Datensatz-Galerie im Bereich Labels die Option Labels verwalten aus, um das Dialogfeld Labels verwalten zu öffnen.
9. Wählen Sie das Label, das Sie bearbeiten oder löschen möchten.



a. Wenn Sie das Symbol „Löschen“ (X) wählen, wird das Label aus der Liste entfernt.

b. Wenn Sie das Label ändern möchten, wählen Sie das Bearbeitungssymbol (Stift und Notizblock) und geben Sie einen neuen Labelnamen in das Bearbeitungsfeld ein.

10. Wählen Sie Speichern, um Ihre Änderungen zu speichern.

Labels verwalten (SDK)

Es gibt keine einzigartige API, die Datensatz-Labels verwaltet. Wenn Sie einen Datensatz mit `CreateDataset` erstellen (den Labels aus der Manifestdatei oder dem kopierten Datensatz), erstellen Sie den ersten Satz von Labels. Wenn Sie mit der `UpdateDatasetEntries`-API weitere Bilder hinzufügen, werden neue Labels, die in den Einträgen gefunden wurden, dem Datensatz hinzugefügt. Weitere Informationen finden Sie unter Weitere Bilder hinzufügen (SDK). Um Labels aus einem Datensatz zu löschen, müssen Sie alle Labelanmerkungen im Datensatz entfernen.

So löschen Sie Labels aus einem Datensatz

1. Rufen Sie `ListDatasetEntries` auf, um die Datensatzeinträge abzurufen. Beispielcode finden Sie unter Datensatzeinträge auflisten (SDK).

2. Entfernen Sie in der Datei alle Labelanmerkungen. Weitere Informationen finden Sie unter Labels auf Bildebene in Manifestdateien und the section called “Objektlokalisierung in Manifestdateien”.
3. Verwenden Sie die Datei, um den Datensatz mit der `UpdateDatasetEntries`-API zu aktualisieren. Weitere Informationen finden Sie unter Weitere Bilder hinzufügen (SDK).

Einem Bild Labels auf Bildebene zuweisen

Sie verwenden Labels auf Bildebene, um Modelle zu trainieren, die Bilder in Kategorien einteilen. Ein Label auf Bildebene gibt an, dass ein Bild ein Objekt, eine Szene oder ein Konzept enthält. Beispielsweise zeigt das folgende Bild einen Fluss. Wenn Ihr Modell Bilder so klassifiziert, dass sie Flüsse enthalten, würden Sie das Label Fluss auf Bildebene hinzufügen. Weitere Informationen finden Sie unter Datensätzen einen Zweck geben.

Für einen Datensatz, der Labels auf Bildebene enthält, müssen mindestens zwei Labels definiert werden. Jedem Bild muss mindestens ein Label zugewiesen werden, das das Objekt, die Szene oder das Konzept im Bild identifiziert.

So weisen Sie einem Bild Labels auf Bildebene zu (Konsole)

1. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Benutzerdefinierte Labels verwenden.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie auf der Seite Projekte das Projekt aus, das Sie verwenden möchten. Die Detailseite für Ihr Projekt wird angezeigt.
6. Wählen Sie im linken Navigationsbereich Dataset aus.
7. Wenn Sie Ihrem Trainingsdatensatz Labels hinzufügen möchten, wählen Sie den Tab Training. Wählen Sie andernfalls den Tab Test, um dem Testdatensatz Labels hinzuzufügen.
8. Wählen Sie Labeling starten, um in den Labeling-Modus zu wechseln.
9. Wählen Sie in der Bildergalerie ein oder mehrere Bilder aus, denen Sie Labels hinzufügen möchten. Sie können nur Bilder auf einer jeweils einer Seite auswählen. So wählen Sie einen zusammenhängenden Bereich von Bildern auf einer Seite aus:
    a. Wählen Sie das erste Bild des Bereichs aus.
    b. Halten Sie die Umschalttaste gedrückt.
    c. Wählen Sie den letzten Bildbereich aus. Die Bilder zwischen dem ersten und dem zweiten Bild werden ebenfalls ausgewählt.
    d. Lassen Sie die Umschalttaste los.
10. Wählen Sie Labels auf Bildebene zuweisen.
11. Wählen Sie im Dialogfeld „Markierten Bildern eine Bezeichnung auf Bildebene zuweisen“ eine Bezeichnung aus, die Sie dem Bild oder den Bildern zuweisen möchten.
12. Wählen Sie Zuweisen, um dem Bild ein Label zuzuweisen.
13. Wiederholen Sie das Hinzufügen von Labels, bis jedes Bild mit den erforderlichen Labels versehen ist.
14. Wählen Sie Änderungen speichern aus, um Ihre Änderungen zu speichern.

Zuweisen von Labels auf Bildebene (SDK)

Sie können die `UpdateDatasetEntries`-API verwenden, um die Labels auf Bildebene, die einem Bild zugewiesen sind, hinzuzufügen oder zu aktualisieren. `UpdateDatasetEntries` benötigt eine

oder mehrere JSON-Zeilen. Jede JSON-Zeile steht für ein einzelnes Bild. Bei einem Bild mit einem Label auf Bildebene sieht die JSON-Zeile wie folgt aus.


```
{"source-ref":"s3://custom-labels-console-us-east-1-nnnnnnnnnn/gt-job/
manifest/IMG_1133.png","TestCLConsoleBucket":0,"TestCLConsoleBucket-metadata":
{"confidence":0.95,"job-name":"labeling-job/testclconsolebucket","class-name":"Echo
 Dot","human-annotated":"yes","creation-
date":"2020-04-15T20:17:23.433061","type":"groundtruth/image-classification"}}
```


Das `source-ref`-Feld gibt die Position des Bildes an. Die JSON-Zeile enthält auch die dem Bild zugewiesenen Labels auf Bildebene. Weitere Informationen finden Sie unter the section called “Labels auf Bildebene in Manifestdateien”.

So weisen Sie einem Bild Labels auf Bildebene zu

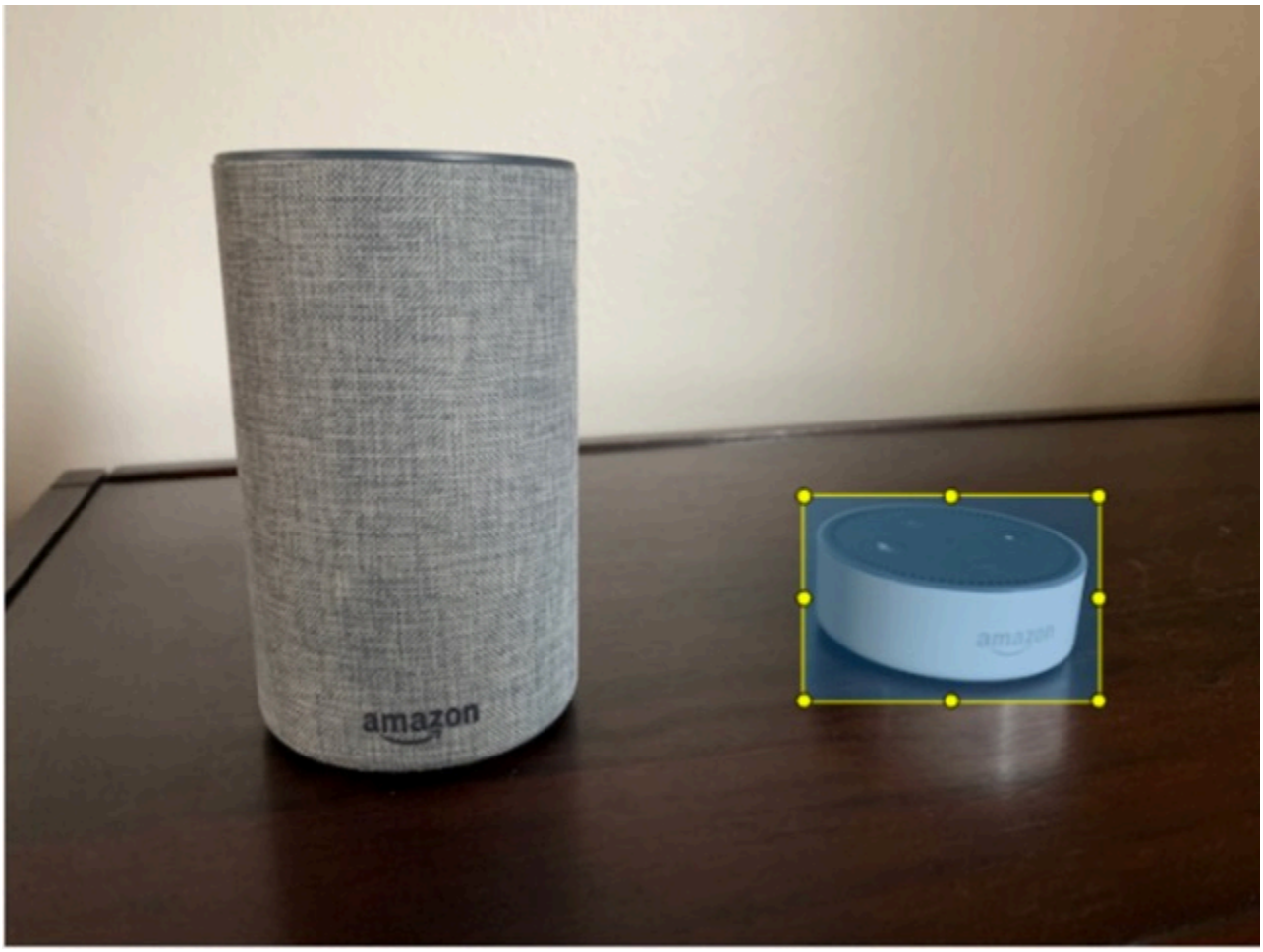
1. Rufen Sie die Get JSON-Zeile für das vorhandene Bild ab, indem Sie den `ListDatasetEntries` verwenden. Geben Sie für das `source-ref`-Feld die Position des Bildes an, dem Sie das Label zuweisen möchten. Weitere Informationen finden Sie unter Datensatzeinträge auflisten (SDK).
2. Aktualisieren Sie die im vorherigen Schritt zurückgegebene JSON-Zeile anhand der Informationen unter Labels auf Bildebene in Manifestdateien.
3. Rufen Sie `UpdateDatasetEntries` auf, um das Bild zu aktualisieren. Weitere Informationen finden Sie unter Hinzufügen weiterer Bilder zu einem Datensatz.

Objekte mit Begrenzungsrahmen mit Labels versehen

Wenn Sie möchten, dass Ihr Modell die Position von Objekten in einem Bild erkennt, müssen Sie herausfinden, um welches Objekt es sich handelt und wo es sich im Bild befindet. Ein Begrenzungsrahmen ist ein Rahmen, der ein Objekt in einem Bild isoliert. Sie verwenden Begrenzungsrahmen, um ein Modell so zu trainieren, dass es verschiedene Objekte in demselben Bild erkennt. Sie identifizieren das Objekt, indem Sie dem Begrenzungsrahmen ein Label zuweisen.

> **Note**
>
> Wenn Sie ein Modell darauf trainieren, Objekte, Szenen und Konzepte mit Labels auf Bildebene zu finden, müssen Sie diesen Schritt nicht ausführen.

Wenn Sie beispielsweise ein Modell trainieren möchten, das Amazon Echo Dot-Geräte erkennt, zeichnen Sie einen Begrenzungsrahmen um jeden Echo Dot in einem Bild und weisen dem Begrenzungsrahmen ein Label namens Echo Dot zu. Das folgende Bild zeigt einen Begrenzungsrahmen um ein Echo Dot-Gerät. Das Bild enthält auch ein Amazon Echo ohne Begrenzungsrahmen.



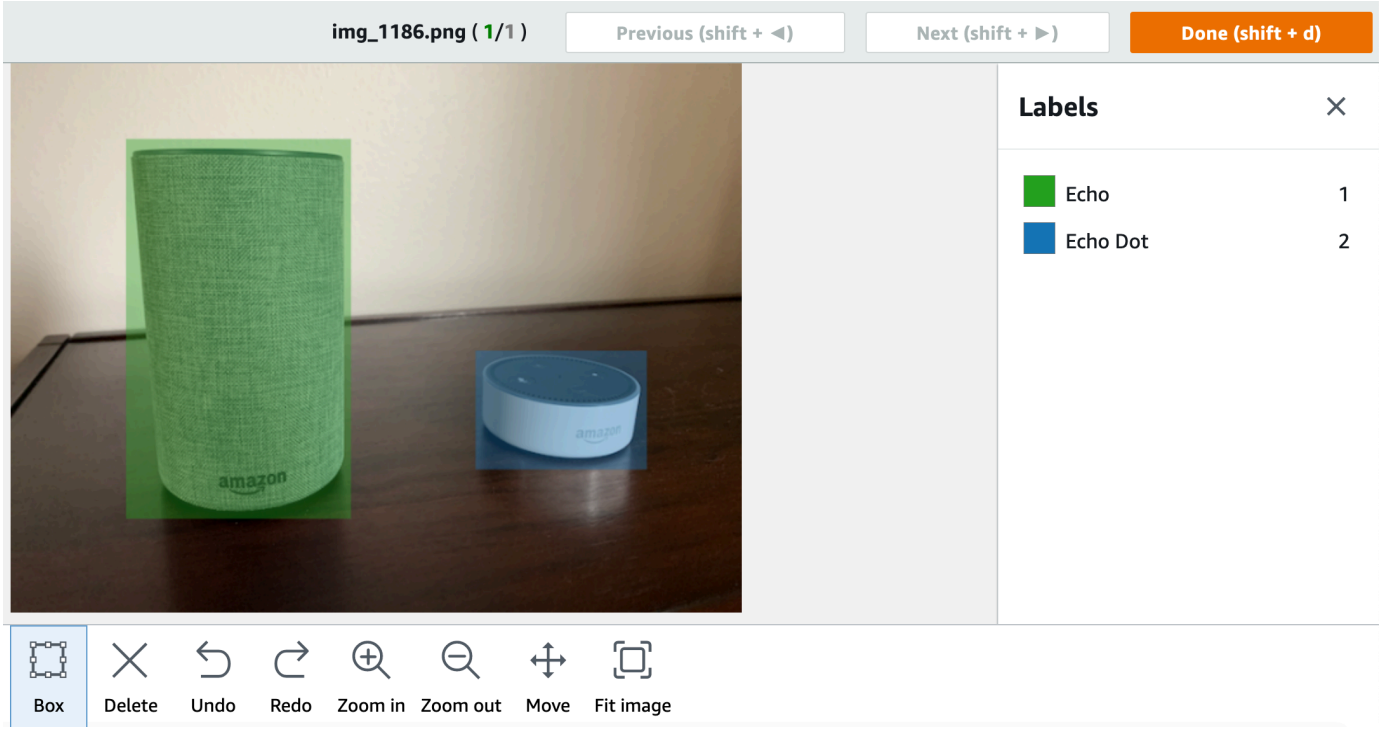
Objekte mit Begrenzungsrahmen finden (Konsole)

In diesem Verfahren verwenden Sie die Konsole, um Begrenzungsrahmen um die Objekte in Ihren Bildern zu zeichnen. Sie können Objekte im Bild auch identifizieren, indem Sie dem Begrenzungsrahmen Labels zuweisen.

> **Note**
>
> Sie können den Safari-Browser nicht verwenden, um Bildern Begrenzungsrahmen hinzuzufügen. Informationen zu unterstützten Browsern finden Sie unter Einrichten von Amazon Rekognition Custom Labels.

Bevor Sie Begrenzungsrahmen hinzufügen können, müssen Sie dem Datensatz mindestens ein Label hinzufügen. Weitere Informationen finden Sie unter Neue Labels hinzufügen (Konsole).

1. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Benutzerdefinierte Labels verwenden.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie auf der Seite Projekte das Projekt aus, das Sie verwenden möchten. Die Detailseite für Ihr Projekt wird angezeigt.
6. Wählen Sie auf der Seite mit den Projektdetails die Option Bilder mit Labels versehen
7. Wenn Sie den Bildern Ihres Trainingsdatensatzes Begrenzungsrahmen hinzufügen möchten, wählen Sie den Tab Training. Wählen Sie andernfalls den Tab Test, um den Bildern des Testdatensatzes Begrenzungsrahmen hinzuzufügen.
8. Wählen Sie Labeling starten, um in den Labeling-Modus zu wechseln.
9. Wählen Sie in der Bildergalerie die Bilder aus, denen Sie Begrenzungsrahmen hinzufügen möchten.
10. Wählen Sie Begrenzungsrahmen zeichnen. Bevor der Begrenzungsrahmen-Editor geöffnet wird, werden eine Reihe von Tipps angezeigt.
11. Wählen Sie im Bereich Labels auf der rechten Seite das Label aus, das Sie einem Begrenzungsrahmen zuweisen möchten.
12. Platzieren Sie den Mauszeiger im Zeichenwerkzeug auf dem oberen linken Bereich des gewünschten Objekts.
13. Drücken Sie die linke Maustaste und zeichnen Sie einen Rahmen um das Objekt. Versuchen Sie, den Begrenzungsrahmen so nahe an das Objekt wie möglich zu zeichnen.
14. Lassen Sie die Maustaste los. Der Begrenzungsrahmen ist hervorgehoben.
15. Wählen Sie Weiter, wenn Sie weitere Bilder mit Labels versehen möchten. Wählen Sie andernfalls Fertig, um das Labeling abzuschließen.



16. Wiederholen Sie die Schritte 1-7, bis Sie in jedem Bild, das Objekte enthält, einen Begrenzungsrahmen erstellt haben.
17. Wählen Sie Änderungen speichern aus, um Ihre Änderungen zu speichern.
18. Wählen Sie Beenden, um den Labeling-Modus zu verlassen.

Suchen Sie nach Objekten mit Begrenzungsrahmen (SDK)

Sie können die `UpdateDatasetEntries`-API verwenden, um Informationen für die Objektposition für ein Bild hinzuzufügen oder zu aktualisieren. `UpdateDatasetEntries` benötigt eine oder mehrere JSON-Zeilen. Jede JSON-Zeile steht für ein einzelnes Bild. Für die Objektlokalisierung sieht eine JSON-Zeile etwa folgendermaßen aus.

```
{"source-ref": "s3://bucket/images/IMG_1186.png", "bounding-box": {"image_size":
 [{"width": 640, "height": 480, "depth": 3}], "annotations": [{ "class_id": 1,
 "top": 251, "left": 399, "width": 155, "height": 101}, {"class_id": 0, "top": 65,
 "left": 86, "width": 220, "height": 334}]}, "bounding-box-metadata": {"objects":
 [{ "confidence": 1}, {"confidence": 1}], "class-map": {"0": "Echo", "1": "Echo Dot"},
 "type": "groundtruth/object-detection", "human-annotated": "yes", "creation-date":
 "2013-11-18T02:53:27", "job-name": "my job"}}
```

Das `source-ref`-Feld gibt die Position des Bildes an. Die JSON-Zeile enthält auch beschriftete Begrenzungsrahmen für jedes Objekt auf dem Bild. Weitere Informationen finden Sie unter the section called “Objektlokalisierung in Manifestdateien”.

So weisen Sie einem Bild Begrenzungsrahmen zu

1. Rufen Sie die Get JSON-Zeile für das vorhandene Bild ab, indem Sie den `ListDatasetEntries` verwenden. Geben Sie für das `source-ref`-Feld den Speicherort des Bildes an, dem Sie das Label auf Bildebene zuweisen möchten. Weitere Informationen finden Sie unter Datensatzeinträge auflisten (SDK).
2. Aktualisieren Sie die im vorherigen Schritt zurückgegebene JSON-Zeile anhand der Informationen unter Objektlokalisierung in Manifestdateien.
3. Rufen Sie `UpdateDatasetEntries` auf, um das Bild zu aktualisieren. Weitere Informationen finden Sie unter Hinzufügen weiterer Bilder zu einem Datensatz.

Debuggen von Datensätzen

Bei der Erstellung von Datensätzen können zwei Arten von Fehlern auftreten: endgültige Fehler und nicht endgültige Fehler. Endgültige Fehler können die Erstellung oder Aktualisierung von Datensätzen verhindern. Nicht endgültige Fehler verhindern die Erstellung oder Aktualisierung von Datensätzen nicht.

Themen

- Endgültige Fehler
- Nicht endgültige Fehler

Endgültige Fehler

Es gibt zwei Arten von endgültigen Fehlern: Dateifehler, die dazu führen, dass die Datensatzerstellung fehlschlägt, und Inhaltsfehler, die Amazon Rekognition Custom Labels aus dem Datensatz entfernt. Die Datensatzerstellung schlägt fehl, wenn zu viele Inhaltsfehler vorliegen.

Themen

- Endgültige Dateifehler
- Endgültige Inhaltsfehler

Endgültige Dateifehler

Die folgenden sind Dateifehler. Sie können Informationen zu Dateifehlern erhalten, indem Sie `DescribeDataset` aufrufen und die Felder `Status` und `StatusMessage` überprüfen. Beispielcode finden Sie unter Beschreibung eines Datensatzes (SDK).

- ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT
- ERROR_MANIFEST_SIZE_TOO_LARGE.
- ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM
- ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET
- ERROR_TOO_MANY_RECORDS_IN_ERROR
- ERROR_MANIFEST_TOO_MANY_LABELS
- ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT

Fehlermeldung

Die Erweiterung oder der Inhalt der Manifestdatei sind ungültig.

Die Trainings- oder Testmanifestdatei hat keine Dateierweiterung oder ihr Inhalt ist ungültig.

So beheben Sie den Fehler ERROR_MANIFEST_INACCESSIBLE_OR_UNSUPPORTED_FORMAT

- Überprüfen Sie die folgenden möglichen Ursachen sowohl in den Trainings- als auch in den Testmanifestdateien.
  - Der Manifestdatei fehlt eine Dateierweiterung. Üblicherweise lautet die Dateierweiterung `.manifest`.
  - Der Amazon-S3-Bucket oder der Schlüssel für die Manifestdatei konnte nicht gefunden werden.

ERROR_MANIFEST_SIZE_TOO_LARGE

Fehlermeldung

Die Manifestdatei überschreitet die maximal unterstützte Größe.

Die Größe der Datei für das Trainings- oder Testmanifest (in Byte) ist zu groß. Weitere Informationen finden Sie unter Richtlinien und Kontingente in Amazon Rekognition Custom Labels. Eine Manifestdatei kann weniger als die maximale Anzahl von JSON-Zeilen haben und trotzdem die maximale Dateigröße überschreiten.

Sie können die Amazon Rekognition Custom Labels-Konsole nicht verwenden, um den Fehler Die Größe der Manifestdatei überschreitet die maximal unterstützte Größe zu beheben.

So beheben Sie den Fehler ERROR_MANIFEST_SIZE_TOO_LARGE

1. Prüfen Sie, welche der Trainings- und Testmanifeste die maximale Dateigröße überschreiten.
2. Reduzieren Sie die Anzahl der zu großen JSON-Zeilen in den Manifestdateien. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM

Fehlermeldung

Die Manifestdatei hat zu viele Zeilen.

Weitere Informationen

Die Anzahl der JSON-Zeilen (Anzahl der Bilder) in der Manifestdatei ist größer als das zulässige Limit. Das Limit ist für Modelle auf Bildebene und für Modelle zur Objektlokalisierung unterschiedlich. Weitere Informationen finden Sie unter Richtlinien und Kontingente in Amazon Rekognition Custom Labels.

JSON-Zeilenfehler werden validiert, bis die Anzahl der JSON-Zeilen das `ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM`-Limit erreicht.

Sie können die Amazon Rekognition Custom Labels-Konsole nicht verwenden, um `ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM` zu beheben.

So beheben Sie **`ERROR_MANIFEST_ROWS_EXCEEDS_MAXIMUM`**

- Reduzieren Sie die Anzahl der JSON-Zeilen im Manifest. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET

Fehlermeldung

Falsche Berechtigungen für den S3-Bucket.

Amazon Rekognition Custom Labels hat keine Berechtigungen für einen oder mehrere Buckets, die die Trainings- und Testmanifestdateien enthalten.

Sie können die Amazon Rekognition Custom Labels-Konsole nicht verwenden, um diesen Fehler zu beheben.

So beheben Sie den Fehler ERROR_INVALID_PERMISSIONS_MANIFEST_S3_BUCKET

- Überprüfen Sie die Berechtigungen für die Buckets, die die Trainings- und Testmanifeste enthalten. Weitere Informationen finden Sie unter Schritt 2: Einrichten von Amazon Rekognition Custom Labels-Konsolenberechtigungen.

ERROR_TOO_MANY_RECORDS_IN_ERROR

Fehlermeldung

Die Manifestdatei enthält zu viele endgültige Fehler.

So beheben Sie **`ERROR_TOO_MANY_RECORDS_IN_ERROR`**

- Reduzieren Sie die Anzahl der JSON-Zeilen (Bilder) mit endgültigen Inhaltsfehlern. Weitere Informationen finden Sie unter Terminal-Manifest-Inhaltsfehler.

Sie können die Amazon Rekognition Custom Labels-Konsole nicht verwenden, um diesen Fehler zu beheben.

ERROR_MANIFEST_TOO_MANY_LABELS

Fehlermeldung

Die Manifestdatei hat zu viele Labels.

Weitere Informationen

Die Anzahl der eindeutigen Labels im Manifest (Datensatz) übersteigt den zulässigen Grenzwert. Wenn der Trainingsdatensatz aufgeteilt wird, um einen Testdatensatz zu erstellen, wird die Anzahl der Labels nach dem Teilen bestimmt.

So beheben Sie den Fehler ERROR_MANIFEST_TOO_MANY_LABELS (Konsole)

- Entfernen Sie Labels aus dem Datensatz. Weitere Informationen finden Sie unter Labels verwalten. Die Labels werden automatisch aus den Bildern und Begrenzungsrahmen in Ihrem Datensatz entfernt.

So beheben Sie den Fehler ERROR_MANIFEST_TOO_MANY_LABELS (JSON-Zeile)

- Manifeste mit JSON-Linien auf Bildebene – Wenn das Bild ein einziges Label hat, entfernen Sie die JSON-Zeilen für Bilder, die das gewünschte Label verwenden. Wenn die JSON-Zeile mehrere Labels enthält, entfernen Sie nur das JSON-Objekt für das gewünschte Label. Weitere Informationen finden Sie unter Hinzufügen mehrerer Labels auf Bildebene zu einem Bild.

  Manifeste mit JSON-Linien mit Objektposition – Entfernen Sie den Begrenzungsrahmen und die zugehörigen Labelinformationen für das Label, das Sie entfernen möchten. Tun Sie dies für jede JSON-Zeile, die das gewünschte Label enthält. Sie müssen das Label aus dem `class-map`-Array und den entsprechenden Objekten im Array `annotations` und `objects` entfernen. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien.

ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_DISTRIBUTE

Fehlermeldung

Die Manifestdatei enthält nicht genügend Bilder mit Labels, um den Datensatz zu verteilen.

Die Datensatzverteilung erfolgt, wenn Amazon Rekognition Custom Labels einen Trainingsdatensatz aufteilt, um einen Testdatensatz zu erstellen. Sie können einen Datensatz auch aufteilen, indem Sie die `DistributeDatasetEntries`-API aufrufen.

So beheben Sie den Fehler ERROR_MANIFEST_TOO_MANY_LABELS

- Fügen Sie dem Trainingsdatensatz weitere Bilder mit Labels hinzu

Endgültige Inhaltsfehler

Im Folgenden sind endgültige Inhaltsfehler aufgeführt. Bei der Datensatzerstellung werden Bilder mit endgültigen Inhaltsfehlern aus dem Datensatz entfernt. Der Datensatz kann weiterhin für Trainings verwendet werden. Wenn es zu viele Inhaltsfehler gibt, schlägt der Datensatz/die Aktualisierung fehl. Endgültige Inhaltsfehler im Zusammenhang mit Datensatzoperationen werden nicht in der Konsole angezeigt oder von `DescribeDataset` oder von einer anderen API zurückgegeben. Wenn Sie feststellen, dass Bilder oder Anmerkungen in Ihren Datensätzen fehlen, überprüfen Sie Ihre Datensatz-Manifestdateien auf die folgenden Probleme:

- Die Länge einer JSON-Zeile ist zu lang. Die maximale Länge beträgt 100 000 Zeichen.
- Der `source-ref`-Wert fehlt in einer JSON-Zeile.
- Das Format eines `source-ref`-Werts in einer JSON-Zeile ist ungültig.
- Der Inhalt einer JSON-Zeile ist nicht gültig.
- Der Wert eines `source-ref`-Felds wird mehr als einmal angezeigt. Auf ein Bild kann nur einmal in einem Datensatz verwiesen werden.

Informationen zu dem Feld `source-ref` finden Sie unter Erstellen einer Manifestdatei.

Nicht endgültige Fehler

Bei den folgenden Fehlern handelt es sich um nicht endgültige Fehler, die bei der Erstellung oder Aktualisierung von Datensätzen auftreten können. Diese Fehler können eine gesamte JSON-Zeile oder Anmerkungen innerhalb einer JSON-Zeile ungültig machen. Wenn eine JSON-Zeile einen Fehler aufweist, wird sie nicht für das Training verwendet. Wenn eine Anmerkung innerhalb einer JSON-Zeile einen Fehler aufweist, wird die JSON-Zeile weiterhin für das Training verwendet, jedoch ohne die defekte Anmerkung. Weitere Informationen zu JSON-Zeilen finden Sie unter Erstellen einer Manifestdatei.

Sie können über die Konsole und durch Aufrufen der `ListDatasetEntries`-API auf nicht endgültige Fehler zugreifen. Weitere Informationen finden Sie unter Datensatzeinträge auflisten (SDK).

Die folgenden Fehler werden auch während des Trainings zurückgegeben. Es wird empfohlen, diese Fehler zu beheben, bevor Sie Ihr Modell trainieren. Weitere Informationen finden Sie unter Fehler bei der Überprüfung von JSON-Zeilen ohne Terminal.

- FEHLER: NO_LABEL_ATTRIBUTES

- ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT
- ERROR_INVALID_LABEL_ATTRIBUTE_METADAT_FORMAT
- ERROR_NO_VALID_LABEL_ATTRIBUTES
- ERROR_INVALID_BOUNDING_BOX
- FEHLER_UNGÜLTIGE_BILDDIMENSION
- ERROR_BOUNDING_BOX_ZU KLEIN
- ERROR_NO_VALID_ANNOTATIONS
- ERROR_MISSING_BOUNDING_BOX_CONFIDENCE
- FEHLER_FEHLENDER_KLASSENKARTE_ID
- FEHLER ZU VIELE_BEGRENZTEN_BOXEN
- FEHLER_UNUNTERSTÜTZTED_ANWENDUNGSFALLTYP
- ERROR_INVALID_LABEL_NAME_LENGTH

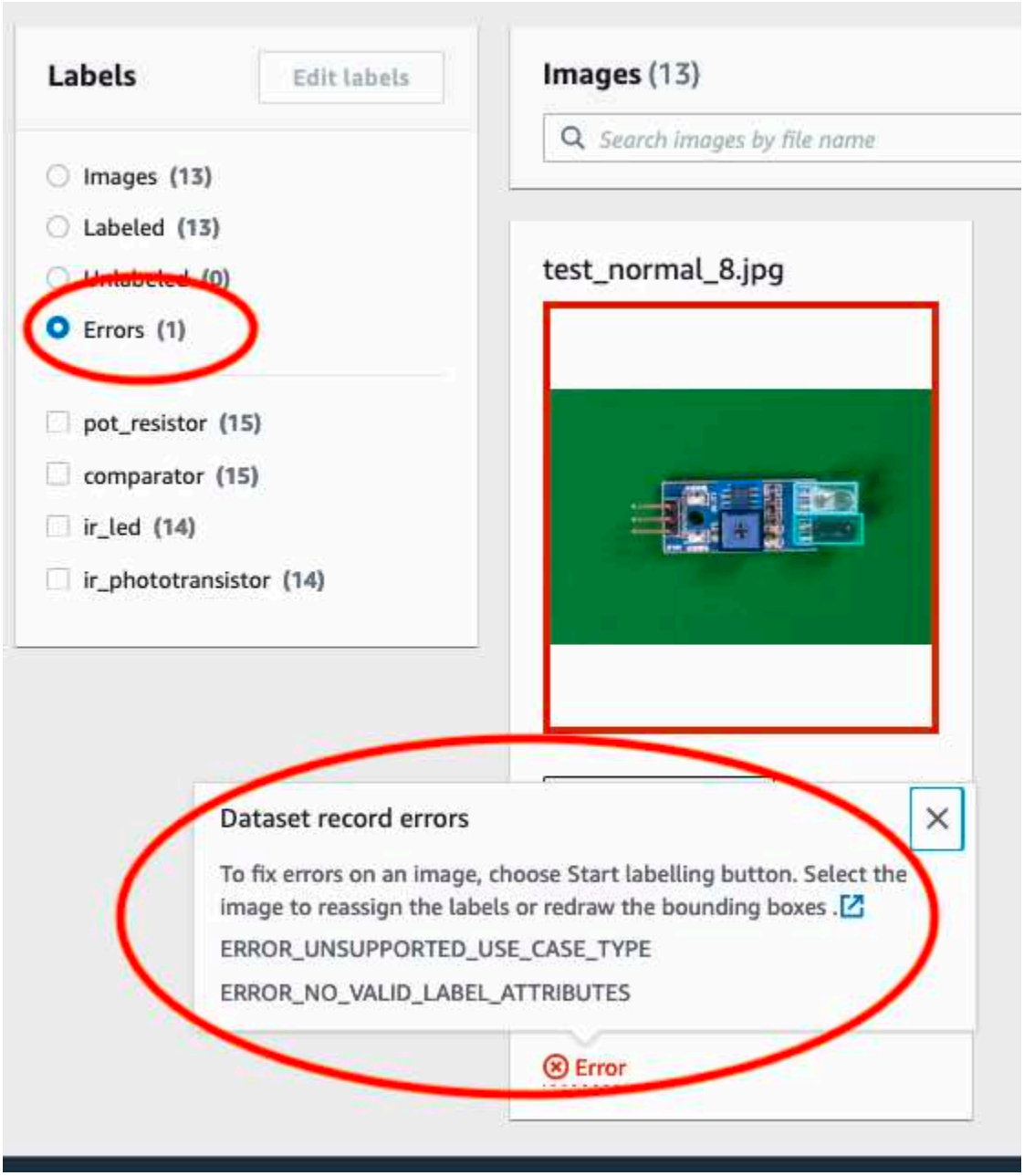
Zugreifen auf nicht endgültige Fehler

Mithilfe der Konsole können Sie herausfinden, bei welchen Bildern in einem Datensatz Fehler auftreten, die nicht endgültig sind. Sie können die `ListDatasetEntries`-API auch aufrufen oder aufrufen, um die Fehlermeldungen abzurufen. Weitere Informationen finden Sie unter Datensatzeinträge auflisten (SDK).

So greifen Sie auf nicht endgültige Fehler zu (Konsole)

1. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Benutzerdefinierte Labels verwenden.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie auf der Seite Projekte das Projekt aus, das Sie verwenden möchten. Die Detailseite für Ihr Projekt wird angezeigt.
6. Wenn Sie nicht endgültige Fehler in Ihrem Trainingsdatensatz anzeigen möchten, wählen Sie den Tab Training. Wählen Sie andernfalls den Tab Test, um sich nicht endgültige Fehler in Ihrem Testdatensatz anzusehen.
7. Wählen Sie in der Datensatz-Galerie im Bereich Labels die Option Fehler aus. Die Datensatz-Galerie wird so gefiltert, dass nur Bilder mit Fehlern angezeigt werden.

8. Wählen Sie unter einem Bild die Option Fehler aus, um den Fehlercode anzuzeigen. Verwenden Sie die Informationen unter Fehler bei der Überprüfung von JSON-Zeilen ohne Terminal, um den Fehler zu beheben.



Schulung eines Amazon-Rekognition-Custom-Labels-Modells

Sie können ein Modell mithilfe der Amazon Rekognition Custom Labels-Konsole oder der Amazon Rekognition Custom Labels API trainieren. Schlägt das Modelltraining fehl, verwenden Sie die

Informationen inDebuggen eines fehlgeschlagenen Modelltrainings, um die Ursache des Fehlers zu ermitteln.

> **Note**
>
> Ihnen wird die Zeit in Rechnung gestellt, die benötigt wird, um ein Modell erfolgreich zu trainieren. In der Regel dauert das Training 30 Minuten bis 24 Stunden. Weitere Informationen finden Sie unter Trainingszeiten.

Jedes Mal, wenn das Modell trainiert wird, wird eine neue Version eines Modells erstellt. Amazon Rekognition Custom Labels erstellt einen Namen für das Modell, der eine Kombination aus dem Projektnamen und dem Zeitstempel für die Erstellung des Modells ist.

Um Ihr Modell zu trainieren, erstellt Amazon Rekognition Custom Labels eine Kopie Ihrer ursprünglichen Trainings- und Testbilder. Standardmäßig werden die kopierten Bilder im Ruhezustand mit einem Schlüssel verschlüsselt, der AWS besitzt und verwaltet. Sie können auch Ihre eigenen verwendenAWS KMS key. Wenn Sie Ihren eigenen KMS-Schlüssel verwenden, benötigen Sie die folgenden Berechtigungen für den KMS-Schlüssel.

- km:CreateGrant
- km:DescribeKey

Weitere Informationen finden Sie im unter AWS Key Management Service Service-Konzepte. Ihre Quellbilder sind davon nicht betroffen.

Sie können die serverseitige KMS-Verschlüsselung (SSE-KMS) verwenden, um die Trainings- und Testbilder in Ihrem Amazon S3 S3-Bucket zu verschlüsseln, bevor sie von Amazon Rekognition Custom Labels kopiert werden. Um Amazon Rekognition Custom Labels Zugriff auf Ihre Bilder zu gewähren, benötigt IhrAWS Konto die folgenden Berechtigungen für den KMS-Schlüssel.

- km:GenerateDataKey
- kms:Decrypt

Weitere Informationen finden Sie im unter Schutz von Daten mit serverseitiger Verschlüsselung mit in AWS Key Management Service (SSE-KMS) gespeicherten KMS-Schlüsseln

Nach dem Training eines Modells können Sie dessen Leistung bewerten und Verbesserungen vornehmen. Weitere Informationen finden Sie unter Verbessern eines geschulten Amazon Rekognition Custom Labels-Modells.

Weitere Modellaufgaben, wie das Taggen eines Modells, finden Sie unterVerwaltung eines Amazon-Rekognition-Custom-Labels-Modells.

Themen

Ein Model trainieren (Konsole)

Sie können die Amazon Rekognition Custom Labels-Konsole verwenden, um ein Modell zu trainieren.

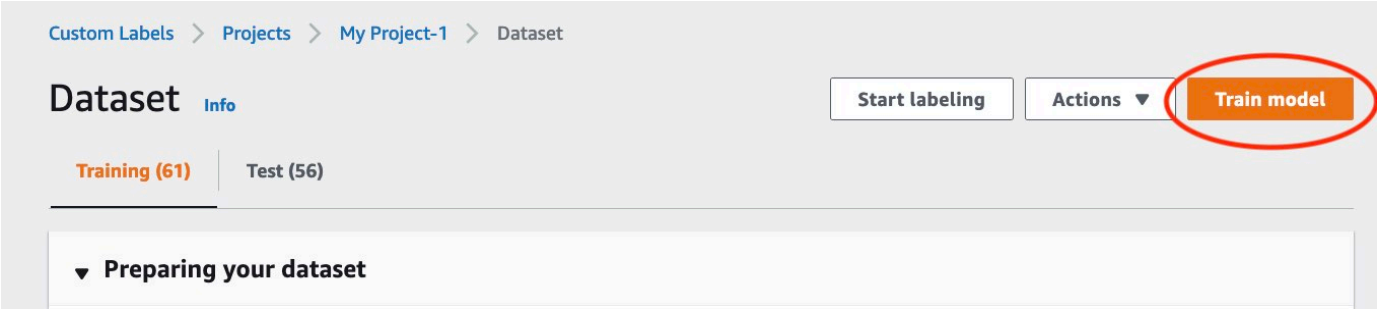
Das Training erfordert ein Projekt mit einem Trainingsdatensatz und einem Testdatensatz. Wenn Ihr Projekt keinen Testdatensatz hat, teilt die Amazon Rekognition Custom Labels-Konsole den Trainingsdatensatz während des Trainings auf, um einen für Ihr Projekt zu erstellen. Die ausgewählten Bilder sind eine repräsentative Stichprobe und werden nicht im Trainingsdatensatz verwendet. Wir empfehlen, Ihren Trainingsdatensatz nur zu teilen, wenn Sie keinen alternativen Testdatensatz haben, den Sie verwenden können. Durch das Aufteilen eines Trainingsdatensatzes wird die Anzahl der für das Training verfügbaren Bilder reduziert.

> **Note**
>
> Ihnen wird die Zeit in Rechnung gestellt, die für das Trainieren eines Modells benötigt wird. Weitere Informationen finden Sie unter Trainingszeiten.

Um dein Modell zu trainieren (Konsole)

1. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie „Benutzerdefinierte Labels verwenden“.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
4. Wählen Sie auf der Seite Projekte das Projekt aus, das das Modell enthält, das Sie trainieren möchten.

5. Wählen Sie auf der Projektseite Zugmodell aus.



6. (Optional) Wenn Sie Ihren eigenen AWS KMS KMS-Verschlüsselungsschlüssel verwenden möchten, gehen Sie wie folgt vor:

   a. Wählen Sie unter Bilddatenverschlüsselung die Option Verschlüsselungseinstellungen anpassen (erweitert).

   b. Geben Sie unter encryption.aws_kms_key den Amazon Resource Name (ARN) Ihres Schlüssels ein, oder wählen Sie einen vorhandenen AWS KMS KMS-Schlüssel aus. Um einen neuen Schlüssel zu erstellen, wählen Sie Create an AWS IMS-Schlüssel.

7. (Optional) gehen Sie wie folgt vor, wenn Sie Ihrem Modell Tags hinzufügen möchten:

   a. Wählen Sie im Abschnitt Tags die Option Neues Tag hinzufügen aus.

   b. Geben Sie Folgendes ein:

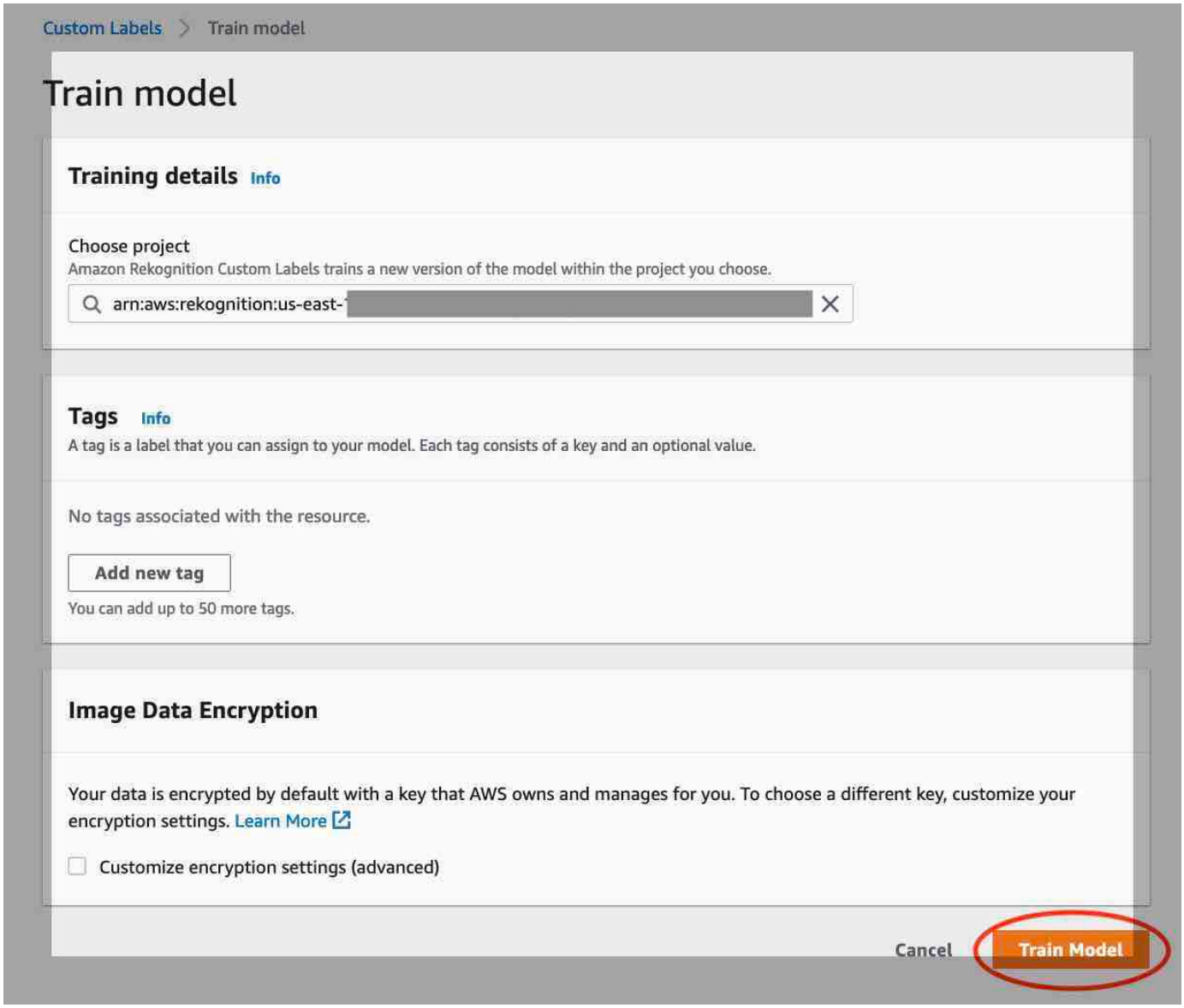
      i. Der Name des Schlüssels in Key.

      ii. Der Wert des Schlüssels in Value.

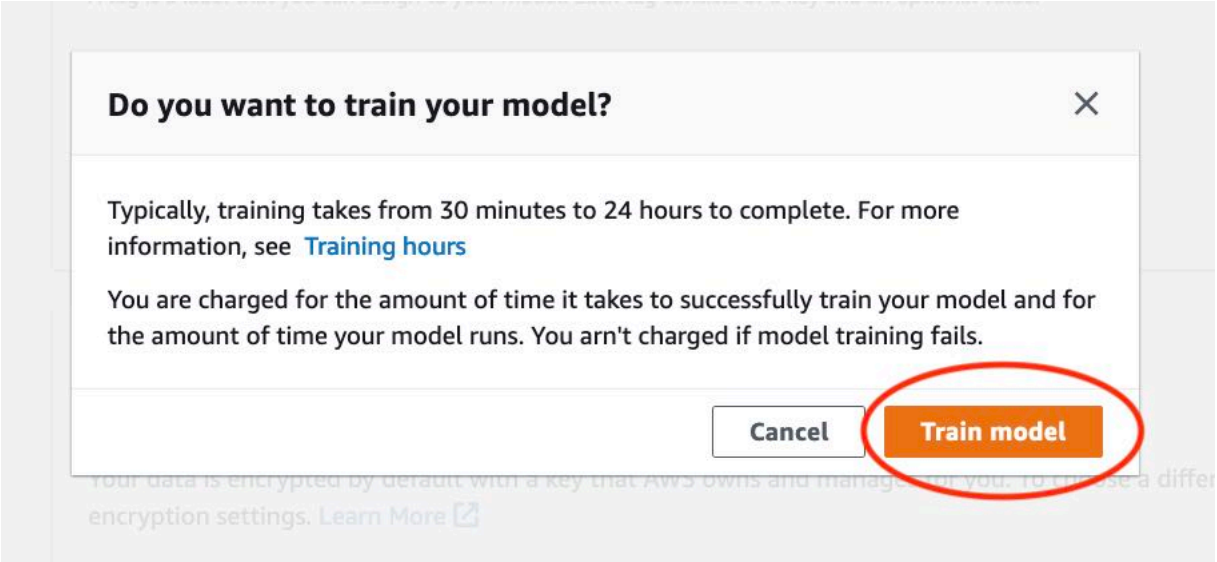
   c. Um weitere Tags hinzuzufügen, wiederholen Sie die Schritte 6a und 6b.

   d. (Optional) neben dem Tag, das Sie entfernen möchten. Wenn Sie ein zuvor gespeichertes Tag entfernen, wird es entfernt, wenn Sie Ihre Änderungen speichern.

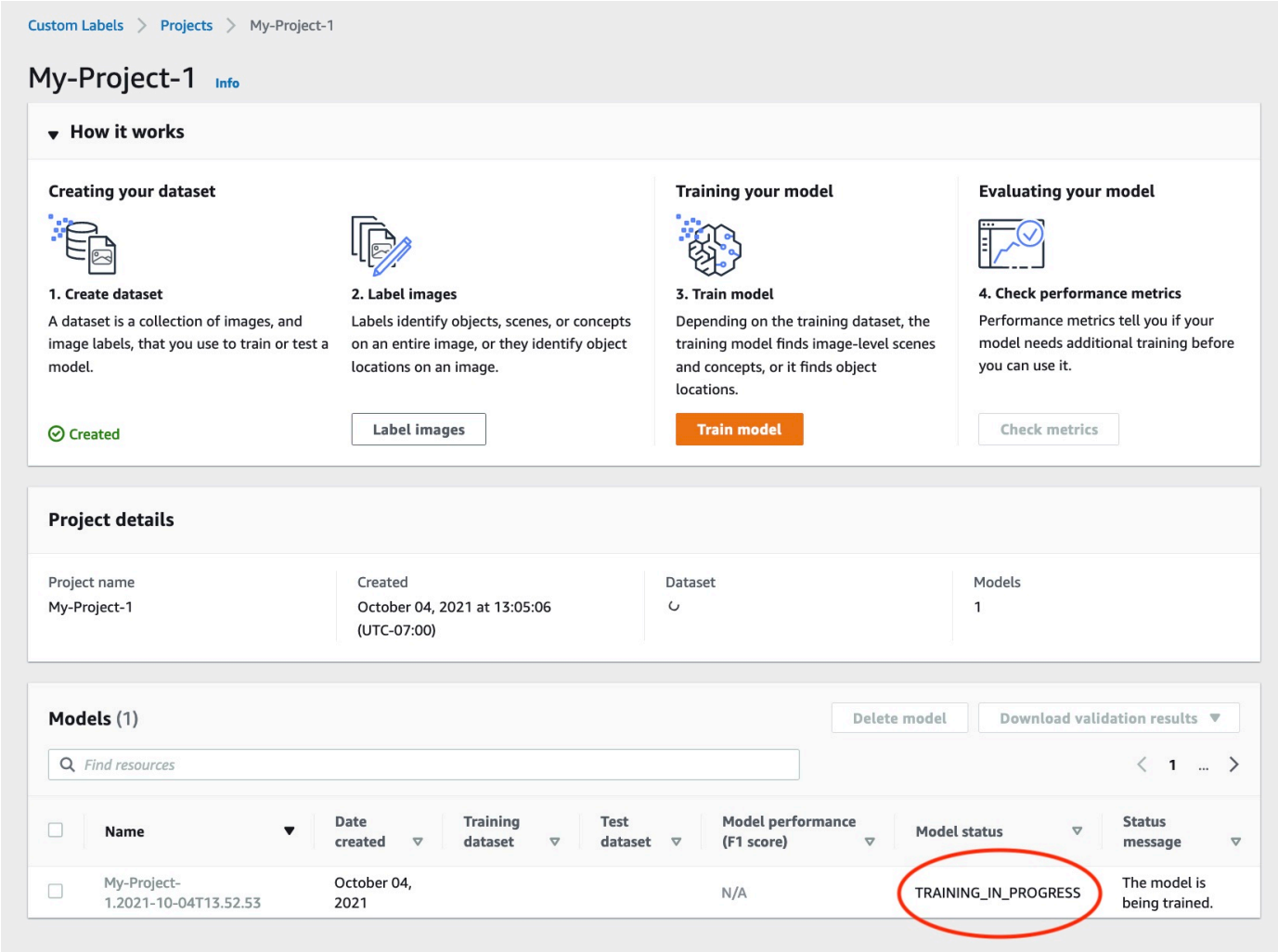
8. Wählen Sie auf der Seite Zugmodell die Option Zugmodell aus. Der Amazon-Ressourcenname (ARN) für Ihr Projekt sollte im Bearbeitungsfeld für Amazon Resource Name (ARN) für Ihr Projekt angegeben werden. Wenn nicht, geben Sie den ARN für Ihr Projekt ein.



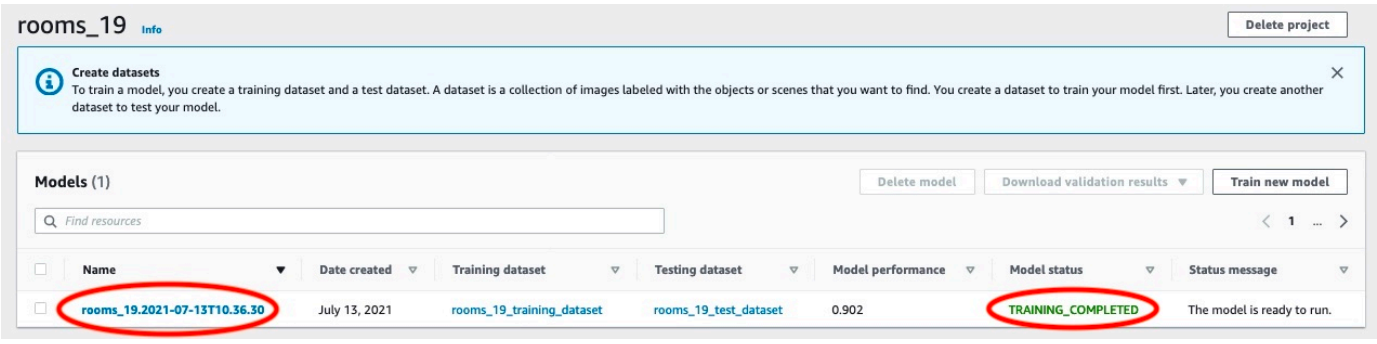
9. In der Möchtest du dein Model trainieren? Wählen Sie in einem Dialogfenster die Option Zugmodell aus.



10. Im Bereich Modelle auf der Projektseite kannst du den aktuellen Status in der`Model Status` Spalte überprüfen, in der das Training läuft. Es dauert eine Weile, bis das Training eines Modells abgeschlossen ist.



11. Wählen Sie nach Abschluss des Trainings den Modellnamen. Das Training ist beendet, wenn der Modellstatus TRAINING_COMPLETED ist. Wenn das Training fehlschlägt, lesen Sie[Debuggen eines fehlgeschlagenen Modelltrainings](#).



12. Nächster Schritt: Evaluieren Sie Ihr Modell. Weitere Informationen finden Sie unter [Verbessern eines geschulten Amazon Rekognition Custom Labels-Modells](#).

Ein Modell trainieren (SDK)

Du trainierst ein Model, indem du anrufst CreateProjectVersion. Um ein Modell zu trainieren, werden die folgenden Informationen benötigt:

- Name — Ein eindeutiger Name für die Modellversion.
- Project ARN) des Projekts, das das Modell verwaltet.
- Ort der Schulungsergebnisse — Der Amazon S3 S3-Standort, an dem die Ergebnisse platziert werden. Sie können denselben Speicherort wie den Amazon S3 S3-Bucket für die Konsole verwenden, oder Sie können einen anderen Standort wählen. Wir empfehlen, einen anderen Standort zu wählen, da Sie auf diese Weise Berechtigungen festlegen und mögliche Namenskonflikte mit dem Trainingsergebnis vermeiden können, das sich aus der Verwendung der Amazon Rekognition Custom Labels-Konsole ergibt.

Das Training verwendet die mit dem Projekt verknüpften Trainings- und Testdatensätze. Weitere Informationen finden Sie unter Verwalten von Datensätzen.

> **Note**
>
> Optional können Sie Manifestdateien für Trainings- und Testdatensätze angeben, die sich außerhalb eines Projekts befinden. Wenn Sie die Konsole öffnen, nachdem Sie ein Modell mit externen Manifestdateien trainiert haben, erstellt Amazon Rekognition Custom Labels die Datensätze für Sie, indem es den letzten Satz von Manifestdateien verwendet, die für das Training verwendet wurden. Sie können eine Modellversion für das Projekt nicht mehr trainieren, indem Sie externe Manifestdateien angeben. Weitere Informationen finden Sie unter CreatePrjectVersion.

Die Antwort von`CreateProjectVersion` ist ein ARN, mit dem Sie die Modellversion in nachfolgenden Anfragen identifizieren. Sie können auch die ARN verwenden, um die Modellversion zu sichern. Weitere Informationen finden Sie unter Sicherung von Amazon Rekognition Custom Labels-Projekten.

Es dauert eine Weile, bis das Training einer Modellversion abgeschlossen ist. In den Python- und Java-Beispielen in diesem Thema warten Kellner auf den Abschluss der Schulung. Ein Kellner ist eine nützliche Methode, mit der nach einem bestimmten Bundesstaat gefragt wird. Alternativ können Sie sich telefonisch über den aktuellen Stand der Schulung

informierenDescribeProjectVersions. Das Training ist abgeschlossen, wenn derStatus Feldwert beträgtTRAINING_COMPLETED. Nach Abschluss des Trainings können Sie die Qualität des Modells bewerten, indem Sie die Bewertungsergebnisse überprüfen.

Ein Modell trainieren (SDK)

Das folgende Beispiel zeigt, wie ein Modell mithilfe der einem Projekt zugeordneten Trainings- und Testdatensätze trainiert wird.

Um ein Modell zu trainieren (SDK)

1. Falls noch nicht erfolgt, installieren und konfigurieren Sie dieAWS CLIAWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Beispielcode, um ein Projekt zu trainieren.

   AWS CLI

   Im folgenden Beispiel wird ein Modell erstellt. Der Trainingsdatensatz wird aufgeteilt, um den Testdatensatz zu erstellen. Ersetzen Sie Folgendes:

   - my_project_arnmit dem Amazon-Ressourcennamen (ARN) des Projekts.
   - version_namemit einem eindeutigen Versionsnamen Ihrer Wahl.
   - output_bucketmit dem Namen des Amazon-S3-Buckets, in dem Amazon Rekognition Custom Labels die Trainingsergebnisse speichert.
   - output_foldermit dem Namen des Ordners, in dem die Trainingsergebnisse gespeichert werden.
   - (optionaler Parameter)--kms-key-id mit einer Kennung für Ihren AWS Key Management Service Service-Kundenmasterschlüssel.

```
aws rekognition create-project-version \
  --project-arn project_arn \
  --version-name version_name \
  --output-config '{"S3Bucket":"output_bucket", "S3KeyPrefix":"output_folder"}'
 \
  --profile custom-labels-access
```

Python

Im folgenden Beispiel wird ein Modell erstellt. Geben Sie die folgenden Befehlszeilenargumente an:

- `project_arn`— Der Amazon-Ressourcenname (ARN) des Projekts.
- `version_name`— Ein eindeutiger Versionsname für das Modell Ihrer Wahl.
- `output_bucket`— Der Name des Amazon-S3-Buckets, in dem Amazon Rekognition Custom Labels die Trainingsergebnisse speichert.
- `output_folder`— der Name des Ordners, in dem die Trainingsergebnisse gespeichert werden.

Geben Sie optional die folgenden Befehlszeilenparameter an, um Ihrem Modell ein Tag zuzuordnen:

- `tag`— ein Tagname Ihrer Wahl, den Sie an dem Modell anhängen möchten.
- `tag_value`der Tag-Wert.

```
#Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see https://github.com/awsdocs/
amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)


import argparse
import logging
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def train_model(rek_client, project_arn, version_name, output_bucket,
 output_folder, tag_key, tag_key_value):
    """
    Trains an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
```

```
   :param project_arn: The ARN of the project in which you want to train a
model.
   :param version_name: A version for the model.
   :param output_bucket: The S3 bucket that hosts training output.
   :param output_folder: The path for the training output within output_bucket
   :param tag_key: The name of a tag to attach to the model. Pass None to
exclude
   :param tag_key_value: The value of the tag. Pass None to exclude

   """

   try:
       #Train the model

       status=""
       logger.info("training model version %s for project %s",
           version_name, project_arn)


       output_config = json.loads(
           '{"S3Bucket": "'
           + output_bucket
           + '", "S3KeyPrefix": "'
           + output_folder
           + '" }  '
       )

       tags={}

       if tag_key is not None and tag_key_value is not None:
           tags = json.loads(
               '{"' + tag_key + '":"' + tag_key_value + '"}'
           )


       response=rek_client.create_project_version(
           ProjectArn=project_arn,
           VersionName=version_name,
           OutputConfig=output_config,
           Tags=tags
       )

       logger.info("Started training: %s", response['ProjectVersionArn'])
```

```
        # Wait for the project version training to complete.

        project_version_training_completed_waiter =
 rek_client.get_waiter('project_version_training_completed')
        project_version_training_completed_waiter.wait(ProjectArn=project_arn,
        VersionNames=[version_name])


        # Get the completion status.

 describe_response=rek_client.describe_project_versions(ProjectArn=project_arn,
            VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            logger.info("Status: %s", model['Status'])
            logger.info("Message: %s", model['StatusMessage'])
            status=model['Status']


        logger.info("finished training")

        return response['ProjectVersionArn'], status

    except ClientError as err:
        logger.exception("Couldn't create model: %s", err.response['Error']
['Message'] )
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to train a
 model"
    )

    parser.add_argument(
        "version_name", help="A version name of your choosing."
    )

    parser.add_argument(
```

```
        "output_bucket", help="The S3 bucket that receives the training
 results."
    )

    parser.add_argument(
        "output_folder", help="The folder in the S3 bucket where training
 results are stored."
    )

    parser.add_argument(
        "--tag_name",  help="The name of a tag to attach to the model",
 required=False
    )

    parser.add_argument(
        "--tag_value",  help="The value for the tag.", required=False
    )




def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Training model version {args.version_name} for project
 {args.project_arn}")

        # Train the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        model_arn, status=train_model(rekognition_client,
            args.project_arn,
            args.version_name,
            args.output_bucket,
            args.output_folder,
```

```
            args.tag_name,
            args.tag_value)


        print(f"Finished training model: {model_arn}")
        print(f"Status: {status}")


    except ClientError as err:
        logger.exception("Problem training model: %s", err)
        print(f"Problem training model: {err}")
    except Exception as err:
        logger.exception("Problem training model: %s", err)
        print(f"Problem training model: {err}")


if __name__ == "__main__":
    main()
```

Java V2

Im folgenden Beispiel wird ein Modell trainiert. Geben Sie die folgenden Befehlszeilenargumente an:

- `project_arn`— Der Amazon-Ressourcenname (ARN) des Projekts.
- `version_name`— Ein eindeutiger Versionsname für das Modell Ihrer Wahl.
- `output_bucket`— Der Name des Amazon-S3-Buckets, in dem Amazon Rekognition Custom Labels die Trainingsergebnisse speichert.
- `output_folder`— der Name des Ordners, in dem die Trainingsergebnisse gespeichert werden.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CreateProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.CreateProjectVersionResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.waiters.RekognitionWaiter;

import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;

public class TrainModel {

    public static final Logger logger =
 Logger.getLogger(TrainModel.class.getName());

    public static String trainMyModel(RekognitionClient rekClient, String
 projectArn, String versionName,
            String outputBucket, String outputFolder) {

        try {

            OutputConfig outputConfig =
 OutputConfig.builder().s3Bucket(outputBucket).s3KeyPrefix(outputFolder).build();

            logger.log(Level.INFO, "Training Model for project {0}",
 projectArn);
            CreateProjectVersionRequest createProjectVersionRequest =
 CreateProjectVersionRequest.builder()

 .projectArn(projectArn).versionName(versionName).outputConfig(outputConfig).build();

            CreateProjectVersionResponse response =
 rekClient.createProjectVersion(createProjectVersionRequest);
```

```
            logger.log(Level.INFO, "Model ARN: {0}",
response.projectVersionArn());
            logger.log(Level.INFO, "Training model...");

            // wait until training completes

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                    .versionNames(versionName)
                    .projectArn(projectArn)
                    .build();

            RekognitionWaiter waiter = rekClient.waiter();

            WaiterResponse<DescribeProjectVersionsResponse> waiterResponse =
waiter

.waitUntilProjectVersionTrainingCompleted(describeProjectVersionsRequest);

            Optional<DescribeProjectVersionsResponse> optionalResponse =
waiterResponse.matched().response();

            DescribeProjectVersionsResponse describeProjectVersionsResponse =
optionalResponse.get();

            for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                    .projectVersionDescriptions()) {
                System.out.println("ARN: " +
projectVersionDescription.projectVersionArn());
                System.out.println("Status: " +
projectVersionDescription.statusAsString());
                System.out.println("Message: " +
projectVersionDescription.statusMessage());
            }

            return response.projectVersionArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not train model: {0}",
e.getMessage());
            throw e;
        }
```

```
    }

    public static void main(String args[]) {

        String versionName = null;
        String projectArn = null;
        String projectVersionArn = null;
        String bucket = null;
        String location = null;

        final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>
 <output_bucket> <output_folder>\n\n" + "Where:\n"
                + "   project_arn - The ARN of the project that you want to use.
 \n\n"
                + "   version_name - A version name for the model.\n\n"
                + "   output_bucket - The S3 bucket in which to place the
 training output. \n\n"
                + "   output_folder - The folder within the bucket that the
 training output is stored in. \n\n";

        if (args.length != 4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        versionName = args[1];
        bucket = args[2];
        location = args[3];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Train model
            projectVersionArn = trainMyModel(rekClient, projectArn, versionName,
 bucket, location);
```

```
            System.out.println(String.format("Created model: %s for Project ARN:
 %s", projectVersionArn, projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

3. Wenn das Training fehlschlägt, lesen SieDebuggen eines fehlgeschlagenen Modelltrainings.

Debuggen eines fehlgeschlagenen Modelltrainings

Beim Modelltraining können Fehler auftreten. Amazon Rekognition Custom Labels meldet Trainingsfehler in der Konsole und im Antwortformular. DescribeProjectVersions

Fehler sind entweder terminal (das Training kann nicht fortgesetzt werden) oder sie sind nicht terminal (das Training kann fortgesetzt werden). Bei Fehlern, die sich auf den Inhalt der Trainings- und Testdatensätze beziehen, können Sie die Validierungsergebnisse herunterladen (eine Zusammenfassung des Manifests und Manifeste zur Trainings- und Testvalidierung). Verwenden Sie die Fehlercodes in den Überprüfungsergebnissen, um weitere Informationen in diesem Abschnitt zu finden. Dieser Abschnitt enthält auch Informationen zu Manifestdateifehlern (Terminalfehler, die auftreten, bevor der Inhalt der Manifestdatei überprüft wird).

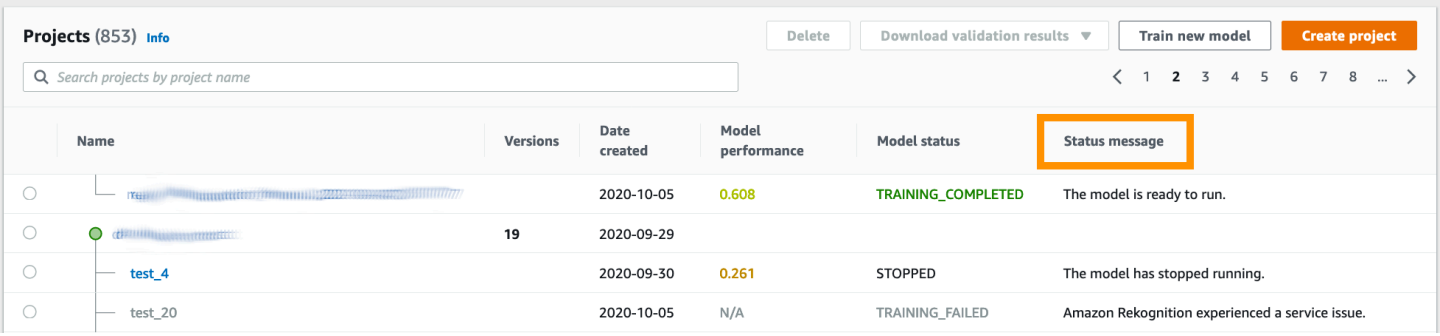
> **Note**
>
> Ein Manifest ist die Datei, die verwendet wird, um den Inhalt eines Datensatzes zu speichern.

Sie können einige Fehler beheben, indem Sie die Amazon Rekognition Custom Labels-Konsole verwenden. Bei anderen Fehlern müssen Sie möglicherweise Aktualisierungen der Trainings- oder Testmanifestdateien vornehmen. Möglicherweise müssen Sie weitere Änderungen vornehmen, z. B. die IAM-Berechtigungen. Weitere Informationen finden Sie in der -Dokumentation zu den einzelnen Fehlern.

Terminalfehler

Terminalfehler verhindern das Training eines Modells. Es gibt drei Kategorien von Terminal-Trainingsfehlern: Servicefehler, Manifestdateifehler und Manifest-Inhaltsfehler.

In der Konsole zeigt Amazon Rekognition Custom Labels Terminalfehler für ein Modell in der Spalte mit Statusmeldungen auf der Projektseite an.



Wenn Sie das AWS SDK verwenden, können Sie anhand der Antwort von herausfinden, ob ein Terminal-Manifestdateifehler oder ein Terminal-Manifest-Inhaltsfehler aufgetreten ist. DescribeProjectVersions In diesem Fall ist der `Status` Wert `TRAINING_FAILED` und das `StatusMessage` Feld enthält den Fehler.

Servicefehler

Terminalservice-Fehler treten auf, wenn bei Amazon Rekognition ein Serviceproblem auftritt und das Training nicht fortgesetzt werden kann. Zum Beispiel der Ausfall eines anderen Dienstes, von dem Amazon Rekognition Custom Labels abhängt. Amazon Rekognition Custom Labels meldet Servicefehler in der Konsole, da bei Amazon Rekognition ein Serviceproblem aufgetreten ist. Wenn Sie das AWS SDK verwenden, werden Servicefehler, die während des Trainings auftreten, `InternalServerError` ausnahmsweise von CreateProjectVersionund ausgelöst DescribeProjectVersions.

Wenn ein Servicefehler auftritt, versuchen Sie erneut, das Modell zu trainieren. Wenn das Training weiterhin fehlschlägt, wenden Sie sich an den AWS-Support und fügen Sie dem Servicefehler alle gemeldeten Fehlerinformationen bei.

Fehler in der Terminal-Manifest-Datei

Manifestdateifehler sind Terminalfehler in den Trainings- und Testdatensätzen, die auf Dateiebene oder in mehreren Dateien auftreten. Fehler in der Manifestdatei werden erkannt, bevor der Inhalt

der Trainings- und Testdatensätze validiert wird. Fehler in der Manifestdatei verhindern die Meldung von Validierungsfehlern, die nicht vom Terminal stammen. Beispielsweise generiert eine leere Trainingsmanifestdatei den Fehler Die Manifestdatei ist leer. Da die Datei leer ist, können keine Fehler bei der Überprüfung von JSON-Zeilen außerhalb des Terminals gemeldet werden. Die Manifestzusammenfassung wird ebenfalls nicht erstellt.

Sie müssen Fehler in der Manifestdatei beheben, bevor Sie Ihr Modell trainieren können.

Im Folgenden werden die Fehler in der Manifestdatei aufgeführt.

- Die Manifest-Dateierweiterung oder der Inhalt sind ungültig.
- Die Manifestdatei ist leer.
- Die Manifestdateigröße überschreitet die maximal unterstützte Größe.
- In den S3-Ausgabe-Bucket kann nicht geschrieben werden.
- Die S3-Bucket-Berechtigungen sind falsch.

Terminal-Manifest-Inhaltsfehler

Manifestinhaltsfehler sind Terminalfehler, die sich auf den Inhalt eines Manifests beziehen. Wenn Sie beispielsweise die Fehlermeldung Die Manifestdatei enthält nicht genügend beschriftete Bilder pro Label erhalten, um die automatische Aufteilung durchzuführen, kann das Training nicht abgeschlossen werden, da der Trainingsdatensatz nicht genügend beschriftete Bilder enthält, um einen Testdatensatz zu erstellen.

Der Fehler wird nicht nur in der Konsole und im Antwortformular gemeldet`DescribeProjectVersions`, sondern auch in der Manifestzusammenfassung zusammen mit allen anderen Inhaltsfehlern des Terminals gemeldet. Weitere Informationen finden Sie unter Verstehen der Manifestübersicht.

JSON-Line-Fehler, die nicht terminal sind, werden auch in separaten Listen mit den Ergebnissen der Trainings- und Testvalidierung gemeldet. Die von Amazon Rekognition Custom Labels gefundenen JSON-Line-Fehler ohne Terminal stehen nicht unbedingt im Zusammenhang mit den offensichtlichen Inhaltsfehlern, die das Training beenden. Weitere Informationen finden Sie unter Grundlegendes zu den Ergebnissen der Trainings- und Testvalidierung.

Sie müssen Fehler im Manifest beheben, bevor Sie Ihr Modell trainieren können.

Im Folgenden sind die Fehlermeldungen für offensichtliche Inhaltsfehler aufgeführt.

- Die Manifestdatei enthält zu viele ungültige Zeilen.
- Die Manifestdatei enthält Bilder aus mehreren S3-Buckets.
- Ungültige Besitzer-ID für den S3-Bucket für Bilder.
- Die Manifestdatei enthält nicht genügend beschriftete Bilder pro Label, um eine automatische Aufteilung durchzuführen.
- Die Manifestdatei hat zu wenige Bezeichnungen.
- Die Manifestdatei hat zu viele Bezeichnungen.
- Weniger als {}% Label-Überlappung zwischen den Trainings- und Testmanifestdateien.
- Die Manifestdatei enthält zu wenige verwendbare Labels.
- Weniger als {}% verwendbares Label überschneiden sich zwischen den Trainings- und Testmanifestdateien.
- Bilder konnten nicht aus dem S3-Bucket kopiert werden.

Fehler bei der Überprüfung von JSON-Zeilen außerhalb des Terminals

Bei JSON-Line-Validierungsfehlern handelt es sich nicht um terminale Fehler, bei denen Amazon Rekognition Custom Labels das Training eines Modells nicht beenden muss.

Fehler bei der JSON-Zeilenüberprüfung werden in der Konsole nicht angezeigt.

In den Trainings- und Testdatensätzen stellt eine JSON-Zeile die Trainings- oder Testinformationen für ein einzelnes Bild dar. Validierungsfehler in einer JSON-Zeile, wie z. B. ein ungültiges Bild, werden in den Validierungsmanifesten für Training und Test gemeldet. Amazon Rekognition Custom Labels schließt das Training mit den anderen gültigen JSON-Zeilen ab, die im Manifest enthalten sind. Weitere Informationen finden Sie unter Grundlegendes zu den Ergebnissen der Trainings- und Testvalidierung. Informationen zu Validierungsregeln finden Sie unterValidierungsregeln für Manifestdateien.

> **Note**
>
> Das Training schlägt fehl, wenn zu viele JSON-Line-Fehler auftreten.

Wir empfehlen, dass Sie auch JSON-Line-Fehler beheben, die nicht im Terminal ausgeführt werden, da diese möglicherweise zu future Fehlern führen oder Ihr Modelltraining beeinträchtigen können.

Amazon Rekognition Custom Labels kann die folgenden Fehler bei der Überprüfung von JSON-Zeilen generieren, die nicht vom Terminal stammen.

- Der Quellrefer-Schlüssel fehlt.
- Das Format des Quellreferenzwerts ist ungültig.
- Keine Labelattribute gefunden.
- Das Format des Label-Attributs {} ist ungültig.
- Das Format des Labels attributemetadata ist ungültig.
- Keine gültigen Labelattribute gefunden.
- Ein oder mehrere Begrenzungsrahmen weisen einen fehlenden Konfidenzwert auf.
- Eine oder mehrere Klassen-IDs fehlen in der Klassenzuordnung.
- Die JSON-Zeile hat ein ungültiges Format.
- Das Bild ist ungültig. Überprüfen Sie den S3-Pfad und/oder die Bildeigenschaften.
- Die Begrenzungsbox enthält Off-Frame-Werte.
- Die Höhe und Breite des Begrenzungsrahmens sind zu klein.
- Es gibt mehr Begrenzungsfelder als das zulässige Maximum.
- Keine gültigen Anmerkungen gefunden.

Verstehen der Manifestübersicht

Die Manifestübersicht enthält die folgenden Informationen.

- Fehlerinformationen über den bei der Validierung Terminal-Manifest-Inhaltsfehler aufgetretenen Fehler.
- Informationen zur Fehlerposition Fehler bei der Überprüfung von JSON-Zeilen außerhalb des Terminals in den Trainings- und Testdatensätzen.
- Fehlerstatistiken wie die Gesamtzahl der ungültigen JSON-Zeilen, die in den Trainings- und Testdatensätzen gefunden wurden.

Die Manifestzusammenfassung wird während des Trainings erstellt, falls keine vorhanden istFehler in der Terminal-Manifest-Datei. Den Speicherort der Manifest-Zusammenfassungsdatei (manifest_summary.json) finden Sie unter. Abrufen der Validierungsergebnisse

> **Note**
>
> Dienstfehler und Manifestdateifehler werden in der Manifestzusammenfassung nicht gemeldet. Weitere Informationen finden Sie unter Terminalfehler.

Hinweise zu bestimmten Fehlern im Manifestinhalt finden Sie unterTerminal-Manifest-Inhaltsfehler.

Format der Manifest-Zusammenfassungsdatei

Eine Manifestdatei besteht aus 2 Abschnitten `statistics` und`errors`.

statistics

`statistics`enthält Informationen zu den Fehlern in den Trainings- und Testdatensätzen.

- `training`— Statistiken und Fehler im Trainingsdatensatz.
- `testing`— Statistiken und Fehler, die im Testdatensatz gefunden wurden.

Die Objekte im `errors` Array enthalten den Fehlercode und die Meldung für Fehler im offensichtlichen Inhalt.

Das `error_line_indices` Array enthält die Zeilennummern für jede JSON-Zeile im Trainings- oder Testmanifest, die einen Fehler enthält. Weitere Informationen finden Sie unter Behebung von Trainingsfehlern.

Fehler

Fehler, die sich sowohl auf den Trainings- als auch auf den Testdatensatz beziehen. Ein FEHLER_UNGENÜGEND_VERWENDBARE_LABEL-ÜBERLAPPUNG tritt beispielsweise auf, wenn nicht genügend verwendbare Labels vorhanden sind, die sich mit den Trainings- und Testdatensätzen überschneiden.

```
{
    "statistics": {
        "training":
            {
                "use_case": String, # Possible values are IMAGE_LEVEL_LABELS,
 OBJECT_LOCALIZATION and NOT_DETERMINED
```

```
                "total_json_lines": Number,   # Total number json lines (images) in the
 training manifest.
                "valid_json_lines": Number,   # Total number of JSON Lines (images)
that can be used for training.
                "invalid_json_lines": Number, # Total number of invalid JSON Lines.
They are not used for training.
                "ignored_json_lines": Number, # JSON Lines that have a valid schema but
have no annotations. The aren't used for training and aren't counted as invalid.
                "error_json_line_indices": List[int], # Contains a list of line numbers
for JSON line errors in the training dataset.
                "errors": [
                    {
                        "code": String, # Error code for a training manifest content
error.
                        "message": String # Description for a training manifest content
error.
                    }
                ]
            },
        "testing":
            {
                "use_case": String, # Possible values are IMAGE_LEVEL_LABELS,
OBJECT_LOCALIZATION and NOT_DETERMINED
                "total_json_lines": Number, # Total number json lines (images) in the
manifest.
                "valid_json_lines": Number,  # Total number of JSON Lines (images) that
can be used for testing.
                "invalid_json_lines": Number, # Total number of invalid JSON Lines.
They are not used for testing.
                "ignored_json_lines": Number, # JSON Lines that have a valid schema but
have no annotations. They aren't used for testing and aren't counted as invalid.
                "error_json_line_indices": List[int], # contains a list of error record
line numbers in testing dataset.
                "errors": [
                    {
                        "code": String,   # # Error code for a testing manifest content
error.
                        "message": String # Description for a testing manifest content
error.
                    }
                ]
            }
    },
    "errors": [
```

```
        {
            "code": String, # # Error code for errors that span the training and
 testing datasets.
            "message": String # Description of the error.
        }
    ]
}
```

Beispiel für eine Manifestübersicht

Das folgende Beispiel ist eine teilweise Manifestzusammenfassung, die einen Fehler im Terminal-Manifestinhalt zeigt (FEHLER_ZU_VIELE_UNGÜLTIGE ZEILEN_IM_MANIFEST). Das `error_json_line_indices` Array enthält die Zeilennummern der nicht terminalen JSON-Zeilenfehler im entsprechenden Trainings- oder Testvalidierungsmanifest.

```
{
    "errors": [],
    "statistics": {
        "training": {
            "use_case": "NOT_DETERMINED",
            "total_json_lines": 301,
            "valid_json_lines": 146,
            "invalid_json_lines": 155,
            "ignored_json_lines": 0,
            "errors": [
                {
                    "code": "ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST",
                    "message": "The manifest file contains too many invalid rows."
                }
            ],
            "error_json_line_indices": [
                15,
                16,
                17,
                22,
                23,
                24,
                 .
                 .
                 .
                 .
                300
            ]
```

```
        },
        "testing": {
            "use_case": "NOT_DETERMINED",
            "total_json_lines": 15,
            "valid_json_lines": 13,
            "invalid_json_lines": 2,
            "ignored_json_lines": 0,
            "errors": [],
            "error_json_line_indices": [
                13,
                15
            ]
        }
    }
}
```

Grundlegendes zu den Ergebnissen der Trainings- und Testvalidierung

Während des Trainings erstellt Amazon Rekognition Custom Labels Validierungsergebnislisten, in denen JSON-Zeilenfehler enthalten sind, die nicht vom Terminal stammen. Die Manifeste der Validierungsergebnisse sind Kopien der Trainings- und Testdatensätze mit hinzugefügten Fehlerinformationen. Sie können nach Abschluss des Trainings auf die Validierungsmanifeste zugreifen. Weitere Informationen finden Sie unter Abrufen der Validierungsergebnisse. Amazon Rekognition Custom Labels erstellt außerdem eine Manifestzusammenfassung, die Übersichtsinformationen zu JSON-Zeilenfehlern enthält, wie z. B. Fehlerorte und Anzahl der JSON-Zeilenfehler. Weitere Informationen finden Sie unter Verstehen der Manifestübersicht.

> **Note**
>
> Validierungsergebnisse (Manifeste mit den Ergebnissen der Trainings- und Testvalidierung und Manifestzusammenfassung) werden nur erstellt, wenn keine Fehler in der Terminal-Manifest-Datei Ergebnisse vorliegen.

Ein Manifest enthält JSON-Zeilen für jedes Bild im Datensatz. In den Manifesten der Validierungsergebnisse werden JSON-Zeilenfehlerinformationen zu den JSON-Zeilen hinzugefügt, in denen Fehler auftreten.

Ein JSON-Zeilenfehler ist ein nicht terminaler Fehler, der sich auf ein einzelnes Bild bezieht. Ein nicht terminaler Validierungsfehler kann die gesamte JSON-Zeile oder nur einen Teil davon ungültig machen. Wenn das in einer JSON-Zeile referenzierte Bild beispielsweise nicht im PNG- oder JPG-Format vorliegt, tritt ein FEHLER_UNGÜLTIGES BILD Fehler auf und die gesamte JSON-Zeile wird vom Training ausgeschlossen. Das Training wird mit anderen gültigen JSON-Zeilen fortgesetzt.

Innerhalb einer JSON-Zeile kann ein Fehler bedeuten, dass die JSON-Zeile weiterhin für das Training verwendet werden kann. Wenn beispielsweise der linke Wert für einen von vier Begrenzungsrahmen, die einer Bezeichnung zugeordnet sind, negativ ist, wird das Modell trotzdem mit den anderen gültigen Begrenzungsrahmen trainiert. Für das ungültige Begrenzungsfeld (ERROR_INVALID_BOUNDING_BOX) werden Informationen zu JSON-Zeilenfehlern zurückgegeben. In diesem Beispiel werden die Fehlerinformationen zu dem `annotation` Objekt hinzugefügt, in dem der Fehler auftritt.

Warnfehler wie WARNUNG_NO_ANNOTATIONS z. B. werden nicht für das Training verwendet und gelten in der Manifestzusammenfassung als ignorierte JSON-Zeilen (`ignored_json_lines`). Weitere Informationen finden Sie unter Verstehen der Manifestübersicht. Außerdem werden ignorierte JSON-Zeilen nicht auf die Fehlerschwelle von 20% für Schulungen und Tests angerechnet.

Hinweise zu bestimmten Fehlern bei der Datenüberprüfung außerhalb des Terminals finden Sie unterFehler bei der Überprüfung von JSON-Zeilen ohne Terminal.

> **Note**
>
> Wenn zu viele Fehler bei der Datenüberprüfung auftreten, wird das Training beendet und in der Manifestzusammenfassung wird ein FEHLER_ZU_VIELE_UNGÜLTIGE ZEILEN_IM_MANIFEST Terminalfehler gemeldet.

Hinweise zur Korrektur von JSON-Line-Fehlern finden Sie unterBehebung von Trainingsfehlern.

JSON-Zeilenfehlerformat

Amazon Rekognition Custom Labels fügt JSON-Lines im Format JSON Lines auf Bildebene und Objektlokalisierung Informationen zu Validierungsfehlern hinzu, die nicht vom Terminal stammen. Weitere Informationen finden Sie unter the section called “Erstellen einer Manifestdatei”.

Fehler auf Bildebene

Das folgende Beispiel zeigt die `Error` Arrays in einer JSON-Zeile auf Bildebene. Es gibt zwei Arten von Fehlern. Fehler im Zusammenhang mit den Metadaten des Label-Attributs (in diesem Beispiel Sport-Metadaten) und Fehler im Zusammenhang mit dem Bild. Ein Fehler beinhaltet einen Fehlercode (Code) und eine Fehlermeldung (Nachricht). Weitere Informationen finden Sie unter Labels auf Bildebene in Manifestdateien.

```
{
    "source-ref": String,
    "sport": Number,
    "sport-metadata": {
        "class-name": String,
        "confidence": Float,
        "type": String,
        "job-name": String,
        "human-annotated": String,
        "creation-date": String,
        "errors": [
            {
                "code": String, # error codes for label
                "message": String # Description and additional contextual details of
 the error
            }
        ]
    },
    "errors": [
        {
            "code": String, # error codes for image
            "message": String # Description and additional contextual details of the
 error
        }
    ]
}
```

Fehler bei der Objektlokalisierung

Das folgende Beispiel zeigt die Fehlerarrays in einer JSON-Zeile zur Objektlokalisierung. Die JSON-Zeile enthält eine `Errors` Array-Information für Felder in den folgenden JSON-Zeilenabschnitten. Jedes `Error` Objekt enthält den Fehlercode und die Fehlermeldung.

- Label-Attribut — Fehler in den Label-Attributfeldern. Siehe `bounding-box` im Beispiel.

- Anmerkungen — Annotationsfehler (Bounding Boxes) werden im `annotations` Array innerhalb des Label-Attributs gespeichert.
- labelattribute-metadata — Fehler bei den Metadaten des Label-Attributs. Siehe `bounding-box-metadata` im Beispiel.
- image — Fehler, die nichts mit den Metadatenfeldern Label-Attribut, Anmerkung und Label-Attribut zu tun haben.

Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien.

```
{
    "source-ref": String,
    "bounding-box": {
        "image_size": [
            {
                "width": Int,
                "height": Int,
                "depth":Int,
            }
        ],
        "annotations": [
            {
                "class_id": Int,
                "left": Int,
                "top": Int,
                "width": Int,
                "height": Int,
                "errors": [   # annotation field errors
                    {
                        "code": String, # annotation field error code
                        "message": String # Description and additional contextual
 details of the error
                    }
                ]
            }
        ],
        "errors": [ #label attribute field errors
            {
                "code": String, # error code
                "message": String # Description and additional contextual details of
 the error
            }
```

```
        ]
    },
    "bounding-box-metadata": {
        "objects": [
            {
                "confidence": Float
            }
        ],
        "class-map": {
            String: String
        },
        "type": String,
        "human-annotated": String,
        "creation-date": String,
        "job-name": String,
        "errors": [  #metadata field errors
            {
                "code": String, # error code
                "message": String # Description and additional contextual details of
the error
            }
        ]
   },
  "errors": [  # image errors
        {
            "code": String, # error code
            "message": String # Description and additional contextual details of the
error
        }
   ]
}
```

Beispiel für einen JSON-Zeilenfehler

Die folgende JSON-Zeile zur Objektlokalisierung (aus Gründen der Lesbarkeit formatiert) zeigt einen ERROR_BOUNDING_BOX_ZU KLEIN Fehler. In diesem Beispiel sind die Abmessungen des Begrenzungsrahmens (Höhe und Breite) nicht größer als 1 x 1.

```
{
    "source-ref": "s3://bucket/Manifests/images/199940-1791.jpg",
    "bounding-box": {
        "image_size": [
            {
```

```
                "width": 3000,
                "height": 3000,
                "depth": 3
            }
        ],
        "annotations": [
            {
                "class_id": 1,
                "top": 0,
                "left": 0,
                "width": 1,
                "height": 1,
                "errors": [
                    {
                        "code": "ERROR_BOUNDING_BOX_TOO_SMALL",
                        "message": "The height and width of the bounding box is too
small."
                    }
                ]
            },
            {
                "class_id": 0,
                "top": 65,
                "left": 86,
                "width": 220,
                "height": 334
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [
            {
                "confidence": 1
            },
            {
                "confidence": 1
            }
        ],
        "class-map": {
            "0": "Echo",
            "1": "Echo Dot"
        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
```

```
        "creation-date": "2019-11-20T02:57:28.288286",
        "job-name": "my job"
    }
}
```

Abrufen der Validierungsergebnisse

Die Überprüfungsergebnisse enthalten Fehlerinformationen für Terminal-Manifest-Inhaltsfehler undFehler bei der Überprüfung von JSON-Zeilen außerhalb des Terminals. Es gibt drei Dateien mit Überprüfungsergebnissen.

- training_manifest_with_validation.json — Eine Kopie der Manifestdatei des Trainingsdatensatzes mit hinzugefügten JSON-Line-Fehlerinformationen.
- testing_manifest_with_validation.json — Eine Kopie der Manifestdatei des Testdatensatzes mit hinzugefügten JSON-Line-Fehlerinformationen.
- manifest_summary.json — Eine Zusammenfassung der Fehler im Manifestinhalt und der JSON-Zeilenfehler, die in den Trainings- und Testdatensätzen gefunden wurden. Weitere Informationen finden Sie unter Verstehen der Manifestübersicht.

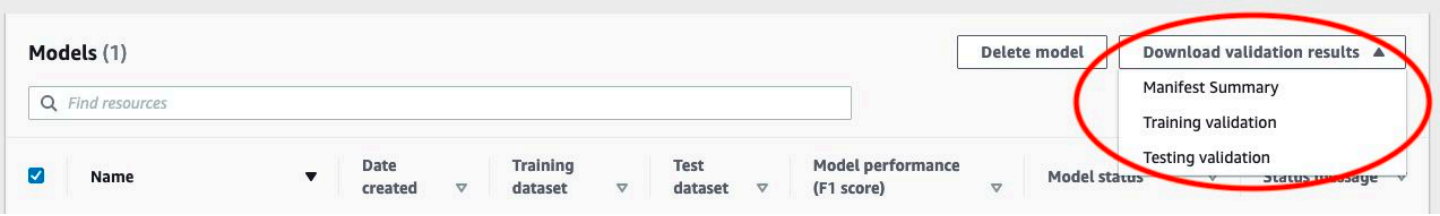
Informationen zum Inhalt der Validierungsmanifeste für Schulungen und Tests finden Sie unterDebuggen eines fehlgeschlagenen Modelltrainings.

> **Note**
>
> - Die Validierungsergebnisse werden nur erstellt, wenn keine Ergebnisse während des Trainings generiert Fehler in der Terminal-Manifest-Datei werden.
> - Tritt nach der Validierung des Trainings- und Testmanifests ein Servicefehler auf, werden die Validierungsergebnisse erstellt, aber die Antwort von enthält DescribeProjectVersionsnicht die Speicherorte der Validierungsergebnisdateien.

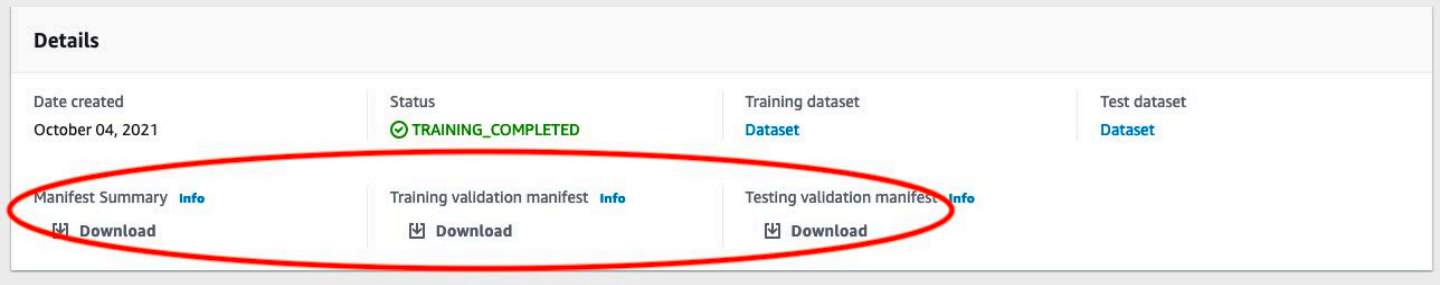
Nach Abschluss oder Fehlschlagen der Schulung können Sie die Validierungsergebnisse mithilfe der Amazon Rekognition Custom Labels-Konsole herunterladen oder den Amazon S3 S3-Bucket-Standort abrufen, indem Sie die API aufrufen DescribeProjectVersions.

Validierungsergebnisse abrufen (Konsole)

Wenn Sie die Konsole verwenden, um Ihr Modell zu trainieren, können Sie die Validierungsergebnisse aus der Modellliste eines Projekts herunterladen, wie in der folgenden Abbildung dargestellt.



Sie können die Validierungsergebnisse auch von der Detailseite eines Modells herunterladen.



Weitere Informationen finden Sie unter Ein Model trainieren (Konsole).

Validierungsergebnisse abrufen (SDK)

Nach Abschluss des Modelltrainings speichert Amazon Rekognition Custom Labels die Validierungsergebnisse in dem Amazon S3 S3-Bucket, der während des Trainings angegeben wurde. Sie können den Standort des S3-Buckets abrufen, indem Sie die DescribeProjectVersionsAPI aufrufen, nachdem das Training abgeschlossen ist. Informationen zum Trainieren eines Modells finden Sie unterEin Modell trainieren (SDK).

Ein ValidationDataObjekt wird für den Trainingsdatensatz (TrainingDataResult) und den Testdatensatz (TestingDataResult) zurückgegeben. Die Manifestzusammenfassung wird zurückgegeben`ManifestSummary`.

Nachdem Sie den Amazon S3 S3-Bucket-Standort abgerufen haben, können Sie die Validierungsergebnisse herunterladen. Weitere Informationen finden Sie unter Wie lade ich ein Objekt von einem S3-Bucket herunter? . Sie können die GetObjectOperation auch verwenden.

Um Validierungsdaten abzurufen (SDK)

1. Falls noch nicht geschehen, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie das folgende Beispiel, um den Speicherort der Überprüfungsergebnisse zu ermitteln.

   Python

   Ersetzen Sie `project_arn` durch den Amazon-Ressourcennamen (ARN) des Projekts, das das Modell enthält. Weitere Informationen finden Sie unter Verwalten eines Amazon Rekognition Custom Labels-Projekts. `version_name`Ersetzen Sie durch den Namen der Modellversion. Weitere Informationen finden Sie unter Ein Modell trainieren (SDK).

```
import boto3
import io
from io import BytesIO
import sys
import json


def describe_model(project_arn, version_name):

    client=boto3.client('rekognition')

    response=client.describe_project_versions(ProjectArn=project_arn,
        VersionNames=[version_name])

    for model in response['ProjectVersionDescriptions']:
        print(json.dumps(model,indent=4,default=str))

def main():

    project_arn='project_arn'
    version_name='version_name'

    describe_model(project_arn, version_name)

if __name__ == "__main__":
    main()
```

3. Notieren Sie sich in der Programmausgabe das `Validation` Feld innerhalb der `TrainingDataResult` Objekte `TestingDataResult` und. Die Zusammenfassung des Manifests ist da`ManifestSummary`.

Behebung von Trainingsfehlern

Sie verwenden die Zusammenfassung des Manifests, um zu identifizieren, Terminal-Manifest-Inhaltsfehler was Ihnen während des Trainings Fehler bei der Überprüfung von JSON-Zeilen außerhalb des Terminals begegnet ist. Sie müssen Fehler im Manifest beheben. Außerdem empfehlen wir die Behebung von -JSON-Line-Fehlern, die sich nicht auf das Terminal beziehen. Informationen zu bestimmten Fehlern finden Sie unter Fehler bei der Überprüfung von JSON-Zeilen ohne Terminal undTerminal-Manifest-Inhaltsfehler.

Sie können Korrekturen an dem für das Training verwendeten Trainings- oder Testdatensatz vornehmen. Alternativ können Sie die Korrekturen in den Manifestdateien für die Trainings- und Testvalidierung vornehmen und sie zum Trainieren des Modells verwenden.

Nachdem Sie Ihre Korrekturen vorgenommen haben, müssen Sie die aktualisierten Manifeste (e) importieren und das Modell neu trainieren. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

Im folgenden Verfahren wird gezeigt, wie Sie mithilfe der Manifestübersicht Fehler im Terminal-Manifest beheben. Das Verfahren zeigt Ihnen auch, wie Sie JSON-Line-Fehler in den Trainings- und Testvalidierungsmanifesten finden und beheben können.

So beheben Sie Amazon Rekognition Custom Labels Trainingsfehler

1. Laden Sie die Dateien mit den Validierungsergebnissen herunter. Die Dateinamen sind training_manifest_with_validation.json, testing_manifest_with_validation.json und manifest_summary.json. Weitere Informationen finden Sie unter Abrufen der Validierungsergebnisse.
2. Öffnen Sie die Manifest-Zusammenfassungsdatei (manifest_summary.json).
3. Korrigieren Sie alle Fehler in der Manifestzusammenfassung. Weitere Informationen finden Sie unter Verstehen der Manifestübersicht.
4. In der Manifestzusammenfassung iterieren Sie das `error_line_indices` Array in `training` und korrigieren Sie die Fehler `training_manifest_with_validation.json` an den entsprechenden JSON-Zeilennummern. Weitere Informationen finden Sie unter the section called “Grundlegendes zu den Ergebnissen der Trainings- und Testvalidierung”.

5. Iterieren Sie das `error_line_indices` Array in `testing` und korrigieren Sie die Fehler `testing_manifest_with_validation.json` an den entsprechenden JSON-Zeilennummern.
6. Trainieren Sie das Modell erneut, indem Sie die Validierungsmanifestdateien als Trainings- und Testdatensätze verwenden. Weitere Informationen finden Sie unter the section called “Ein Model ausbilden”.

Wenn Sie das AWS SDK verwenden und sich dafür entscheiden, die Fehler in den Manifestdateien der Trainings- oder Testvalidierungsdaten zu beheben, verwenden Sie den Speicherort der Validierungsdatenmanifestdateien in den TestingDataEingabeparametern TrainingDataund CreateProjectVersion. Weitere Informationen finden Sie unter Ein Modell trainieren (SDK).

Priorität von JSON-Zeilenfehlern

Die folgenden JSON-Line-Fehler werden zuerst erkannt. Wenn einer dieser Fehler auftritt, wird die Überprüfung von JSON-Line-Fehlern gestoppt. Sie müssen diese Fehler beheben, bevor Sie die anderen JSON-Line-Fehler beheben können.

- FEHLENDER_QUELLENREFERENZ
- ERROR_INVALID_SOURCE_REF_FORMAT
- FEHLER: NO_LABEL_ATTRIBUTES
- ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT
- ERROR_INVALID_LABEL_ATTRIBUTE_METADAT_FORMAT
- ERROR_MISSING_BOUNDING_BOX_CONFIDENCE
- FEHLER_FEHLENDER_KLASSENKARTE_ID
- ERROR_INVALID_JSON_LINE

Fehler in der Terminal-Manifest-Datei

In diesem Thema wird die beschriebenFehler in der Terminal-Manifest-Datei. Manifestdateifehlern ist kein entsprechender Fehlercode zugeordnet. Die Manifeste der Überprüfungsergebnisse werden nicht erstellt, wenn ein Fehler in der Terminal-Manifestdatei auftritt. Weitere Informationen finden Sie unter Verstehen der Manifestübersicht. Terminal-Manifestfehler verhindern die Meldung vonFehler bei der Überprüfung von JSON-Zeilen ohne Terminal.

Die Manifest-Dateierweiterung oder der Inhalt sind ungültig.

Die Trainings- oder Testmanifestdatei hat keine Dateierweiterung oder ihr Inhalt ist ungültig.

Um den Fehler zu beheben Die Manifest-Dateierweiterung oder der Inhalt sind ungültig.

- Überprüfen Sie die folgenden möglichen Ursachen sowohl in den Trainings- als auch in den Testmanifestdateien.
  - In der Manifestdatei fehlt eine Dateierweiterung. Konventionell lautet die Dateierweiterung `.manifest`.
  - Der Amazon S3 S3-Bucket oder Schlüssel für die Manifestdatei konnte nicht gefunden werden.

Die Manifestdatei ist leer.

Die für das Training verwendete Trainings- oder Testmanifestdatei ist vorhanden, aber sie ist leer. Die Manifestdatei benötigt eine JSON-Zeile für jedes Bild, das Sie für Schulungen und Tests verwenden.

Um den Fehler zu beheben Die Manifestdatei ist leer.

1. Prüfen Sie, welche der Trainings- oder Testmanifeste leer sind.
2. Fügen Sie der leeren Manifestdatei JSON-Zeilen hinzu. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei. Alternativ, können Sie mit der -Konsole einen neuen Datensatz auch mit der -Konsole erstellen. Weitere Informationen finden Sie unter the section called “Erstellen von Datensätzen mit Bildern”.

Die Manifestdateigröße überschreitet die maximal unterstützte Größe.

Die Größe der Trainings- oder Testmanifestdatei (in Byte) ist zu groß. Weitere Informationen finden Sie unter Richtlinien und Kontingente in Amazon Rekognition Custom Labels. Eine Manifestdatei kann weniger als die maximale Anzahl von JSON-Zeilen haben und trotzdem die maximale Dateigröße überschreiten.

Sie können die Amazon Rekognition Custom Labels-Konsole nicht verwenden, um den Fehler zu beheben. Die Größe der Manifestdatei überschreitet die maximal unterstützte Größe.

Um den Fehler zu beheben Die Manifestdatei überschreitet die maximal unterstützte Größe.

1. Prüfen Sie, welche der Trainings- und Testmanifeste die maximale Dateigröße überschreiten.
2. Reduzieren Sie die Anzahl der JSON-Zeilen in den Manifestdateien, die zu groß sind. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

Die S3-Bucket-Berechtigungen sind falsch.

Amazon Rekognition Custom Labels hat keine Berechtigungen für einen oder mehrere der Buckets, die die Trainings- und Testmanifestdateien enthalten.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

Um den Fehler zu beheben Die S3-Bucket-Berechtigungen sind falsch.

- Überprüfen Sie die Berechtigungen für die Buckets, die die Trainings- und Testmanifeste enthalten. Weitere Informationen finden Sie unter Schritt 2: Einrichten von Amazon Rekognition Custom Labels-Konsolenberechtigungen.

In den S3-Ausgabe-Bucket kann nicht geschrieben werden.

Der Dienst kann die Trainingsausgabedateien nicht generieren.

Um den Fehler zu beheben: In den S3-Ausgabe-Bucket konnte nicht geschrieben werden.

- Vergewissern Sie sich, dass die Amazon S3 S3-Bucket-Informationen im OutputConfigEingabeparameter to CreateProjectVersionkorrekt sind.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

Terminal-Manifest-Inhaltsfehler

In diesem Thema wird der in der Manifestzusammenfassung Terminal-Manifest-Inhaltsfehler gemeldete Inhalt beschrieben. Die Manifestzusammenfassung enthält einen Fehlercode und eine Meldung für jeden erkannten Fehler. Weitere Informationen finden Sie unter Verstehen der

Manifestübersicht. Fehler im Terminal-Manifest verhindern nicht die Meldung vonFehler bei der Überprüfung von JSON-Zeilen außerhalb des Terminals.

FEHLER_ZU_VIELE_UNGÜLTIGE ZEILEN_IM_MANIFEST

Fehlermeldung

Die Manifestdatei enthält zu viele ungültige Zeilen.

Weitere Informationen

Ein `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST` Fehler tritt auf, wenn zu viele JSON-Zeilen ungültigen Inhalt enthalten.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um das Löschen eines `ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST` Fehlers nicht verwenden.

Um ERROR_TOO_MANY_INVALID_ROWS_IN_MANIFEST zu beheben

1. Überprüfen Sie das Manifest auf JSON-Line-Fehler. Weitere Informationen finden Sie unter Grundlegendes zu den Ergebnissen der Trainings- und Testvalidierung.
2. Fehlerhafte JSON-Zeilen korrigieren Weitere Informationen finden Sie unterFehler bei der Überprüfung von JSON-Zeilen ohne Terminal.

FEHLERBILDER IN MEHREREN S3_BUCKETS

Fehlermeldung

Die Manifestdatei enthält Bilder aus mehreren S3-Buckets.

Weitere Informationen

Ein Manifest kann nur auf Bilder verweisen, die in einem einzigen Bucket gespeichert sind. Jede JSON-Zeile speichert den Amazon S3 S3-Standort einer Image-Position im Wert von`source-ref`. Im folgenden Beispiel lautet der Bucket-Name my-bucket.

```
"source-ref": "s3://my-bucket/images/sunrise.png"
```

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

Zu reparieren **`ERROR_IMAGES_IN_MULTIPLE_S3_BUCKETS`**

- Stellen Sie sicher, dass sich alle Ihre Bilder im selben Amazon S3 S3-Bucket befinden und dass der Wert von `source-ref` in jeder JSON-Zeile auf den Bucket verweist, in dem Ihre Bilder gespeichert sind. Wählen Sie alternativ einen bevorzugten Amazon S3 S3-Bucket und entfernen Sie die JSON-Zeilen, die `source-ref` nicht auf Ihren bevorzugten Bucket verweisen.

ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET

Fehlermeldung

Die Berechtigungen für den S3-Bucket für Bilder sind ungültig.

Weitere Informationen

Die Berechtigungen für den Amazon S3 S3-Bucket, der die Images enthält, sind falsch.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

Zu reparieren **`ERROR_INVALID_PERMISSIONS_IMAGES_S3_BUCKET`**

- Überprüfen Sie die Berechtigungen des Buckets, der die Bilder enthält. Der Wert von `source-ref` für ein Bild enthält die Bucket-Position.

ERROR_INVALID_IMAGES_S3_BUCKET_OWNER

Fehlermeldung

Ungültige Besitzer-ID für den S3-Bucket für Bilder.

Weitere Informationen

Der Besitzer des Buckets, der die Trainings- oder Testbilder enthält, unterscheidet sich vom Besitzer des Buckets, der das Schulungs- oder Testmanifest enthält. Sie können folgenden Befehl verwenden, um den Besitzer eines Buckets zu finden.

```
aws s3api get-bucket-acl --bucket bucket name
```

Das `OWNER ID` muss mit den Buckets übereinstimmen, in denen die Bilder und Manifestdateien gespeichert sind.

Um ERROR_INVALID_IMAGES_S3_BUCKET_OWNER zu beheben

1. Wählen Sie den gewünschten Besitzer der Trainings-, Test-, Output- und Image-Buckets aus. Der Besitzer muss über die erforderlichen Berechtigungen verfügen, um Amazon Rekognition Custom Labels verwenden zu können.
2. Erstellen Sie für jeden Bucket, der derzeit nicht dem gewünschten Besitzer gehört, einen neuen Amazon S3 S3-Bucket, der dem bevorzugten Besitzer gehört.
3. Kopieren Sie den Inhalt des alten Buckets in den neuen Bucket. Weitere Informationen finden Sie unter Wie kann ich Objekte zwischen Amazon S3 S3-Buckets kopieren? .

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

FEHLER_UNGENÜGEND_BILDER_PER_LABEL_FÜR_AUTOSPLIT

Fehlermeldung

Die Manifestdatei enthält nicht genügend beschriftete Bilder pro Label, um eine automatische Aufteilung durchzuführen.

Weitere Informationen

Während des Modelltrainings können Sie einen Testdatensatz erstellen, indem Sie 20% der Bilder aus dem Trainingsdatensatz verwenden. ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT tritt auf, wenn nicht genügend Bilder vorhanden sind, um einen akzeptablen Testdatensatz zu erstellen.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

Um ERROR_INSUFFICIENT_IMAGES_PER_LABEL_FOR_AUTOSPLIT zu beheben

- Füge deinem Trainingsdatensatz weitere beschriftete Bilder hinzu. Sie können Bilder in der Amazon Rekognition Custom Labels-Konsole hinzufügen, indem Sie Bilder zum Trainingsdatensatz hinzufügen oder indem Sie Ihrem Trainingsmanifest JSON-Zeilen hinzufügen. Weitere Informationen finden Sie unter Verwalten von Datensätzen.

FEHLERMANIFEST_ZU_WEN_BEZEICHNUNGEN

Fehlermeldung

Die Manifestdatei hat zu wenige Bezeichnungen.

Weitere Informationen

Trainings- und Testdatensätze verfügen über eine erforderliche Mindestanzahl von Labels. Das Minimum hängt davon ab, ob der Datensatz ein Modell trainiert/testet, um Beschriftungen auf Bildebene zu erkennen (Klassifizierung), oder ob das Modell Objektpositionen erkennt. Wenn der Trainingsdatensatz aufgeteilt wird, um einen Testdatensatz zu erstellen, wird die Anzahl der Labels im Datensatz bestimmt, nachdem der Trainingsdatensatz aufgeteilt wurde. Weitere Informationen finden Sie unter Richtlinien und Kontingente in Amazon Rekognition Custom Labels.

Um ERROR_MANIFEST_TOO_FEW_LABELS zu beheben (Konsole)

1. Fügen Sie dem Datensatz weitere neue Labels hinzu. Weitere Informationen finden Sie unter Labels verwalten.
2. Fügen Sie die neuen Labels zu Bildern im Datensatz hinzu. Wenn Ihr Modell Beschriftungen auf Bildebene erkennt, finden Sie weitere Informationen unter. Einem Bild Labels auf Bildebene zuweisen Wenn Ihr Modell Objektpositionen erkennt, finden Sie weitere Informationen unterthe section called “Objekte mit Begrenzungsrahmen mit Labels versehen”.

Um ERROR_MANIFEST_TOO_FEW_LABELS zu beheben (JSON-Zeile)

- Fügen Sie JSON-Zeilen für neue Bilder mit neuen Labels hinzu. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei. Wenn Ihr Modell Beschriftungen auf Bildebene erkennt, fügen Sie dem `class-name` Feld neue Labelnamen hinzu. Die Bezeichnung für das folgende Bild lautet beispielsweise Sonnenaufgang.

```
{
    "source-ref": "s3://bucket/images/sunrise.png",
    "testdataset-classification_Sunrise": 1,
    "testdataset-classification_Sunrise-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/testdataset-classification_Sunrise",
        "class-name": "Sunrise",
        "human-annotated": "yes",
```

```
        "creation-date": "2018-10-18T22:18:13.527256",
        "type": "groundtruth/image-classification"
    }
}
```

Wenn Ihr Modell Objektpositionen erkennt, fügen Sie dem neue Beschriftungen hinzu`class-map`, wie im folgenden Beispiel gezeigt.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
```

```
  "job-name": "my job"
 }
}
```

Sie müssen die Klassenzuordnungstabelle den Bounding-Box-Anmerkungen zuordnen. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien.

FEHLERMANIFEST_ZU_VIELE_BEZEICHNUNGEN

Fehlermeldung

Die Manifestdatei hat zu viele Bezeichnungen.

Weitere Informationen

Die Anzahl der eindeutigen Labels im Manifest (Datensatz) überschreitet den zulässigen Grenzwert. Wenn der Trainingsdatensatz aufgeteilt wird, um einen Testdatensatz zu erstellen, wird die Anzahl der Labels nach der Aufteilung bestimmt.

Um ERROR_MANIFEST_TOO_MANY_LABELS zu beheben (Konsole)

- Entfernen Sie Labels aus dem Datensatz. Weitere Informationen finden Sie unter Labels verwalten. Die Beschriftungen werden automatisch von den Bildern und Begrenzungsfeldern in Ihrem Datensatz entfernt.

Um ERROR_MANIFEST_TOO_MANY_LABELS (JSON-Zeile) zu beheben

- Manifeste mit JSON-Zeilen auf Bildebene — Wenn das Bild ein einzelnes Label hat, entfernen Sie die JSON-Zeilen für Bilder, die das gewünschte Label verwenden. Wenn die JSON-Zeile mehrere Labels enthält, entfernen Sie nur das JSON-Objekt für das gewünschte Label. Weitere Informationen finden Sie unter Hinzufügen mehrerer Labels auf Bildebene zu einem Bild.

  Manifeste mit JSON-Linien an der Objektposition — Entfernen Sie den Begrenzungsrahmen und die zugehörigen Labelinformationen für das Label, das Sie entfernen möchten. Tun Sie dies für jede JSON-Zeile, die das gewünschte Label enthält. Sie müssen das Label aus dem `class-map` Array und den entsprechenden Objekten im `annotations` Array `objects` und entfernen. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien.

FEHLER_UNGENÜGEND_LABEL-ÜBERLAPPUNG

Fehlermeldung

Weniger als {}% Label-Überlappung zwischen den Trainings- und Testmanifestdateien.

Weitere Informationen

Es gibt weniger als 50% Überlappung zwischen den Labelnamen des Test-Datensatzes und den Labelnamen des Trainingsdatensatzes.

Um ERROR_INSUFFICIENT_LABEL_OVERLAP zu beheben (Konsole)

- Entfernen Sie die Labels aus dem Trainingsdatensatz. Alternativ können Sie Ihrem Testdatensatz weitere gebräuchliche Labels hinzufügen. Weitere Informationen finden Sie unter Labels verwalten. Die Beschriftungen werden automatisch von den Bildern und Begrenzungsfeldern in Ihrem Datensatz entfernt.

Um ERROR_INSUFFICIENT_LABEL_OVERLAP zu beheben, indem Labels aus dem Trainingsdatensatz entfernt werden (JSON-Zeile)

- Manifeste mit JSON-Zeilen auf Bildebene — Wenn das Bild ein einzelnes Label hat, entfernen Sie die JSON-Zeile für das Bild, das das gewünschte Label verwendet. Wenn die JSON-Zeile mehrere Labels enthält, entfernen Sie nur das JSON-Objekt für das gewünschte Label. Weitere Informationen finden Sie unter Hinzufügen mehrerer Labels auf Bildebene zu einem Bild. Tun Sie dies für jede JSON-Zeile im Manifest, die das Label enthält, das Sie entfernen möchten.

  Manifeste mit JSON-Linien an der Objektposition — Entfernen Sie den Begrenzungsrahmen und die zugehörigen Labelinformationen für das Label, das Sie entfernen möchten. Tun Sie dies für jede JSON-Zeile, die das gewünschte Label enthält. Sie müssen das Label aus dem `class-map` Array und den entsprechenden Objekten im `annotations` Array `objects` und entfernen. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien.

Um ERROR_INSUFFICIENT_LABEL_OVERLAP zu beheben, indem Sie dem Testdatensatz allgemeine Labels hinzufügen (JSON-Zeile)

- Fügen Sie dem Testdatensatz JSON-Zeilen hinzu, die Bilder enthalten, die mit Labels beschriftet sind, die sich bereits im Trainingsdatensatz befinden. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

FEHLERMANIFEST_ZU_WENDEN_NUTZBARE_LABELS

Fehlermeldung

Die Manifestdatei enthält zu wenige verwendbare Labels.

Weitere Informationen

Ein Trainingsmanifest kann JSON-Zeilen im Labelformat auf Bildebene und im Objektpositionsformat enthalten. Abhängig vom Typ der JSON-Linien im Trainingsmanifest erstellt Amazon Rekognition Custom Labels entweder ein Modell, das Beschriftungen auf Bildebene erkennt, oder ein Modell, das Objektpositionen erkennt. Amazon Rekognition Custom Labels filtert gültige JSON-Datensätze für JSON-Zeilen heraus, die nicht im ausgewählten Format vorliegen. ERROR_MANIFEST_TOO_FEW_USABLE_LABELS tritt auf, wenn die Anzahl der Labels im gewählten Modelltyp-Manifest nicht ausreicht, um das Modell zu trainieren.

Um ein Modell zu trainieren, das Beschriftungen auf Bildebene erkennt, ist mindestens 1 Etikett erforderlich. Es sind mindestens 2 Labels erforderlich, um ein Modell zu trainieren, das Objekte lokalisiert.

Um ERROR_MANIFEST_TOO_FEW_USABLE_LABELS zu beheben (Konsole)

1. Markieren Sie das `use_case` Feld in der Manifestzusammenfassung.
2. Fügen Sie dem Trainingsdatensatz weitere Labels für den Anwendungsfall (Bildebene oder Objektlokalisierung) hinzu, der dem Wert von entsprichtuse_case. Weitere Informationen finden Sie unter Labels verwalten. Die Beschriftungen werden automatisch von den Bildern und Begrenzungsfeldern in Ihrem Datensatz entfernt.

Um ERROR_MANIFEST_TOO_FEW_USABLE_LABELS (JSON-Zeile) zu beheben

1. Markieren Sie das `use_case` Feld in der Manifestzusammenfassung.

2. Fügen Sie dem Trainingsdatensatz weitere Labels für den Anwendungsfall (Bildebene oder Objektlokalisierung) hinzu, der dem Wert von entsprichtuse_case. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

FEHLER_UNGENÜGEND_VERWENDBARE_LABEL-ÜBERLAPPUNG

Fehlermeldung

Weniger als {}% verwendbares Label überschneiden sich zwischen den Trainings- und Testmanifestdateien.

Weitere Informationen

Ein Trainingsmanifest kann JSON-Zeilen im Labelformat auf Bildebene und im Objektpositionsformat enthalten. Abhängig von den Formaten im Trainingsmanifest entscheidet sich Amazon Rekognition Custom Labels dafür, ein Modell zu erstellen, das Beschriftungen auf Bildebene erkennt, oder ein Modell, das Objektpositionen erkennt. Amazon Rekognition Custom Labels verwendet keine gültigen JSON-Datensätze für JSON-Zeilen, die nicht im ausgewählten Modellformat vorliegen. ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP tritt auf, wenn sich die verwendeten Test- und Trainingslabels zu weniger als 50% überschneiden.

Um ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP zu beheben (Konsole)

- Entfernen Sie die Labels aus dem Trainingsdatensatz. Alternativ können Sie Ihrem Testdatensatz weitere gebräuchliche Labels hinzufügen. Weitere Informationen finden Sie unter Labels verwalten. Die Beschriftungen werden automatisch von den Bildern und Begrenzungsfeldern in Ihrem Datensatz entfernt.

Um ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP zu beheben, indem Labels aus dem Trainingsdatensatz entfernt werden (JSON-Zeile)

- Datensätze, die zur Erkennung von Labels auf Bildebene verwendet werden — Wenn das Bild ein einzelnes Label hat, entfernen Sie die JSON-Zeile für das Bild, das das gewünschte Label verwendet. Wenn die JSON-Zeile mehrere Labels enthält, entfernen Sie nur das JSON-Objekt für das gewünschte Label. Weitere Informationen finden Sie unter Hinzufügen mehrerer Labels auf Bildebene zu einem Bild. Tun Sie dies für jede JSON-Zeile im Manifest, die das Label enthält, das Sie entfernen möchten.

Datensätze, die zur Erkennung von Objektpositionen verwendet werden — Entfernen Sie den Begrenzungsrahmen und die zugehörigen Labelinformationen für das Label, das Sie entfernen möchten. Tun Sie dies für jede JSON-Zeile, die das gewünschte Label enthält. Sie müssen das Label aus dem `class-map` Array und den entsprechenden Objekten im `annotations` Array `objects` und entfernen. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien.

Um ERROR_INSUFFICIENT_USABLE_LABEL_OVERLAP zu beheben, indem Sie dem Testdatensatz allgemeine Labels hinzufügen (JSON-Zeile)

- Fügen Sie dem Testdatensatz JSON-Zeilen hinzu, die Bilder enthalten, die mit Labels beschriftet sind, die sich bereits im Trainingsdatensatz befinden. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

ERROR_FAILED_IMAGES_S3_COPY

Fehlermeldung

Bilder konnten nicht aus dem S3-Bucket kopiert werden.

Weitere Informationen

Der Dienst konnte keines der Bilder in Ihrem Datensatz kopieren.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

Um ERROR_FAILED_IMAGES_S3_COPY zu beheben

1. Überprüfe die Berechtigungen deiner Bilder.
2. Wenn Sie es verwendenAWS KMS, überprüfen Sie die Bucket-Richtlinie. Weitere Informationen finden Sie unter Entschlüsseln von Dateien verschlüsselt mit AWS Key Management Service.

Die Manifestdatei enthält zu viele Terminalfehler.

Es gibt zu viele JSON-Zeilen mit Terminalinhaltsfehlern.

Zu reparieren **`ERROR_TOO_MANY_RECORDS_IN_ERROR`**

- Reduzieren Sie die Anzahl der JSON-Zeilen (Bilder) mit Terminalinhaltsfehlern. Weitere Informationen finden Sie unter Terminal-Manifest-Inhaltsfehler.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

Fehler bei der Überprüfung von JSON-Zeilen ohne Terminal

In diesem Thema werden die von Amazon Rekognition Custom Labels während des Trainings gemeldeten Fehler bei der JSON-Zeilenüberprüfung aufgeführt, die nicht vom Terminal stammen. Die Fehler werden im Manifest für die Trainings- und Testvalidierung gemeldet. Weitere Informationen finden Sie unter Grundlegendes zu den Ergebnissen der Trainings- und Testvalidierung. Sie können einen JSON-Line-Fehler beheben, der kein Terminal ist, indem Sie die JSON-Zeile in der Trainings- oder Testmanifestdatei aktualisieren. Sie können die JSON-Zeile auch aus dem Manifest entfernen, dies kann jedoch die Qualität Ihres Modells beeinträchtigen. Wenn viele Validierungsfehler außerhalb des Terminals auftreten, ist es möglicherweise einfacher, die Manifestdatei neu zu erstellen. Überprüfungsfehler treten in der Regel in manuell erstellten Manifestdateien auf. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei. Hinweise zur Behebung von Überprüfungsfehlern finden Sie unterBehebung von Trainingsfehlern. Einige Fehler können mithilfe der Amazon Rekognition Custom Labels-Konsole behoben werden.

FEHLER_FEHLENDER_QUELLREFERENZ

Fehlermeldung

Der Quellrefer-Schlüssel fehlt.

Weitere Informationen

Das `source-ref` Feld JSON Line gibt den Amazon S3 S3-Speicherort eines Images an. Dieser Fehler tritt auf, wenn der `source-ref` Schlüssel fehlt oder falsch geschrieben ist. Dieser Fehler tritt normalerweise in manuell erstellten Manifestdateien auf. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

Zu reparieren **`ERROR_MISSING_SOURCE_REF`**

1. Prüfen Sie, ob der `source-ref` Schlüssel vorhanden und richtig geschrieben ist. Ein vollständiger `source-ref` Schlüssel und Wert ähnelt dem Folgen. ist`"source-ref": "s3://bucket/path/image"`.
2. Aktualisieren Sie den `source-ref` Schlüssel in der JSON-Zeile. Alternativ können Sie die JSON-Zeile aus der Manifestdatei entfernen.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

ERROR_INVALID_SOURCE_REF_FORMAT

Fehlermeldung

Das Format des Quellreferen-Werts ist ungültig.

Weitere Informationen

Der `source-ref` Schlüssel ist in der JSON-Zeile vorhanden, aber das Schema des Amazon S3 S3-Pfads ist falsch. Der Pfad ist beispielsweise `https://....` anstelle von`S3://....`. Ein ERROR_INVALID_SOURCE_REF_FORMAT-Fehler tritt normalerweise in manuell erstellten Manifestdateien auf. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

Zu reparieren **`ERROR_INVALID_SOURCE_REF_FORMAT`**

1. Überprüfen Sie, ob das Schema vorhanden ist`"source-ref": "s3://bucket/path/image"`. Zum Beispiel `"source-ref": "s3://custom-labels-console-us-east-1-1111111111/images/000000242287.jpg"`.
2. Aktualisieren oder entfernen Sie die JSON-Zeile in der Manifestdatei.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um das Problem nicht zu beheben. `ERROR_INVALID_SOURCE_REF_FORMAT`

FEHLER: NO_LABEL_ATTRIBUTES

Fehlermeldung

Keine Labelattribute gefunden.

Weitere Informationen

Das Label-Attribut oder der `-metadata` Schlüsselname des Label-Attributs (oder beide) sind ungültig oder fehlen. `ERROR_NO_LABEL_ATTRIBUTES`Tritt im folgenden Beispiel immer dann auf, wenn der `bounding-box` oder `bounding-box-metadata` -Schlüssel (oder beide) fehlt. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
```

```
  }
}
```

In einer manuell erstellten Manifestdatei tritt normalerweise ein `ERROR_NO_LABEL_ATTRIBUTES` Fehler auf. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

Zu reparieren **`ERROR_NO_LABEL_ATTRIBUTES`**

1. Vergewissern Sie sich, dass die `-metadata` Schlüssel zur Label-Attribut-ID und zur Label-Attribut-ID vorhanden sind und dass die Schlüsselnamen richtig geschrieben sind.
2. Aktualisieren oder entfernen Sie die JSON-Zeile in der Manifestdatei.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um das Problem zu machen. `ERROR_NO_LABEL_ATTRIBUTES`

ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT

Fehlermeldung

Das Format des Label-Attributs {} ist ungültig.

Weitere Informationen

Das Schema für den Label-Attributschlüssel fehlt oder ist ungültig. Ein ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT-Fehler tritt normalerweise in manuell erstellten Manifestdateien auf. Weitere Informationen finden Sie unter. Erstellen einer Manifestdatei

Zu reparieren **`ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT`**

1. Überprüfen Sie, ob der JSON-Zeilenabschnitt für den Label-Attributschlüssel korrekt ist. Im folgenden Beispiel für die Objektposition müssen die `annotations` Objekte `image_size` und korrekt sein. Der Label-Attributschlüssel wird benanntbounding-box.

   ```
   "bounding-box": {
    "image_size": [{
     "width": 640,
     "height": 480,
     "depth": 3
    }],
    "annotations": [{
     "class_id": 1,
   ```

```
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }, {
  "class_id": 0,
  "top": 65,
  "left": 86,
  "width": 220,
  "height": 334
 }]
},
```

2. Aktualisieren oder entfernen Sie die JSON-Zeile in der Manifestdatei.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

ERROR_INVALID_LABEL_ATTRIBUTE_METADAT_FORMAT

Fehlermeldung

Das Format der Metadaten des Label-Attributs ist ungültig.

Weitere Informationen

Das Schema für den Metadatenschlüssel des Label-Attributs fehlt oder ist ungültig. Ein ERROR_INVALID_LABEL_ATTRIBUTE_METADATA_FORMAT-Fehler tritt normalerweise in manuell erstellten Manifestdateien auf. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

Zu reparieren **`ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT`**

1. Stellen Sie sicher, dass das JSON-Zeilenschema für den Metadatenschlüssel des Label-Attributs dem folgenden Beispiel ähnelt. Der Metadatenschlüssel des Label-Attributs wird benanntbounding-box-metadata.

```
"bounding-box-metadata": {
 "objects": [{
  "confidence": 1
 }, {
  "confidence": 1
 }],
```

```
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
```

2. Aktualisieren oder entfernen Sie die JSON-Zeile in der Manifestdatei.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

ERROR_NO_VALID_LABEL_ATTRIBUTES

Fehlermeldung

Keine gültigen Labelattribute gefunden.

Weitere Informationen

In der JSON-Zeile wurden keine gültigen Label-Attribute gefunden. Amazon Rekognition Custom Labels überprüft sowohl das Label-Attribut als auch die Label-Attribut-ID. Ein ERROR_INVALID_LABEL_ATTRIBUTE_FORMAT-Fehler tritt normalerweise in manuell erstellten Manifestdateien auf. Weitere Informationen finden Sie unter. Erstellen einer Manifestdatei

Wenn eine JSON-Zeile nicht in einem unterstützten SageMaker Manifestformat vorliegt, markiert Amazon Rekognition Custom Labels die JSON-Zeile als ungültig und es wird ein `ERROR_NO_VALID_LABEL_ATTRIBUTES` Fehler gemeldet. Derzeit unterstützt Amazon Rekognition Custom Labels die Formate Klassifikationsjob und Bounding Box. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

Zu reparieren **`ERROR_NO_VALID_LABEL_ATTRIBUTES`**

1. Vergewissern Sie sich, dass der JSON-Code für den Label-Attributschlüssel und die Metadaten des Label-Attributs korrekt ist.
2. Aktualisieren oder entfernen Sie die JSON-Zeile in der Manifestdatei. Weitere Informationen finden Sie unter the section called “Erstellen einer Manifestdatei”.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

ERROR_MISSING_BOUNDING_BOX_CONFIDENCE

Fehlermeldung

Ein oder mehrere Begrenzungsrahmen weisen einen fehlenden Konfidenzwert auf.

Weitere Informationen

Der Konfidenzschlüssel fehlt für einen oder mehrere Begrenzungsrahmen für die Objektposition. Der Vertrauensschlüssel für eine Bounding Box befindet sich in den Metadaten des L-Attributs angeben, wie im folgenden Beispiel gezeigt. Ein ERROR_MISSING_BOUNDING_BOX_CONFIDENCE-Fehler tritt normalerweise in manuell erstellten Manifestdateien auf. Weitere Informationen finden Sie unter the section called “Objektlokalisierung in Manifestdateien”.

```
"bounding-box-metadata": {
 "objects": [{
  "confidence": 1
 }, {
  "confidence": 1
 }],
```

Zu reparieren **`ERROR_MISSING_BOUNDING_BOX_CONFIDENCE`**

1. Stellen Sie sicher, dass das `objects` Array im Label-Attribut dieselbe Anzahl von Konfidenzschlüsseln enthält wie Objekte im `annotations` Label-Attribut-Array.
2. Aktualisieren oder entfernen Sie die JSON-Zeile in der Manifestdatei.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

FEHLER_FEHLENDER_KLASSENKARTE_ID

Fehlermeldung

Eine oder mehrere Klassen-IDs fehlen in der Klassenzuordnung.

Weitere Informationen

Das Objekt `class_id` in einer Annotation (Bounding Box) hat keinen passenden Eintrag in der Metadatenklasse map (`class-map`) für das Labelattribut. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien. Ein ERROR_MISSING_CLASS_MAP_ID-Fehler tritt normalerweise in manuell erstellten Manifestdateien auf.

Um ERROR_MISSING_CLASS_MAP_ID zu beheben

1. Stellen Sie sicher, dass der `class_id` Wert in jedem Annotationsobjekt (Bounding Box) einen entsprechenden Wert im `class-map` Array hat, wie im folgenden Beispiel gezeigt. Das `annotations` `class_map` Array und das Array sollten die gleiche Anzahl von Elementen haben.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [{
   "confidence": 1
  }, {
   "confidence": 1
  }],
```

```
  "class-map": {
   "0": "Echo",
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
}
```

2. Aktualisieren oder entfernen Sie die JSON-Zeile in der Manifestdatei.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

ERROR_INVALID_JSON_LINE

Fehlermeldung

Die JSON-Zeile hat ein ungültiges Format.

Weitere Informationen

In der JSON-Zeile wurde ein unerwartetes Zeichen gefunden. Die JSON-Zeile wird durch eine neue JSON-Zeile ersetzt, die nur die Fehlerinformationen enthält. Ein ERROR_INVALID_JSON_LINE-Fehler tritt normalerweise in manuell erstellten Manifestdateien auf. Weitere Informationen finden Sie unter the section called “Objektlokalisierung in Manifestdateien”.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

Zu reparieren **`ERROR_INVALID_JSON_LINE`**

1. Öffnen Sie die Manifestdatei und navigieren Sie zu der JSON-Zeile, in der der Fehler ERROR_INVALID_JSON_LINE auftritt.
2. Stellen Sie sicher, dass die JSON-Zeile keine ungültigen Zeichen enthält und dass keine erforderlichen `,` Zeichen `;` oder Zeichen fehlen.
3. Aktualisieren oder entfernen Sie die JSON-Zeile in der Manifestdatei.

FEHLER_UNGÜLTIGES BILD

Fehlermeldung

Das Bild ist ungültig. Überprüfen Sie den S3-Pfad und/oder die Bildeigenschaften.

Weitere Informationen

Die Datei, auf die verwiesen `source-ref` wird, ist kein gültiges Bild. Mögliche Ursachen sind das Bildseitenverhältnis, die Größe des Bildes und das Bildformat.

Weitere Informationen finden Sie unter Richtlinien und Kontingente.

Zu reparieren **`ERROR_INVALID_IMAGE`**

1. Überprüfen Sie Folgendes.
   - Das Seitenverhältnis des Bildes beträgt weniger als 20:1.
   - Die Größe des Images ist größer als 15 MB
   - Das Bild ist im PNG- oder JPEG-Format.
   - Der Pfad zum Abbild in `source-ref` ist korrekt.
   - Die minimale Bildgröße des Bildes ist größer 64 Pixel x 64 Pixel.
   - Die maximale Bildgröße des Bildes beträgt weniger als 4096 Pixel x 4096 Pixel.
2. Aktualisieren oder entfernen Sie die JSON-Zeile in der Manifestdatei.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

FEHLER_UNGÜLTIGE_BILDDIMENSION

Fehlermeldung

Die Bildabmessungen entsprechen nicht den zulässigen Abmessungen.

Weitere Informationen

Das Bild, auf das verwiesen wird, entspricht `source-ref` nicht den zulässigen Bildabmessungen. Die Mindestabmessung beträgt 64 Pixel. Die maximale Abmessung beträgt 4096 Pixel. `ERROR_INVALID_IMAGE_DIMENSION`wird für Bilder mit Begrenzungsrahmen gemeldet.

Weitere Informationen finden Sie unter Richtlinien und Kontingente.

Um das Problem zu beheben **`ERROR_INVALID_IMAGE_DIMENSION`** (Konsole)

1. Aktualisieren Sie das Bild im Amazon S3 S3-Bucket mit Dimensionen, die Amazon Rekognition Custom Labels verarbeiten kann.
2. Führen Sie in der Amazon Rekognition Custom Labels -Konsole die folgenden Schritte aus:
   a. Entfernen Sie die vorhandenen Begrenzungsrahmen aus dem Bild.
   b. Fügen Sie die Begrenzungsrahmen erneut zum Bild hinzu.
   c. Speichern Sie Ihre Änderungen.

   Weitere Informationen finden Sie unter Objekte mit Begrenzungsrahmen mit Labels versehen.

Zum Reparieren **`ERROR_INVALID_IMAGE_DIMENSION`** (SDK)

1. Aktualisieren Sie das Bild im Amazon S3 S3-Bucket mit Dimensionen, die Amazon Rekognition Custom Labels verarbeiten kann.
2. Rufen Sie die vorhandene JSON-Zeile für das Bild ab ListDatasetEntries. Geben Sie für den `SourceRefContains` Eingabeparameter den Amazon S3 S3-Speicherort und den Dateinamen des Images an.
3. Rufen Sie die JSON-Zeile für das Bild auf UpdateDatasetEntriesund geben Sie sie an. Vergewissern Sie sich, dass der Wert von mit dem Image-Speicherort im Amazon S3 S3-Bucket `source-ref` übereinstimmt. Aktualisieren Sie die Anmerkungen des Begrenzungsrahmens, sodass sie den Abmessungen des Begrenzungsrahmens entsprechen, die für das aktualisierte Bild erforderlich sind.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [{
   "class_id": 1,
   "top": 251,
```

```
    "left": 399,
    "width": 155,
    "height": 101
   }, {
    "class_id": 0,
    "top": 65,
    "left": 86,
    "width": 220,
    "height": 334
   }]
  },
  "bounding-box-metadata": {
   "objects": [{
    "confidence": 1
   }, {
    "confidence": 1
   }],
   "class-map": {
    "0": "Echo",
    "1": "Echo Dot"
   },
   "type": "groundtruth/object-detection",
   "human-annotated": "yes",
   "creation-date": "2013-11-18T02:53:27",
   "job-name": "my job"
  }
}
```

ERROR_INVALID_BOUNDING_BOX

Fehlermeldung

Die Begrenzungsbox enthält Off-Frame-Werte.

Weitere Informationen

Die Informationen zum Begrenzungsrahmen geben ein Bild an, das sich entweder außerhalb des Bildrahmens befindet oder negative Werte enthält.

Weitere Informationen finden Sie unter Richtlinien und Kontingente.

Zu reparieren **`ERROR_INVALID_BOUNDING_BOX`**

1. Überprüfen Sie die Werte der Begrenzungsfelder im `annotations` Array.

```
"bounding-box": {
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
 "annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }]
},
```

2. Aktualisieren Sie die JSON-Zeile oder entfernen Sie sie alternativ aus der Manifestdatei.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

ERROR_NO_VALID_ANNOTATIONS

Fehlermeldung

Keine gültigen Anmerkungen gefunden.

Weitere Informationen

Keines der Annotationsobjekte in der JSON-Zeile enthält gültige Bounding-Box-Informationen.

Zu reparieren **`ERROR_NO_VALID_ANNOTATIONS`**

1. Aktualisieren Sie das `annotations` Array, sodass es gültige Bounding-Box-Objekte enthält. Überprüfen Sie außerdem, ob die entsprechenden Begrenzungsfeld-Informationen (`confidenceundclass_map`) in den Metadaten des Label-Attributs korrekt sind. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien.

```
{
```

```
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [
     {
   "class_id": 1,    #annotation object
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
 },
 "bounding-box-metadata": {
  "objects": [
  >{
   "confidence": 1          #confidence  object
  },
        {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",    #label
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
}
```

2. Aktualisieren Sie die JSON-Zeile oder entfernen Sie sie alternativ aus der Manifestdatei.

Sie können die Amazon Rekognition Custom Labels Konsole nicht verwenden, um diesen Fehler nicht zu verwenden.

ERROR_BOUNDING_BOX_ZU KLEIN

Fehlermeldung

Die Höhe und Breite des Begrenzungsrahmens sind zu klein.

Weitere Informationen

Die Abmessungen des Begrenzungsrahmens (Höhe und Breite) müssen größer als 1 x 1 Pixel sein.

Während des Trainings ändert Amazon Rekognition Custom Labels die Größe eines Bilds, wenn eine der Abmessungen größer als 1280 Pixel ist (die Quellbilder sind davon nicht betroffen). Die resultierenden Höhen und Breiten des Begrenzungsrahmens müssen größer als 1 x 1 Pixel sein. Eine Bounding-Box-Position wird im `annotations` Array einer Objektposition in der JSON-Zeile gespeichert. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien

```
"bounding-box": {
 "image_size": [{
  "width": 640,
  "height": 480,
  "depth": 3
 }],
 "annotations": [{
  "class_id": 1,
  "top": 251,
  "left": 399,
  "width": 155,
  "height": 101
 }]
},
```

Die Fehlerinformationen werden dem Annotationsobjekt hinzugefügt.

Um ERROR_BOUNDING_BOX_TOO_SMALL zu beheben

- Wählen Sie eine der folgenden Optionen.
  - Erhöhen Sie die Größe der zu kleinen Begrenzungsrahmen.

- Entfernen Sie zu kleine Begrenzungsrahmen. Hinweise zum Entfernen eines Begrenzungsrahmens finden Sie unterFEHLER ZU VIELE_BEGRENZTEN_BOXEN.
- Entfernen Sie das Bild (JSON-Zeile) aus dem Manifest.

FEHLER ZU VIELE_BEGRENZTEN_BOXEN

Fehlermeldung

Es gibt mehr Begrenzungsfelder als das zulässige Maximum.

Weitere Informationen

Es gibt mehr Begrenzungsfelder als das zulässige Limit (50). Sie können überschüssige Begrenzungsfelder in der Amazon Rekognition Custom Labels-Konsole entfernen oder sie aus der JSON-Zeile entfernen.

Um das Problem zu beheben **`ERROR_TOO_MANY_BOUNDING_BOXES`** (Konsole).

1. Entscheiden Sie, welche Begrenzungsfelder entfernt werden sollen.
2. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
3. Wählen Sie Benutzerdefinierte Labels verwenden aus.
4. Wählen Sie Get started (Erste Schritte) aus.
5. Wählen Sie im linken Navigationsbereich das Projekt aus, das den Datensatz enthält, den Sie verwenden möchten.
6. Wählen Sie im Abschnitt Datensätze den Datensatz aus, den Sie verwenden möchten.
7. Wählen Sie auf der Datensatz-Galerie-Seite die Option Kennzeichnung starten, um in den Labeling-Modus zu wechseln.
8. Wählen Sie das Bild aus, von dem Sie die Begrenzungsrahmen entfernen möchten.
9. Wählen Sie „Umgrenzungsrahmen zeichnen“.
10. Wählen Sie im Zeichenwerkzeug den Begrenzungsrahmen aus, den Sie löschen möchten.
11. Drücken Sie die Löschtaste auf Ihrer Tastatur, um das Begrenzungsfeld zu löschen.
12. Wiederholen Sie die vorherigen 2 Schritte, bis Sie genügend Begrenzungsrahmen gelöscht haben.
13. Wählen Sie Fertig

14. Wählen Sie Save changes (Änderungen speichern) aus, um die Änderungen zu speichern.
15. Wählen Sie Exit, um den Beschriftungsmodus zu verlassen.

Um ERROR_TOO_MANY_BOUNDING_BOXES (JSON-Zeile) zu beheben.

1. Öffnen Sie die Manifestdatei und navigieren Sie zu der JSON-Zeile, in der der Fehler ERROR_TOO_MANY_BOUNDING_BOXES auftritt.
2. Entfernen Sie Folgendes für jeden Begrenzungsrahmen, den Sie entfernen möchten.

   - Entfernen Sie das erforderliche `annotation` Objekt aus dem `annotations` Array.
   - Entfernen Sie das entsprechende `confidence` Objekt aus dem `objects` Array in den Metadaten des Label-Attributs.
   - Wenn es nicht mehr von anderen Umgrenzungsfeldern verwendet wird, entfernen Sie die Bezeichnung von der`class-map`.

   Verwenden Sie das folgende Beispiel, um zu ermitteln, welche Elemente entfernt werden sollen.

```
{
 "source-ref": "s3://custom-labels-bucket/images/IMG_1186.png",
 "bounding-box": {
  "image_size": [{
   "width": 640,
   "height": 480,
   "depth": 3
  }],
  "annotations": [
     {
   "class_id": 1,    #annotation object
   "top": 251,
   "left": 399,
   "width": 155,
   "height": 101
  }, {
   "class_id": 0,
   "top": 65,
   "left": 86,
   "width": 220,
   "height": 334
  }]
```

```
 },
 "bounding-box-metadata": {
  "objects": [
  >{
   "confidence": 1          #confidence  object
  },
        {
   "confidence": 1
  }],
  "class-map": {
   "0": "Echo",    #label
   "1": "Echo Dot"
  },
  "type": "groundtruth/object-detection",
  "human-annotated": "yes",
  "creation-date": "2018-10-18T22:18:13.527256",
  "job-name": "my job"
 }
}
```

WARNUNG: DATENSATZ OHNE ANMERKUNG

Warnmeldung

Der Datensatz ist nicht annotiert.

Weitere Informationen

Ein Bild, das mithilfe der Amazon Rekognition Custom Labels-Konsole zu einem Datensatz hinzugefügt wurde, wurde nicht beschriftet. Die JSON-Zeile für das Bild wird nicht für das Training verwendet.

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "warnings": [
        {
            "code": "WARNING_UNANNOTATED_RECORD",
            "message": "Record is unannotated."
        }
    ]
}
```

Um WARNING_UNANNOTATED_RECORD zu reparieren

- Benennen Sie das Bild mithilfe der Amazon Rekognition Custom Labels-Konsole. Detaillierte Anweisungen finden Sie unter Einem Bild Labels auf Bildebene zuweisen.

WARNUNG_NO_ANNOTATIONS

Warnmeldung

Es wurden keine Anmerkungen bereitgestellt.

Weitere Informationen

Eine JSON-Zeile im Format Object Localization enthält keine Bounding-Box-Informationen, obwohl sie von einem Menschen () annotiert wurde. `human-annotated = yes` Die JSON-Zeile ist gültig, wird aber nicht für das Training verwendet. Weitere Informationen finden Sie unter Grundlegendes zu den Ergebnissen der Trainings- und Testvalidierung.

```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "bounding-box": {
        "image_size": [
            {
                "width": 640,
                "height": 480,
                "depth": 3
            }
        ],
        "annotations": [

        ],
        "warnings": [
            {
                "code": "WARNING_NO_ATTRIBUTE_ANNOTATIONS",
                "message": "No attribute annotations were found."
            }
        ]
    },
```

```
    "bounding-box-metadata": {
        "objects": [

        ],
        "class-map": {

        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2013-11-18 02:53:27",
        "job-name": "my job"
    },
    "warnings": [
        {
            "code": "WARNING_NO_ANNOTATIONS",
            "message": "No annotations were found."
        }
    ]
}
```

Um WARNING_NO_ANNOTATIONS zu reparieren

- Wählen Sie eine der folgenden Optionen.
  - Fügen Sie die Informationen zum Begrenzungsfeld (`annotations`) zur JSON-Zeile hinzu. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien.
  - Entfernen Sie das Bild (JSON-Zeile) aus dem Manifest.

WARNUNG: NO_ATTRIBUTE_ANNOTIONS

Warnmeldung

Es wurden keine Attributanmerkungen bereitgestellt.

Weitere Informationen

Eine JSON-Zeile im Format Object Localization enthält keine Bounding-Box-Annotationsinformationen, obwohl sie von einem Menschen () annotiert wurde. `human-annotated = yes` Das `annotations` Array ist nicht vorhanden oder nicht gefüllt. Die JSON-Zeile ist gültig, wird

aber nicht für das Training verwendet. Weitere Informationen finden Sie unter Grundlegendes zu den Ergebnissen der Trainings- und Testvalidierung.


```
{
    "source-ref": "s3://bucket/images/IMG_1186.png",
    "bounding-box": {
        "image_size": [
            {
                "width": 640,
                "height": 480,
                "depth": 3
            }
        ],
        "annotations": [

        ],
        "warnings": [
            {
                "code": "WARNING_NO_ATTRIBUTE_ANNOTATIONS",
                "message": "No attribute annotations were found."
            }
        ]
    },
    "bounding-box-metadata": {
        "objects": [

        ],
        "class-map": {

        },
        "type": "groundtruth/object-detection",
        "human-annotated": "yes",
        "creation-date": "2013-11-18 02:53:27",
        "job-name": "my job"
    },
    "warnings": [
        {
            "code": "WARNING_NO_ANNOTATIONS",
            "message": "No annotations were found."
        }
    ]
}
```

Um WARNING_NO_ATTRIBUTE_ANNOTATIONS zu korrigieren

- Wählen Sie eine der folgenden Optionen.
  - Fügen Sie der JSON-Zeile ein oder mehrere `annotation` Bounding-Box-Objekte hinzu. Weitere Informationen finden Sie unter Objektlokalisierung in Manifestdateien.
  - Entfernen Sie das Bounding Box-Attribut.
  - Entfernen Sie das Bild (JSON-Zeile) aus dem Manifest. Wenn in der JSON-Zeile andere gültige Boundingbox-Attribute vorhanden sind, können Sie stattdessen nur das ungültige Boundingbox-Attribut aus der JSON-Zeile entfernen.

FEHLER_UNUNTERSTÜTZTED_ANWENDUNGSFALLTYP

Warnmeldung

Weitere Informationen

Der Wert des `type` Felds ist nicht `groundtruth/image-classification` oder`groundtruth/object-detection`. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

```
{
    "source-ref": "s3://bucket/test_normal_8.jpg",
    "BB": {
        "annotations": [
            {
                "left": 1768,
                "top": 1007,
                "width": 448,
                "height": 295,
                "class_id": 0
            },
            {
                "left": 1794,
                "top": 1306,
                "width": 432,
                "height": 411,
                "class_id": 1
            },
            {
                "left": 2568,
                "top": 1346,
```

```
                "width": 710,
                "height": 305,
                "class_id": 2
            },
            {
                "left": 2571,
                "top": 1020,
                "width": 644,
                "height": 312,
                "class_id": 3
            }
        ],
        "image_size": [
            {
                "width": 4000,
                "height": 2667,
                "depth": 3
            }
        ]
    },
    "BB-metadata": {
        "job-name": "labeling-job/BB",
        "class-map": {
            "0": "comparator",
            "1": "pot_resistor",
            "2": "ir_phototransistor",
            "3": "ir_led"
        },
        "human-annotated": "yes",
        "objects": [
            {
                "confidence": 1
            },
            {
                "confidence": 1
            },
            {
                "confidence": 1
            },
            {
                "confidence": 1
            }
        ],
        "creation-date": "2021-06-22T09:58:34.811Z",
```

```
        "type": "groundtruth/wrongtype",
        "cl-errors": [
            {
                "code": "ERROR_UNSUPPORTED_USE_CASE_TYPE",
                "message": "The use case type of the BB-metadata label attribute
 metadata is unsupported. Check the type field."
            }
        ]
    },
    "cl-metadata": {
        "is_labeled": true
    },
    "cl-errors": [
        {
            "code": "ERROR_NO_VALID_LABEL_ATTRIBUTES",
            "message": "No valid label attributes found."
        }
    ]
}
```

Um ERROR_UNSUPPORTED_USE_CASE_TYPE zu beheben

- Wählen Sie eine der folgenden Optionen:
  - Ändern Sie den Wert des `type` Felds in `groundtruth/image-classification` oder`groundtruth/object-detection`, je nachdem, welchen Modelltyp Sie erstellen möchten. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.
  - Entfernen Sie das Bild (JSON-Zeile) aus dem Manifest.

ERROR_INVALID_LABEL_NAME_LENGTH

Weitere Informationen

Die Länge eines Labelnamens ist zu lang. Die maximale Länge beträgt 256 Zeichen.

Um ERROR_INVALID_LABEL_NAME_LENGTH zu beheben

- Wählen Sie eine der folgenden Optionen:
  - Reduzieren Sie die Länge des Labelnamens auf 256 Zeichen oder weniger.
  - Entfernen Sie das Bild (JSON-Zeile) aus dem Manifest.

Verbessern eines geschulten Amazon Rekognition Custom Labels-Modells

Nach Abschluss des Trainings bewerten Sie die Leistung des Modells. Um Ihnen zu helfen, bietet Amazon Rekognition Custom Labels zusammenfassende Metriken und Bewertungsmetriken für jedes Label. Weitere Informationen zu den verfügbaren Metriken finden Sie unter Metriken für die Bewertung Ihres Modells. Informationen zur Verbesserung Ihres Modells mithilfe von Metriken finden Sie unter Verbessern eines Amazon Rekognition Custom Labels-Modells.

Wenn Sie mit der Genauigkeit Ihres Modells zufrieden sind, können Sie damit beginnen, es zu verwenden. Weitere Informationen finden Sie unter Ausführen eines trainierten Amazon Rekognition Custom Labels-Modells.

Themen

Metriken für die Bewertung Ihres Modells

Nachdem Ihr Modell trainiert wurde, gibt Amazon Rekognition Custom Labels Metriken aus Modelltests zurück, anhand derer Sie die Leistung Ihres Modells bewerten können. In diesem Thema werden die Metriken beschrieben, die Ihnen zur Verfügung stehen, und es wird beschrieben, wie Sie feststellen können, ob Ihr trainiertes Modell gut funktioniert.

Die Amazon Rekognition Custom Labels-Konsole bietet die folgenden Metriken als Zusammenfassung der Trainingsergebnisse und als Metriken für jedes Label:

- Genauigkeit
- Wiedererkennung
- F1

Jede von uns bereitgestellte Metrik ist eine häufig verwendete Metrik zur Bewertung der Leistung eines Modells für Machine Learning. Amazon Rekognition Custom Labels gibt Metriken für die Testergebnisse für den gesamten Testdatensatz zurück, zusammen mit Metriken für jedes benutzerdefinierte Label. Sie können auch die Leistung Ihres trainierten benutzerdefinierten Modells für jedes Bild in Ihrem Testdatensatz überprüfen. Weitere Informationen finden Sie unter Zugreifen auf Bewertungsmetriken (Konsole).

Bewerten der Modellleistung

Während des Tests prognostiziert Amazon Rekognition Custom Labels, ob ein Testbild ein benutzerdefiniertes Label enthält. Der Konfidenzwert ist ein Wert, der die Sicherheit der Vorhersage des Modells quantifiziert.

Wenn der Konfidenzwert für ein benutzerdefiniertes Label den Schwellenwert überschreitet, wird dieses Label in die Modellausgabe aufgenommen. Vorhersagen können auf folgende Weise kategorisiert werden:

- Richtig positiv — Das Amazon Rekognition Custom Labels-Modell sagt das Vorhandensein des benutzerdefinierten Labels im Testbild korrekt voraus. Das heißt, das vorhergesagte Label ist auch ein „Ground Truth“-Label für dieses Bild. Amazon Rekognition Custom Labels gibt beispielsweise korrekt ein Fußball-Label zurück, wenn ein Fußball in einem Bild vorhanden ist.
- Falsch positiv — Das Amazon Rekognition Custom Labels-Modell sagt fälschlicherweise das Vorhandensein eines benutzerdefinierten Labels in einem Testbild. Das heißt, das vorhergesagte Label ist kein Ground-Truth-Label für das Bild. Amazon Rekognition Custom Labels gibt beispielsweise ein Fußball-Label zurück, aber in Ground Truth gibt es kein Fußball-Label für dieses Bild.
- Falsch negativ — Das Amazon Rekognition Custom Labels-Modell sagt nicht voraus, dass ein benutzerdefiniertes Label im Bild vorhanden ist, aber die „Ground Truth“ für dieses Bild beinhaltet dieses Label. Amazon Rekognition Custom Labels gibt beispielsweise kein benutzerdefiniertes Label „Fußball“ für ein Bild zurück, das einen Fußball enthält.
- Richtig negativ — Das Amazon Rekognition Custom Labels-Modell sagt korrekt voraus, dass im Testbild kein benutzerdefiniertes Label vorhanden ist. Amazon Rekognition Custom Labels gibt beispielsweise kein Fußball-Label für ein Bild zurück, das keinen Fußball enthält.

Die Konsole bietet Zugriff auf echte positive, falsch positive und falsch negative Werte für jedes Bild in Ihrem Testdatensatz. Weitere Informationen finden Sie unter Zugreifen auf Bewertungsmetriken (Konsole).

Diese Vorhersageergebnisse werden verwendet, um die folgenden Metriken für jedes Label und eine Zusammenfassung für Ihren gesamten Testsatz zu berechnen. Dieselben Definitionen gelten für Vorhersagen, die das Modell auf der Begrenzungsrahmen-Ebene trifft, mit dem Unterschied, dass alle Metriken für jeden Begrenzungsrahmen (Vorhersage oder Ground Truth) in jedem Testbild berechnet werden.

Intersection over Union (IoU) und Objekterkennung

Intersection over Union (IoU) misst den Prozentsatz der Überlappung zwischen zwei Objektbegrenzungsrahmen in ihrer kombinierten Fläche. Der Bereich reicht von 0 (niedrigste Überlappung) bis 1 (vollständige Überlappung). Beim Testen ist ein vorhergesagter Begrenzungsrahmen korrekt, wenn der IoU des Ground Truth-Begrenzungsrahmen und des vorhergesagten Begrenzungsrahmen mindestens 0,5 beträgt.

Angenommener Schwellenwert

Amazon Rekognition Custom Labels berechnet automatisch einen angenommenen Schwellenwert (0-1) für jedes Ihrer benutzerdefinierten Labels. Sie können den angenommenen Schwellenwert für ein benutzerdefiniertes Label nicht festlegen. Der angenommene Schwellenwert für jedes Label ist der Wert, ab dem eine Vorhersage als richtig oder falsch positiv gewertet wird. Er wird auf der Grundlage Ihres Testdatensatzes festgelegt. Der angenommene Schwellenwert wird auf der Grundlage des besten F1-Werts berechnet, der beim Modelltraining im Testdatensatz erzielt wurde.

Sie können den Wert des angenommenen Schwellenwerts für ein Label aus den Trainingsergebnissen des Modells ermitteln. Weitere Informationen finden Sie unter Zugreifen auf Bewertungsmetriken (Konsole).

Änderungen der angenommenen Schwellenwerte werden in der Regel dazu verwendet, die Präzision und Erinnerungsvermögen eines Modells zu verbessern. Weitere Informationen finden Sie unter Verbessern eines Amazon Rekognition Custom Labels-Modells. Da Sie den angenommenen Schwellenwert eines Modells für ein Label nicht festlegen können, können Sie dieselben Ergebnisse erzielen, indem Sie ein Bild mit `MinConfidence` analysieren und `DetectCustomLabels`-Eingabeparameter angeben. Weitere Informationen finden Sie unter Analysieren eines Bildes mit einem trainierten Modell.

Genauigkeit

Amazon Rekognition Custom Labels bietet Präzisionsmetriken für jedes Label und eine durchschnittliche Präzisionsmetrik für den gesamten Testdatensatz.

Präzision ist der Anteil der korrekten Vorhersagen (wahre positive Ergebnisse) im Vergleich zu allen Modellvorhersagen (wahr und falsch positiv) beim angenommenen Schwellenwert für ein einzelnes Label. Wenn der Schwellenwert erhöht wird, trifft das Modell möglicherweise weniger Vorhersagen. Im Allgemeinen wird es jedoch im Vergleich zu einem niedrigeren Schwellenwert ein höheres Verhältnis zwischen echten positiven Ergebnissen und falsch positiven Ergebnissen aufweisen. Mögliche Werte für die Präzision liegen zwischen 0 und 1, und höhere Werte bedeuten eine höhere Präzision.

Wenn das Modell beispielsweise voraussagt, dass ein Fußball in einem Bild zu sehen ist, wie oft ist diese Vorhersage richtig? Angenommen, es gibt ein Bild mit 8 Fußbällen und 5 Steinen. Wenn das Modell 9 Fußbälle vorhersagt (8 richtig vorhergesagt und 1 falsch positives Ergebnis), dann beträgt die Präzision für dieses Beispiel 0,89. Wenn das Modell jedoch 13 Fußbälle im Bild mit 8 richtigen und 5 falschen Vorhersagen vorhergesagt hat, ist die resultierende Präzision geringer.

Weitere Informationen finden Sie unter Präzision und Erinnerung.

Wiedererkennung

Amazon Rekognition Custom Labels bietet durchschnittliche Erinnerungsmetriken für jedes Label und eine durchschnittliche Erinnerungsmetrik für den gesamten Testdatensatz.

Erinnerung ist die Fraktion Ihrer Testset-Labels, die korrekt vorhergesagt wurde und über dem angenommenen Schwellenwert liegt Es ist ein Maß dafür, wie oft das Modell ein benutzerdefiniertes Label korrekt vorhersagen kann, obwohl es tatsächlich in den Bildern Ihres Testsatzes vorhanden ist. Der Bereich für das Erinnerungsvermögen liegt zwischen 0 und 1. Höhere Werte deuten auf ein höheres Erinnerungsvermögen hin.

Wenn ein Bild beispielsweise 8 Fußbälle enthält, wie viele davon werden korrekt erkannt? In diesem Beispiel, in dem ein Bild 8 Fußbälle und 5 Steine enthält, beträgt das Erinnerungsvermögen 0,62, wenn das Modell 5 der Fußbälle erkennt. Wenn das neue Modell nach dem erneuten Taining 9 Fußbälle erkennt, einschließlich aller 8, die im Bild vorhanden waren, beträgt das Erinnerungsvermögen 1,0.

Weitere Informationen finden Sie unter Präzision und Erinnerung.

F1

Amazon Rekognition Custom Labels verwendet die F1-Wert-Metrik, um die durchschnittliche Modellleistung jedes Labels und die durchschnittliche Modellleistung des gesamten Testdatensatzes zu messen.

Die Modellleistung ist eine aggregierte Metrik, die sowohl die Präzision als auch das Erinnerungsvermögen aller Labels berücksichtigt. (z. B. F1-Wert oder durchschnittliche Präzision). Der Modellleistungswert ist ein Wert zwischen 0 und 1. Je höher der Wert, desto besser schneidet das Modell sowohl im Hinblick auf Erinnerung als auch Präzision ab. Insbesondere die Modellleistung bei Klassifikationsaufgaben wird üblicherweise anhand des F1-Werts gemessen. Dieser Wert ist das harmonische Mittel der Präzision- und Erinnerungswerte am angenommenen Schwellenwert. Für ein Modell mit einer Präzision von 0,9 und einem Erinnerungswert von 1,0 beträgt der F1-Wert beispielsweise 0,947.

Ein hoher Wert für den F1-Wert weist darauf hin, dass das Modell sowohl im Hinblick auf Präzision als auch Erinnerung eine gute Leistung erbringt. Wenn das Modell nicht gut abschneidet, z. B. mit einer niedrigen Präzision von 0,30 und einem hohen Erinnerungsvermögen von 1,0, beträgt der F1-Wert 0,46. In ähnlicher Weise beträgt der F1-Wert 0,33, wenn die Präzision hoch (0,95) und das Erinnerungsvermögen niedrig (0,20) ist. In beiden Fällen ist der F1-Wert niedrig und deutet auf Probleme mit dem Modell hin.

Weitere Informationen finden Sie unter F1-Wert.

Verwenden von -Metriken

Für ein bestimmtes Modell, das Sie trainiert haben, können Sie je nach Anwendung einen Kompromiss zwischen Präzision und Erinnerung eingehen, indem Sie den `MinConfidence` Eingabeparameter für `DetectCustomLabels` verwenden. Bei einem höheren `MinConfidence` Wert erhalten Sie in der Regel eine höhere Präzision (genauere Vorhersagen von Fußbällen), aber ein geringeres Erinnerungsvermögen (es werden mehr echte Fußbälle übersehen). Bei einem niedrigeren `MinConfidence` Wert erhalten Sie ein höheres Erinnerungsvermögen (mehr tatsächlich korrekt vorhergesagte Fußbälle), aber eine geringere Präzision (mehr dieser Vorhersagen werden falsch sein). Weitere Informationen finden Sie unter Analysieren eines Bildes mit einem trainierten Modell.

Die Kennzahlen informieren Sie auch darüber, welche Maßnahmen Sie ergreifen könnten, um die Modellleistung bei Bedarf zu verbessern. Weitere Informationen finden Sie unter Verbessern eines Amazon Rekognition Custom Labels-Modells.

> **Note**
>
> `DetectCustomLabels` gibt Vorhersagen im Bereich von 0 bis 100 zurück, die dem Metrikbereich von 0-1 entsprechen.

Zugreifen auf Bewertungsmetriken (Konsole)

Während des Tests wird das Modell anhand des Testdatensatzes auf seine Leistung hin bewertet. Die Labels im Testdatensatz gelten als „Ground Truth“, da sie das darstellen, was das tatsächliche Bild darstellt. Während des Tests trifft das Modell anhand des Testdatensatzes Vorhersagen. Die vorhergesagten Labels werden mit den Ground Truth-Bezeichnungen verglichen, und die Ergebnisse sind auf der Bewertungsseite der Konsole verfügbar.

Die Amazon Rekognition Custom Labels-Konsole zeigt zusammenfassende Metriken für das gesamte Modell und Metriken für einzelne Labels. Die in der Konsole verfügbaren Metriken sind Präzision, Erinnerung, F1-Wert, Konfidenz und Konfidenzschwelle. Weitere Informationen finden Sie unter Verbessern eines geschulten Amazon Rekognition Custom Labels-Modells.

Sie können die Konsole verwenden, um sich auf einzelne Metriken zu konzentrieren. Um beispielsweise Präzisionsprobleme bei einem Label zu untersuchen, können Sie die Trainingsergebnisse nach Label und nach falsch positiven Ergebnissen filtern. Weitere Informationen finden Sie unter Metriken für die Bewertung Ihres Modells.

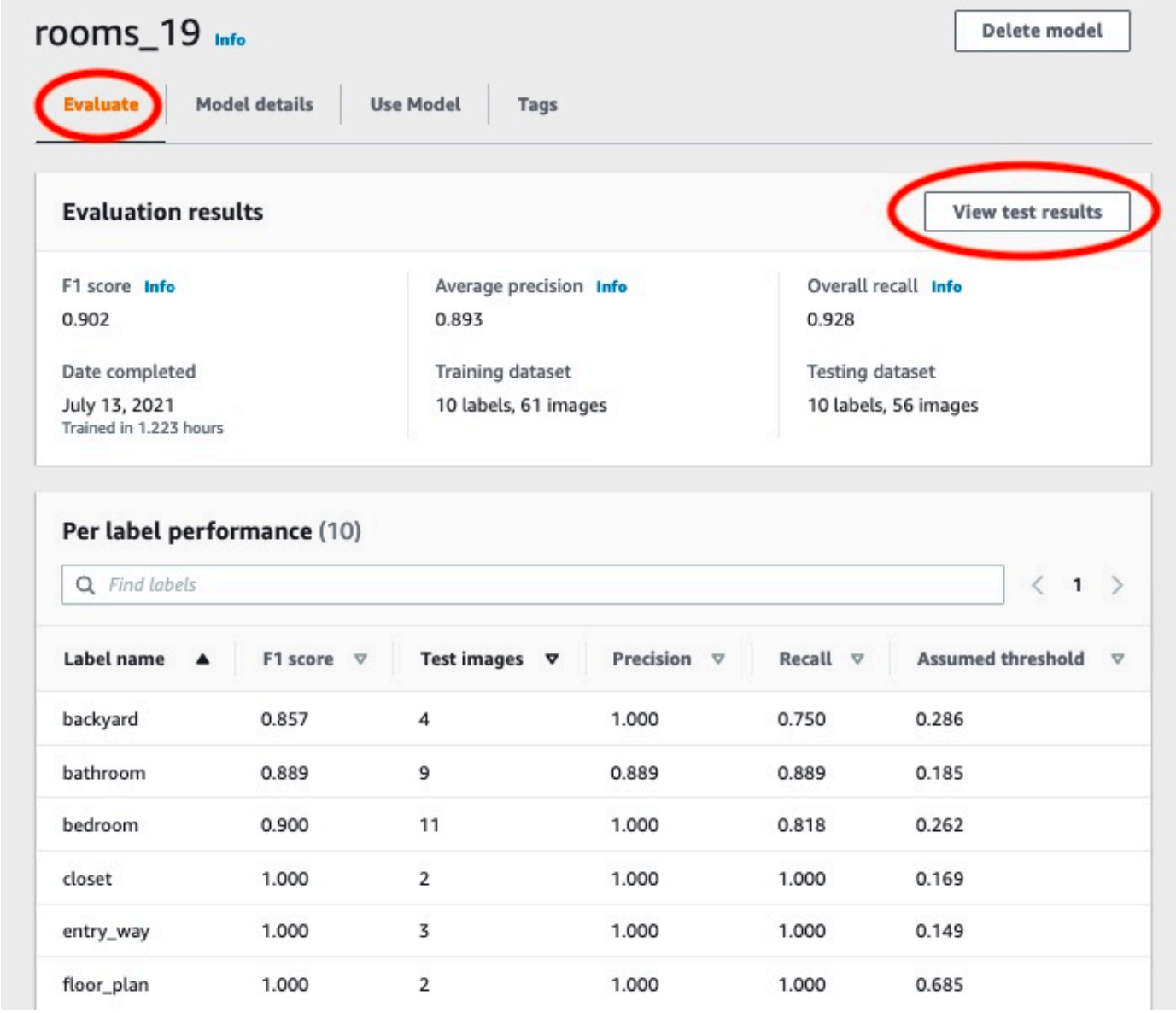
Nach dem Training ist der Trainingsdatensatz schreibgeschützt. Wenn Sie das Modell verbessern möchten, können Sie den Trainingsdatensatz in einen neuen Datensatz kopieren. Sie verwenden die Kopie des Datensatzes, um eine neue Version des Modells zu trainieren.

In diesem Schritt verwenden Sie die Konsole, um auf die Trainingsergebnisse in der Konsole zuzugreifen.

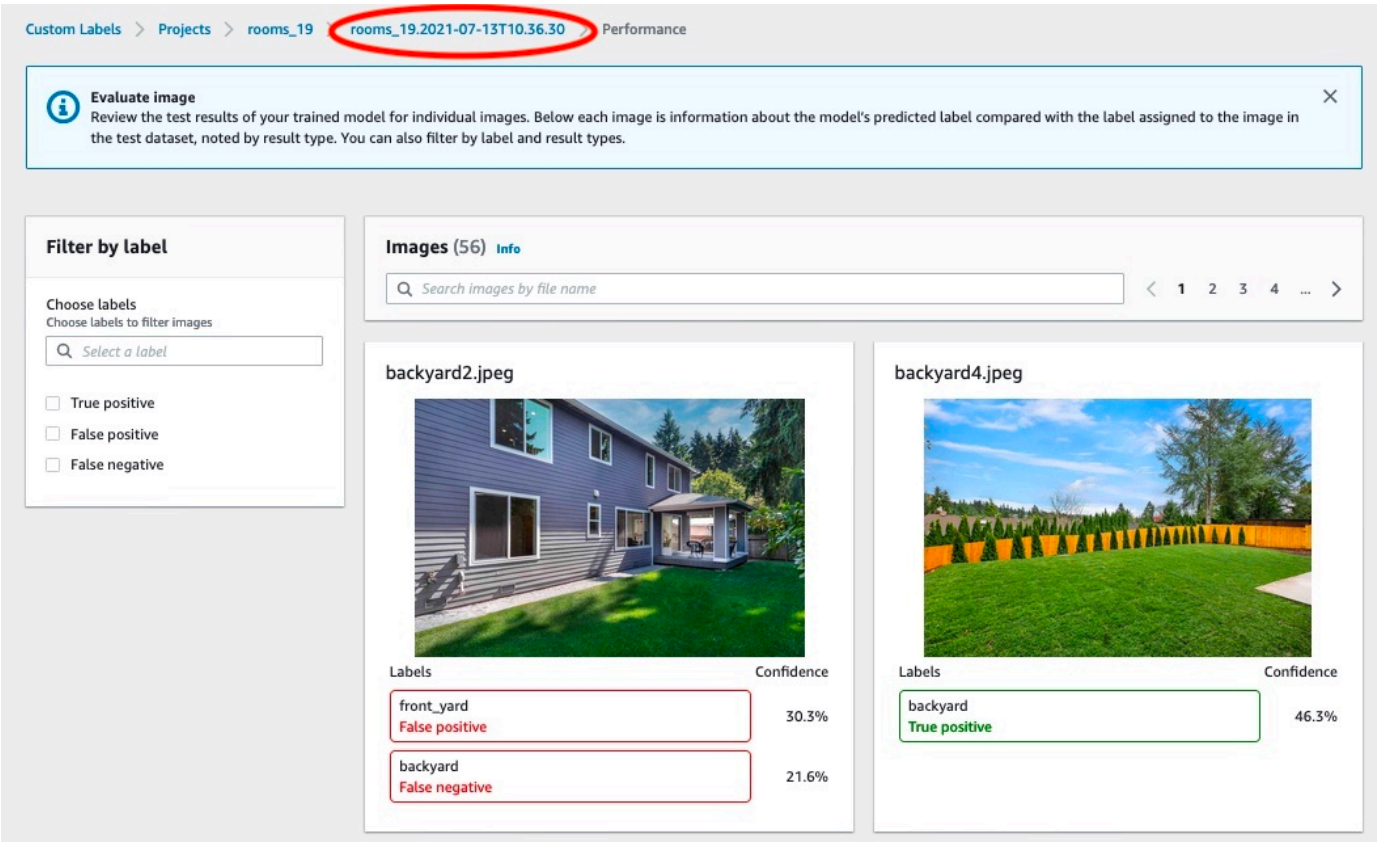
So greifen Sie auf Bewertungsmetriken zu (Konsole)

1. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Benutzerdefinierte Labels verwenden.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie auf der Seite Projekte das Projekt aus, das das trainierte Modell enthält, das Sie auswerten möchten.
6. Wählen Sie unter Modelle das Modell aus, das Sie bewerten möchten.
7. Wählen Sie die Registerkarte Bewertung, um die Bewertungsergebnisse zu sehen. Weitere Informationen zum Bewerten eines Modells finden Sie unter Verbessern eines geschulten Amazon Rekognition Custom Labels-Modells.

8. Wählen Sie Testergebnisse anzeigen, um die Ergebnisse für einzelne Testbilder anzuzeigen. Weitere Informationen finden Sie unter Metriken für die Bewertung Ihres Modells. Der folgende Screenshot der Zusammenfassung der Modellbewertung zeigt den F1-Wert, die durchschnittliche Genauigkeit und den Gesamterinnerungswert für 6 Labels mit Testergebnissen und Leistungskennzahlen. Einzelheiten zur Verwendung des trainierten Modells werden ebenfalls bereitgestellt.



9. Nachdem Sie sich die Testergebnisse angesehen haben, wählen Sie den Projektnamen aus, um zur Modellseite zurückzukehren. Auf der Seite mit den Testergebnissen werden Bilder mit vorhergesagten Kennzeichnungen und Konfidenzwerten für ein maschinelles Lernmodell angezeigt, das auf die Bildkategorien Hinterhof und Vorgarten trainiert wurde. Zwei Beispielbilder werden angezeigt.



10. Verwenden Sie die Metriken, um die Leistung des Modells zu bewerten. Weitere Informationen finden Sie unter Verbessern eines Amazon Rekognition Custom Labels-Modells.

Zugreifen auf Amazon Rekognition Custom Labels-Bewertungsmetriken (SDK)

Der Vorgang DescribeProjectVersionen bietet Zugriff auf Metriken, die über die in der Konsole bereitgestellten hinausgehen.

`DescribeProjectVersions` bietet wie die Konsole Zugriff auf die folgenden Metriken als zusammenfassende Informationen zu den Testergebnissen und als Testergebnisse für jedes Label:

- Genauigkeit
- Wiedererkennung
- F1

Der durchschnittliche Schwellenwert für alle Labels und der Schwellenwert für einzelne Labels wird zurückgegeben.

`DescribeProjectVersions` bietet außerdem Zugriff auf die folgenden Metriken für die Klassifizierung und Bilderkennung (Objektposition auf dem Bild).

- Konfusionsmatrix zur Bildklassifizierung. Weitere Informationen finden Sie unter Die Konfusionsmatrix für ein Modell anzeigen.
- Mittlere durchschnittliche Präzision (mAP) für die Bilderkennung.
- Mittlere durchschnittliche Erinnerung (mAR) für die Bilderkennung.

`DescribeProjectVersions` bietet auch Zugriff auf echte positive, falsch positive, falsch negative und echte negative Werte. Weitere Informationen finden Sie unter Metriken für die Bewertung Ihres Modells.

Die aggregierte F1-Werte-Metrik wird direkt von `DescribeProjectVersions` zurückgegeben. Auf andere Metriken kann über Übersichtsdatei und Bewertungsmanifest-Snapshot-Dateien zugegriffen werden, die in einem Amazon-S3-Bucket gespeichert sind. Weitere Informationen finden Sie unter Zugriff auf die Übersichtsdatei und den Snapshot (SDK) des Bewertungsmanifests.

Themen

- Übersichtsdatei
- Bewertungsmanifest-Snapshot
- Zugriff auf die Übersichtsdatei und den Snapshot (SDK) des Bewertungsmanifests
- Die Konfusionsmatrix für ein Modell anzeigen
- Referenz: Datei mit der Übersicht über die Trainingsergebnisse

Übersichtsdatei

Die Übersichtsdatei enthält Informationen zu den Bewertungsergebnissen über das gesamte Modell und Metriken für jedes Label. Die Metriken lauten Präzision, Erinnerung und F1-Wert. Der Schwellenwert für das Modell wird ebenfalls angegeben. Auf den Speicherort der Übersichtsdatei kann über das `EvaluationResult` Objekt zugegriffen werden, das von `DescribeProjectVersions`zurückgegeben wurde. Weitere Informationen finden Sie unter Referenz: Datei mit der Übersicht über die Trainingsergebnisse.

Im Folgenden sehen Sie ein Beispiel für eine Übersichtsdatei.

```
{
```

```
  "Version": 1,
  "AggregatedEvaluationResults": {
    "ConfusionMatrix": [
      {
        "GroundTruthLabel": "CAP",
        "PredictedLabel": "CAP",
        "Value": 0.9948717948717949
      },
      {
        "GroundTruthLabel": "CAP",
        "PredictedLabel": "WATCH",
        "Value": 0.008547008547008548
      },
      {
        "GroundTruthLabel": "WATCH",
        "PredictedLabel": "CAP",
        "Value": 0.1794871794871795
      },
      {
        "GroundTruthLabel": "WATCH",
        "PredictedLabel": "WATCH",
        "Value": 0.7008547008547008
      }
    ],
    "F1Score": 0.9726959470546408,
    "Precision": 0.9719115848331294,
    "Recall": 0.9735042735042735
  },
  "EvaluationDetails": {
    "EvaluationEndTimestamp": "2019-11-21T07:30:23.910943",
    "Labels": [
      "CAP",
      "WATCH"
    ],
    "NumberOfTestingImages": 624,
    "NumberOfTrainingImages": 5216,
    "ProjectVersionArn": "arn:aws:rekognition:us-east-1:nnnnnnnnn:project/my-project/
version/v0/1574317227432"
  },
  "LabelEvaluationResults": [
    {
      "Label": "CAP",
      "Metrics": {
        "F1Score": 0.9794344473007711,
```

```
        "Precision": 0.9819587628865979,
        "Recall": 0.9769230769230769,
        "Threshold": 0.9879502058029175
      },
      "NumberOfTestingImages": 390
    },
    {
      "Label": "WATCH",
      "Metrics": {
        "F1Score": 0.9659574468085106,
        "Precision": 0.961864406779661,
        "Recall": 0.9700854700854701,
        "Threshold": 0.014450683258473873
      },
      "NumberOfTestingImages": 234
    }
  ]
}
```

Bewertungsmanifest-Snapshot

Der Bewertungs-Manifest-Snapshot enthält detaillierte Informationen über die Testergebnisse. Der Snapshot enthält die Konfidenzbewertung für jede Vorhersage. Er beinhaltet auch die Klassifizierung der Vorhersage im Vergleich zur tatsächlichen Klassifizierung des Bildes (richtig positiv, wahr negativ, falsch positiv oder falsch negativ).

Bei den Dateien handelt es sich um Schnappschüsse, da nur Bilder enthalten sind, die für Tests und Trainings verwendet werden könnten. Bilder, die nicht verifiziert werden können, z. B. Bilder im falschen Format, sind nicht im Manifest enthalten. Auf den Speicherort des Test-Snapshots kann über das `TestingDataResult` Objekt zugegriffen werden, das von `DescribeProjectVersions` zurückgegeben wurde. Auf den Speicherort des Trainings-Snapshots kann über das `TrainingDataResult` Objekt zugegriffen werden, das von `DescribeProjectVersions` zurückgegeben wurde.

Der Snapshot liegt im SageMaker Ground-Truth-Manifest-Ausgabeformat vor, wobei Felder hinzugefügt wurden, um zusätzliche Informationen bereitzustellen, z. B. das Ergebnis der binären Klassifizierung einer Erkennung. Der folgende Ausschnitt zeigt die zusätzlichen Felder.

```
"rekognition-custom-labels-evaluation-details": {
    "version": 1,
    "is-true-positive": true,
```

```
    "is-true-negative": false,
    "is-false-positive": false,
    "is-false-negative": false,
    "is-present-in-ground-truth": true
    "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"]
}
```

- Version — Die Version des `rekognition-custom-labels-evaluation-details` Feldformats im Manifest-Snapshot.
- ist wahr-positiv... — Die binäre Klassifikation der Vorhersage basiert darauf, wie der Konfidenzwert im Vergleich zum Mindestschwellenwert für das Label abschneidet.
- ist-präsent-in-ground-truth — Wahr, wenn die vom Modell getroffene Vorhersage in den für das Training verwendeten Ground Truth-Informationen enthalten ist, andernfalls falsch. Dieser Wert basiert nicht darauf, ob der Konfidenzwert den vom Modell berechneten Mindestschwellenwert überschreitet.
- Ground Truth-Labeling-Jobs — Eine Liste von Ground-Truth-Feldern in der Manifestzeile, die für Trainings verwendet werden.

Informationen zum SageMaker Ground Truth-Manifestformat finden Sie unter Ausgabe.

Im Folgenden finden Sie ein Beispiel für einen Snapshot eines Testmanifests, der Metriken für die Bildklassifizierung und Objekterkennung zeigt.

```
// For image classification
{
  "source-ref": "s3://test-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": 1,
  "rekognition-custom-labels-training-0-metadata": {
    "confidence": 1.0,
    "job-name": "rekognition-custom-labels-training-job",
    "class-name": "Football",
    "human-annotated": "yes",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification"
  },
  "rekognition-custom-labels-evaluation-0": 1,
  "rekognition-custom-labels-evaluation-0-metadata": {
    "confidence": 0.95,
    "job-name": "rekognition-custom-labels-evaluation-job",
    "class-name": "Football",
```

```
    "human-annotated": "no",
    "creation-date": "2019-09-06T00:07:25.488243",
    "type": "groundtruth/image-classification",
    "rekognition-custom-labels-evaluation-details": {
      "version": 1,
      "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
      "is-true-positive": true,
      "is-true-negative": false,
      "is-false-positive": false,
      "is-false-negative": false,
      "is-present-in-ground-truth": true
    }
  }
}


// For object detection
{
  "source-ref": "s3://test-bucket/dataset/beckham.jpeg",
  "rekognition-custom-labels-training-0": {
    "annotations": [
      {
        "class_id": 0,
        "width": 39,
        "top": 409,
        "height": 63,
        "left": 712
      },
      ...
    ],
    "image_size": [
      {
        "width": 1024,
        "depth": 3,
        "height": 768
      }
    ]
  },
  "rekognition-custom-labels-training-0-metadata": {
    "job-name": "rekognition-custom-labels-training-job",
    "class-map": {
      "0": "Cap",
      ...
    },
```

```
    "human-annotated": "yes",
    "objects": [
      {
        "confidence": 1.0
      },
      ...
    ],
    "creation-date": "2019-10-21T22:02:18.432644",
    "type": "groundtruth/object-detection"
  },
  "rekognition-custom-labels-evaluation": {
    "annotations": [
      {
        "class_id": 0,
        "width": 39,
        "top": 409,
        "height": 63,
        "left": 712
      },
      ...
    ],
    "image_size": [
      {
        "width": 1024,
        "depth": 3,
        "height": 768
      }
    ]
  },
  "rekognition-custom-labels-evaluation-metadata": {
    "confidence": 0.95,
    "job-name": "rekognition-custom-labels-evaluation-job",
    "class-map": {
      "0": "Cap",
      ...
    },
    "human-annotated": "no",
    "objects": [
      {
        "confidence": 0.95,
        "rekognition-custom-labels-evaluation-details": {
          "version": 1,
          "ground-truth-labelling-jobs": ["rekognition-custom-labels-training-job"],
          "is-true-positive": true,
```

```
          "is-true-negative": false,
          "is-false-positive": false,
          "is-false-negative": false,
          "is-present-in-ground-truth": true
        }
      },
      ...
    ],
    "creation-date": "2019-10-21T22:02:18.432644",
    "type": "groundtruth/object-detection"
  }
}
```

Zugriff auf die Übersichtsdatei und den Snapshot (SDK) des Bewertungsmanifests

Um Trainingsergebnisse zu erhalten, rufen Sie DescribeProjectVersions auf. Beispielcode finden Sie unter Beschreibung eines Modells (SDK).

Der Speicherort der Metriken wird in der `ProjectVersionDescription` Antwort von `DescribeProjectVersions` zurückgegeben.

- `EvaluationResult`— Der Speicherort der Übersichtsdatei.
- `TestingDataResult`— Der Speicherort des Bewertungsmanifest-Snapshots, der für Tests verwendet wurde.

Der F1-Wert und der Speicherort der Übersichtsdatei werden in `EvaluationResult` zurückgegeben. Beispielsweise:

```
"EvaluationResult": {
                "F1Score": 1.0,
                "Summary": {
                    "S3Object": {
                        "Bucket": "echo-dot-scans",
                        "Name": "test-output/EvaluationResultSummary-my-echo-dots-
project-v2.json"
                    }
                }
            }
```

Der Snapshot des Bewertungsmanifests wird an dem Speicherort gespeichert, den Sie in dem `--output-config` Eingabeparameter angegeben haben, den Sie in Ein Modell trainieren (SDK) angegeben haben.

> **Note**
>
> Die Zeit in Sekunden, in der Ihnen das Training in Rechnung gestellt wird, wird in `BillableTrainingTimeInSeconds` zurückgegeben.

Informationen zu den Metriken, die von den Amazon Rekognition Custom Labels zurückgegeben werden, finden Sie unter Zugreifen auf Amazon Rekognition Custom Labels-Bewertungsmetriken (SDK).

Die Konfusionsmatrix für ein Modell anzeigen

Eine Konfusionsmatrix ermöglicht es Ihnen, die Labels zu sehen, die Ihr Modell mit anderen Labels in Ihrem Modell verwechselt. Durch die Verwendung einer Konfusionsmatrix können Sie Ihre Verbesserungen auf das Modell konzentrieren.

Während der Modellbewertung erstellen Amazon Rekognition Custom Labels eine Verwechslungsmatrix, indem die Testbilder verwendet werden, um falsch identifizierte (verwechselte) Labels zu identifizieren. Amazon Rekognition Custom Labels erstellt nur eine Verwechslungsmatrix für Klassifizierungsmodelle. Auf die Klassifizierungsmatrix kann über die Übersichtsdatei zugegriffen werden, die Amazon Rekognition Custom Labels während des Modelltrainings erstellt. Sie können die Konfusionsmatrix in der Amazon Rekognition Custom Labels-Konsole nicht anzeigen.

Themen

Verwenden Sie eine Konfusionsmatrix

Die folgende Tabelle enthält die Konfusionsmatrix für das Beispielprojekt Raumklassifizierung. Bei den Spaltenüberschriften handelt es sich um die Bezeichnungen (Ground Truth-Labels), die den Testbildern zugewiesen wurden. Zeilenüberschriften sind die Labels, die das Modell für die Testbilder vorhersagt. Jede Zelle gibt den Prozentsatz der Vorhersagen für ein Label (Zeile) an, bei dem es sich um das Ground Truth-Etikett (Spalte) handeln sollte. Beispielsweise waren 67 % der Vorhersagen

für Badezimmer korrekt als Badezimmer gekennzeichnet. 33 % Prozent der Badezimmer waren falsch als Küchen gekennzeichnet. Ein leistungsstarkes Modell weist hohe Zellenwerte auf, wenn das vorhergesagte Label mit dem Ground Truth-Label übereinstimmt. Sie können diese Werte als diagonale Linie vom ersten bis zum letzten vorhergesagten Wert und als Ground Truth-Labels erkennen. Wenn ein Zellenwert 0 ist, wurden keine Vorhersagen für das vorhergesagte Label der Zelle getroffen, bei der es sich um das Ground Truth-Labels der Zelle handeln sollte.

> **Note**
>
> Da Modelle nicht deterministisch sind, können die Werte der Konfusionsmatrix-Zellen, die Sie beim Training des Raumprojekts erhalten, von der folgenden Tabelle abweichen.

Die Konfusionsmatrix identifiziert Bereiche, auf die Sie sich konzentrieren sollten. Die Konfusionsmatrix zeigt beispielsweise, dass das Modell in 50 % der Fälle Ankleidezimmer mit Schlafzimmern verwechselt hat. In diesem Fall sollten Sie Ihrem Trainingsdatensatz weitere Bilder von Ankleidezimmern und Schlafzimmern hinzufügen. Überprüfen Sie auch, ob die vorhandenen Ankleidezimmer- und Schlafzimmerbilder die korrekten Label haben. Dies sollte dem Modell helfen, besser zwischen den beiden Labels zu unterscheiden. Informationen zum Hinzufügen weiterer Bilder zu einem Datensatz finden Sie unter Hinzufügen weiterer Bilder zu einem Datensatz.

Die Konfusionsmatrix ist zwar hilfreich, es ist jedoch wichtig, andere Metriken zu berücksichtigen. Beispielsweise haben 100 % der Vorhersagen das Label Grundriss korrekt gefunden, was auf eine hervorragende Leistung hinweist. Der Testdatensatz enthält jedoch nur 2 Bilder mit dem Label Grundriss. Es enthält auch 11 Bilder mit dem Label Wohnraum. Dieses Ungleichgewicht besteht auch im Trainingsdatensatz (13 Wohnraum-Bilder und 2 Ankleidezimmer-Bilder). Um eine genauere Bewertung zu erhalten, sollten Sie die Trainings- und Testdatensätze ausbalancieren, indem Sie weitere Bilder von unterrepräsentierten Labels hinzufügen (Grundrisse in diesem Beispiel). Informationen zur Anzahl der Testbilder pro Label finden Sie unter Zugreifen auf Bewertungsmetriken (Konsole).

	Ground Truth Label									
Vorhergesagtes Label	Hinterhof	Badezimmer	Schlafzimmer	Ankleidezimmer	Eingang	Grundriss	Vorgarten	Küche	Wohnraum	Terrasse
Hinterhof	75 %	0%	0%	0%	0%	0%	33%	0%	0%	0%

	Ground Truth Label									
Vorhergesagtes Label	Hinterhof	Badezimmer	Schlafzimmer	Ankleidezimmer	Eingang	Grundriss	Vorgarten	Küche	Wohnraum	Terrasse
Badezimmer	0%	67%	0%	0%	0%	0%	0%	0%	0%	0%
Schlafzimmer	0%	0%	82%	50 %	0%	0%	0%	0%	9%	0%
Ankleidezimmer	0%	0%	0%	50 %	0%	0%	0%	0%	0%	0%
Eingang	0%	0%	0%	0%	33%	0%	0%	0%	0%	0%
Grundriss	0%	0%	0%	0%	0%	100 %	0%	0%	0%	0%
Vorgarten	25 %	0%	0%	0%	0%	0%	67%	0%	0%	0%
Küche	0%	33%	0%	0%	0%	0%	0%	88%	0%	0%
Wohnraum	0%	0%	18%	0%	67%	0%	0%	12%	91%	33%
Terrasse	0%	0%	0%	0%	0%	0%	0%	0%	0%	67%

Abrufen der Konfusionsmatrix für ein Modell

Der folgende Code verwendet die Operationen DescribeProjectsund DescribeProjectVersionen, um die Zusammenfassungsdatei für ein Modell abzurufen. Anschließend wird die Zusammenfassungsdatei verwendet, um die Konfusionsmatrix für das Modell anzuzeigen.

So zeigen Sie die Konfusionsmatrix für ein Modell (SDK) an

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.

2. Verwenden Sie den folgenden Code, um die Konfusionsmatrix für ein Modell anzuzeigen. Geben Sie die folgenden Befehlszeilenargumente an:

   - `project_name` — der Name des Projekts, das Sie verwenden möchten. Sie können den Projektnamen auf der Projektseite in der Amazon Rekognition Custom Labels-Konsole abrufen.
   - `version_name` – Die Versionsnummer des Modells, das Sie verwenden möchten. Sie können den Versionsnamen auf der Seite mit den Projektdetails in der Amazon Rekognition Custom Labels-Konsole abrufen.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose

Shows how to display the confusion matrix for an Amazon Rekognition Custom labels
 image
classification model.
"""


import json
import argparse
import logging
import boto3
import pandas as pd
from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def get_model_summary_location(rek_client, project_name, version_name):
    """
    Get the summary file location for a model.

    :param rek_client: A Boto3 Rekognition client.
    :param project_arn: The Amazon Resource Name (ARN) of the project that contains
 the model.
    :param model_arn: The Amazon Resource Name (ARN) of the model.
```

```
    :return: The location of the model summary file.
    """

    try:
        logger.info(
            "Getting summary file for model %s in project %s.", version_name,
 project_name)

        summary_location = ""

        # Get the project ARN from the project name.
        response = rek_client.describe_projects(ProjectNames=[project_name])

        assert len(response['ProjectDescriptions']) > 0, \
            f"Project {project_name} not found."

        project_arn = response['ProjectDescriptions'][0]['ProjectArn']

        # Get the summary file location for the model.
        describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
        assert len(describe_response['ProjectVersionDescriptions']) > 0, \
            f"Model {version_name} not found."

        model=describe_response['ProjectVersionDescriptions'][0]

        evaluation_results=model['EvaluationResult']

        summary_location=(f"s3://{evaluation_results['Summary']['S3Object']
['Bucket']}"
                          f"/{evaluation_results['Summary']['S3Object']
['Name']}")

        return summary_location

    except ClientError as err:
        logger.exception(
            "Couldn't get summary file location: %s", err.response['Error']
['Message'])
        raise
```

```
def show_confusion_matrix(summary):
    """
    Shows the confusion matrix for an Amazon Rekognition Custom Labels
    image classification model.
    :param summary: The summary file JSON object.
    """
    pd.options.display.float_format = '{:.0%}'.format

    # Load the model summary JSON into a DataFrame.

    summary_df = pd.DataFrame(
        summary['AggregatedEvaluationResults']['ConfusionMatrix'])

    # Get the confusion matrix.
    confusion_matrix = summary_df.pivot_table(index='PredictedLabel',
                                              columns='GroundTruthLabel',
                                              fill_value=0.0).astype(float)

    # Display the confusion matrix.
    print(confusion_matrix)


def get_summary(s3_resource, summary):
    """
    Gets the summary file.
    : return: The summary file in bytes.
    """
    try:
        summary_bucket, summary_key = summary.replace(
            "s3://", "").split("/", 1)

        bucket = s3_resource.Bucket(summary_bucket)
        obj = bucket.Object(summary_key)
        body = obj.get()['Body'].read()
        logger.info(
            "Got summary file '%s' from bucket '%s'.",
            obj.key, obj.bucket_name)
    except ClientError:
        logger.exception(
            "Couldn't get summary file '%s' from bucket '%s'.",
            obj.key, obj.bucket_name)
        raise
    else:
        return body
```

```
def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    : param parser: The command line parser.
    """

    parser.add_argument(
        "project_name", help="The ARN of the project in which the model resides."
    )
    parser.add_argument(
        "version_name", help="The version of the model that you want to describe."
    )


def main():
    """
    Entry point for script.
    """

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get the command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Showing confusion matrix for: {args.version_name} for project
 {args.project_name}.")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")
        s3_resource = session.resource('s3')

        # Get the summary file for the model.
        summary_location = get_model_summary_location(rekognition_client,
 args.project_name,
                                                      args.version_name
                                                      )
```

```
        summary = json.loads(get_summary(s3_resource, summary_location))

        # Check that the confusion matrix is available.
        assert 'ConfusionMatrix' in summary['AggregatedEvaluationResults'], \
            "Confusion matrix not found in summary. Is the model a classification
 model?"

        # Show the confusion matrix.
        show_confusion_matrix(summary)
        print("Done")

    except ClientError as err:
        logger.exception("Problem showing confusion matrix: %s", err)
        print(f"Problem describing model: {err}")

    except AssertionError as err:
        logger.exception(
            "Error: %s.\n", err)
        print(
            f"Error: {err}\n")


if __name__ == "__main__":
    main()
```

Referenz: Datei mit der Übersicht über die Trainingsergebnisse

Die Übersicht über die Trainingsergebnisse enthält Metriken, die Sie zur Bewertung Ihres Modells verwenden können. Die Übersichtsdatei wird auch verwendet, um Metriken auf der Seite mit den Trainingsergebnissen der Konsole anzuzeigen. Die Übersichtsdatei wird nach dem Training in einem Amazon-S3-Bucket gespeichert. Rufen Sie `DescribeProjectVersion` auf, um die Übersichtsdatei zu erhalten. Beispielcode finden Sie unter Zugriff auf die Übersichtsdatei und den Snapshot (SDK) des Bewertungsmanifests.

Übersichtsdatei

Das folgende JSON-Format ist das Format der Übersichtsdatei.

EvaluationDetails (Abschnitt 3)

Übersichtsinformationen über die Trainingsaufgabe. Dazu gehören der ARN des Projekts, zu dem das Modell gehört (`ProjectVersionArn`), Datum und Uhrzeit des Abschlusses des Trainings, die Version des Modells, die bewertet wurde (`EvaluationEndTimestamp`) und eine Liste der während des Trainings erkannten Labels (`Labels`). Ebenfalls enthalten ist die Anzahl der Bilder, die für das Training (`NumberOfTrainingImages`) und die Bewertung (`NumberOfTestingImages`) verwendet wurden.

AggregatedEvaluationResults (Abschnitt 1)

Sie können `AggregatedEvaluationResults` verwenden, um die Gesamtleistung des trainierten Modells zu bewerten, wenn es mit dem Testdatensatz verwendet wird. Aggregierte Metriken sind für die Metriken `Precision`, `Recall` und `F1Score` enthalten. Für die Objekterkennung (die Objektposition auf einem Bild) werden `AverageRecall` (mAR) und `AveragePrecision` (mAP) Metriken zurückgegeben. Für die Klassifizierung (der Objekttyp in einem Bild) wird eine Metrik mit einer Konfusionsmatrix zurückgegeben.

LabelEvaluationResults (Abschnitt 2)

Sie können `labelEvaluationResults` verwenden, um die Leistung einzelner Labels zu bewerten. Die Labels sind nach dem F1-Wert der einzelnen Labels sortiert. Die enthaltenen Metriken sind `Precision`, `Recall`, `F1Score` und `Threshold` (werden zur Klassifizierung verwendet).

Der Dateiname ist wie folgt formatiert: `EvaluationSummary-ProjectName-VersionName.json`.

```
{
  "Version": "integer",
  // section-3
  "EvaluationDetails": {
    "ProjectVersionArn": "string",
    "EvaluationEndTimestamp": "string",
    "Labels": "[string]",
    "NumberOfTrainingImages": "int",
    "NumberOfTestingImages": "int"
  },
  // section-1
  "AggregatedEvaluationResults": {
    "Metrics": {
      "Precision": "float",
      "Recall": "float",
      "F1Score": "float",
      // The following 2 fields are only applicable to object detection
```

```
      "AveragePrecision": "float",
      "AverageRecall": "float",
      // The following field is only applicable to classification
      "ConfusionMatrix":[
        {
          "GroundTruthLabel": "string",
          "PredictedLabel": "string",
          "Value": "float"
        },
        ...
      ],
    }
  },
  // section-2
  "LabelEvaluationResults": [
    {
      "Label": "string",
      "NumberOfTestingImages", "int",
      "Metrics": {
        "Threshold": "float",
        "Precision": "float",
        "Recall": "float",
        "F1Score": "float"
      },
    },
    ...
  ]
}
```

Verbessern eines Amazon Rekognition Custom Labels-Modells

Die Leistung von Modellen für Machine Learning hängt weitgehend von Faktoren wie der Komplexität und Variabilität Ihrer benutzerdefinierten Labels (der spezifischen Objekte und Szenen, an denen Sie interessiert sind), der Qualität und Repräsentativität des von Ihnen bereitgestellten Trainingsdatensatzes sowie den Modellframeworks und Methoden des Machine Learning ab, die zum Trainieren des Modells verwendet werden.

Amazon Rekognition Custom Labels vereinfacht diesen Prozess, und es sind keine Fachkenntnisse im Bereich Machine Learning erforderlich. Der Prozess der Erstellung eines guten Modells beinhaltet jedoch häufig Iterationen von Daten und Modellverbesserungen, um die gewünschte Leistung zu erzielen. Im Folgenden finden Sie Informationen zur Verbesserung Ihres Modells.

Daten

Im Allgemeinen können Sie die Qualität Ihres Modells verbessern, indem Sie größere Mengen an Daten mit besserer Qualität verwenden. Verwenden Sie Trainingsbilder, die das Objekt oder die Szene deutlich zeigen und nicht mit unnötigen Objekten überladen sind. Verwenden Sie für Begrenzungsrahmen rund um Objekte Trainingsbilder, auf denen das Objekt vollständig sichtbar ist und nicht von anderen Objekten verdeckt wird.

Stellen Sie sicher, dass Ihre Trainings- und Testdatensätze mit der Art von Bildern übereinstimmen, für die Sie letztendlich Inferenz durchführen werden. Für Objekte wie Logos, für die Sie nur wenige Trainingsbeispiele haben, sollten Sie in Ihren Testbildern Begrenzungsrahmen um das Logo herum platzieren. Diese Bilder stellen die Szenarien dar oder stellen sie dar, in denen Sie das Objekt lokalisieren möchten.

Informationen zum Hinzufügen weiterer Bilder zu einem Trainings- oder Testdatensatz finden Sie unter Hinzufügen weiterer Bilder zu einem Datensatz.

Reduzierung falsch positiver Ergebnisse (höhere Präzision)

- Prüfen Sie zunächst, ob Sie durch eine Erhöhung des angenommenen Schwellenwerts die richtigen Vorhersagen beibehalten und gleichzeitig die Zahl der falsch positiven Ergebnisse verringern können. Irgendwann hat dies aufgrund des Kompromisses zwischen Präzision und Erinnerungsvermögen bei einem bestimmten Modell immer weniger Vorteile zur Folge. Sie können den angenommenen Schwellenwert für ein Label nicht festlegen, aber Sie können dasselbe Ergebnis erzielen, indem Sie einen hohen Wert für den `MinConfidence` Eingabeparameter auf angeben. `DetectCustomLabels` Weitere Informationen finden Sie unter Analysieren eines Bildes mit einem trainierten Modell.
- Möglicherweise werden eines oder mehrere Ihrer benutzerdefinierten Labels von Interesse (A) immer wieder mit derselben Objektklasse (aber nicht mit einem Label, an dem Sie interessiert sind) (B) verwechselt. Um Ihnen zu helfen, fügen Sie B als Objektklassenlabel zu Ihrem Trainingsdatensatz hinzu (zusammen mit den Bildern, bei denen Sie das falsch positive Ergebnis erhalten haben). Tatsächlich helfen Sie dem Modell mit den neuen Trainingsbilder zu lernen, B und nicht A vorherzusagen. Informationen zum Hinzufügen von Bildern zu einem Trainingsdatensatz finden Sie unter Hinzufügen weiterer Bilder zu einem Datensatz.
- Möglicherweise stellen Sie fest, dass das Modell durch zwei Ihrer benutzerdefinierten Labels (A und B) verwechselt wird. Es wird vorhergesagt, dass das Testbild mit dem Label A das Label B hat und umgekehrt. Prüfen Sie in diesem Fall zunächst, ob Ihre Trainings- und Testsätze falsch

beschriftete Bilder enthalten. Verwenden Sie die Datensatz-Galerie, um die einem Datensatz zugewiesenen Labels zu verwalten. Weitere Informationen finden Sie unter Labels verwalten. Wenn Sie weitere Trainingsbilder hinzufügen, die sich auf diese Art von Verwechslungen beziehen, kann ein neu trainiertes Modell außerdem besser zwischen A und B unterscheiden. Informationen zum Hinzufügen von Bildern zu einem Trainingsdatensatz finden Sie unter. Hinzufügen weiterer Bilder zu einem Datensatz

Reduzierung falsch negativer Ergebnisse (besseres Erinnerungsvermögen)

- Verwenden Sie einen niedrigeren Wert für den angenommenen Schwellenwert. Sie können den angenommenen Schwellenwert für ein Label nicht festlegen, aber Sie können dasselbe Ergebnis erzielen, indem Sie einen niedrigeren `MinConfidence` Eingabeparameter für `DetectCustomLabels` angeben. Weitere Informationen finden Sie unter Analysieren eines Bildes mit einem trainierten Modell.
- Verwenden Sie bessere Beispiele, um die Vielfalt sowohl des Objekts als auch der Bilder, in denen sie vorkommen, zu modellieren.
- Teilen Sie Ihr Label in zwei Klassen auf, die leichter zu erlernen sind. Beispielsweise könnten Sie anstelle von guten und schlechten Keksen gute, verbrannte und kaputte Kekse verwenden, damit das Modell jedes einzelne Konzept besser erlernen kann.

Ausführen eines trainierten Amazon Rekognition Custom Labels-Modells

Wenn Sie mit der Leistung des Modells zufrieden sind, können Sie damit beginnen, es zu verwenden. Sie können ein Modell mithilfe der Konsole oder des AWS SDK starten und beenden. Die Konsole enthält auch Beispiele für SDK-Operationen, die Sie verwenden können.

Themen

Inferenzeinheiten

Wenn Sie Ihr Modell starten, geben Sie die Anzahl der Rechenressourcen an, die als Inferenzeinheit bezeichnet werden, die das Modell verwendet.

> **Important**
>
> Ihnen werden die Anzahl der Stunden, in denen Ihr Modell läuft, und die Anzahl der Inferenzeinheiten, die Ihr Modell während der Ausführung verwendet, in Rechnung gestellt, je nachdem, wie Sie den Betrieb Ihres Modells konfigurieren. Wenn Sie das Modell beispielsweise mit zwei Inferenzeinheiten starten und das Modell 8 Stunden lang verwenden, werden Ihnen 16 Inferenzstunden in Rechnung gestellt (8 Stunden Laufzeit x zwei Inferenzeinheiten). Weitere Informationen finden Sie unter Inferenzstunden. Wenn Sie Ihr Modell nicht ausdrücklich stoppen, werden Ihnen Gebühren berechnet, auch wenn Sie nicht aktiv Bilder mit Ihrem Modell analysieren.

Die Transaktionen pro Sekunde (TPS), die eine einzelne Inferenzeinheit unterstützt, werden durch die folgenden Faktoren beeinflusst.

- Ein Modell, das Labels auf Bildebene erkennt (Klassifizierung), hat im Allgemeinen einen höheren TPS als ein Modell, das Objekte mit Begrenzungsrahmen erkennt und lokalisiert (Objekterkennung).
- Die Komplexität des Modells.
- Ein Bild mit höherer Auflösung benötigt mehr Zeit für die Analyse.
- Mehr Objekte in einem Bild erfordern mehr Zeit für die Analyse.
- Kleinere Bilder werden schneller analysiert als größere Bilder.
- Ein als Bildbyte übergebenes Bild wird schneller analysiert, als das Bild zuerst in einen Amazon-S3-Bucket hochzuladen und dann auf das hochgeladene Bild zu verweisen. Bilder, die als Bildbytes übergeben werden, müssen kleiner als 4.0 MB sein. Wir empfehlen die Verwendung von Bildbytes für die Verarbeitung von Bildern nahezu in Echtzeit und bei einer Bildgröße von weniger als 4.0 MB. Zum Beispiel Bilder, die mit einer IP-Kamera aufgenommen wurden.
- Die Verarbeitung von Bildern, die in einem Amazon-S3-Bucket gespeichert sind, ist schneller als das Herunterladen der Bilder, das Konvertieren in Bildbytes und das anschließende Übergeben der Bildbytes zur Analyse.
- Die Analyse eines Bildes, das bereits in einem Amazon-S3-Bucket gespeichert ist, ist wahrscheinlich schneller als die Analyse desselben Bilds, das als Bildbytes übergeben wurde. Das gilt insbesondere, wenn die Bildgröße größer ist.

Wenn die Anzahl der Aufrufe von `DetectCustomLabels` die maximale Anzahl an TPS überschreitet, die von der Summe der von einem Modell verwendeten Inferenzeinheiten unterstützt wird, gibt Amazon Rekognition Custom Labels eine `ProvisionedThroughputExceededException`-Ausnahme zurück.

Verwaltung des Durchsatzes mit Inferenzeinheiten

Sie können den Durchsatz Ihres Modells je nach den Anforderungen an Ihre Anwendung erhöhen oder verringern. Verwenden Sie zusätzliche Inferenzeinheiten, um den Durchsatz zu erhöhen. Jede zusätzliche Inferenzeinheit erhöht Ihre Verarbeitungsgeschwindigkeit um eine Inferenzeinheit. Informationen zur Berechnung der Anzahl der benötigten Inferenzeinheiten finden Sie unter Berechnung von Inferenzeinheiten für Amazon Rekognition Custom Labels und Amazon Lookout für Vision-Modelle. Wenn Sie den unterstützten Durchsatz Ihres Modells ändern möchten, haben Sie zwei Möglichkeiten:

Manuelles Hinzufügen oder Entfernen von Inferenzeinheiten

Stoppen Sie das Modell und starten Sie es dann mit der erforderlichen Anzahl von Inferenzeinheiten neu. Der Nachteil dieses Ansatzes besteht darin, dass das Modell während des Neustarts keine Anfragen empfangen kann und nicht zur Bewältigung von Nachfragespitzen verwendet werden kann. Verwenden Sie diesen Ansatz, wenn Ihr Modell einen konstanten Durchsatz aufweist und Ihr Anwendungsfall Ausfallzeiten von 10-20 Minuten tolerieren kann. Ein Beispiel wäre, wenn Sie Ihr Modell mithilfe eines wöchentlichen Zeitplans stapelweise aufrufen möchten.

Automatische Skalierung von Inferenzeinheiten

Wenn Ihr Modell Nachfragespitzen bewältigen muss, kann Amazon Rekognition Custom Labels die Anzahl der Inferenzeinheiten, die Ihr Modell verwendet, automatisch skalieren. Bei steigender Nachfrage fügt Amazon Rekognition Custom Labels dem Modell zusätzliche Inferenzeinheiten hinzu und entfernt sie, wenn die Nachfrage sinkt.

Damit Amazon Rekognition Custom Labels automatisch Inferenzeinheiten für ein Modell skalieren kann, starten Sie das Modell und legen Sie mithilfe des `MaxInferenceUnits`-Parameters die maximale Anzahl von Inferenzeinheiten fest, die es verwenden kann. Durch die Festlegung einer maximalen Anzahl von Inferenzeinheiten können Sie die Kosten für den Betrieb des Modells verwalten, indem Sie die Anzahl der verfügbaren Inferenzeinheiten einschränken. Wenn Sie keine maximale Anzahl von Einheiten angeben, skaliert Amazon Rekognition Custom Labels Ihr Modell nicht automatisch, sondern verwendet nur die Anzahl der Inferenzeinheiten, mit der Sie begonnen haben. Informationen zur maximalen Anzahl von Inferenzeinheiten finden Sie unter Service Quotas.

Mithilfe des `MinInferenceUnits`-Parameters können Sie auch eine Mindestanzahl von Inferenzeinheiten angeben. Auf diese Weise können Sie den Mindestdurchsatz für Ihr Modell angeben, wobei eine einzelne Inferenzeinheit einer Stunde Verarbeitungszeit entspricht.

> **Note**
>
> Sie können die maximale Anzahl von Inferenzeinheiten nicht mit der Amazon Rekognition Custom Labels-Konsole festlegen. Geben Sie stattdessen den `MaxInferenceUnits` Eingabeparameter für den `StartProjectVersion`-Vorgang an.

Amazon Rekognition Custom Labels bietet die folgenden Amazon CloudWatch Logs-Metriken, anhand derer Sie den aktuellen Status der automatischen Skalierung für ein Modell ermitteln können.

Metrik	Beschreibung
`DesiredInferenceUnits`	Die Anzahl der Inferenzeinheiten, auf die Amazon Rekognition Custom Labels nach oben oder unten skaliert.
`InServiceInferenceUnits`	Die Anzahl der Inferenzeinheiten, die das Modell verwendet.

Wenn `DesiredInferenceUnits` = `InServiceInferenceUnits`, skaliert Amazon Rekognition Custom Labels derzeit nicht die Anzahl der Inferenzeinheiten.

Wenn `DesiredInferenceUnits` > `InServiceInferenceUnits`, skaliert Amazon Rekognition Custom Labels auf den Wert von `DesiredInferenceUnits`.

Wenn `DesiredInferenceUnits` < `InServiceInferenceUnits` wird Amazon Rekognition Custom Labels auf den Wert von `DesiredInferenceUnits` herunterskaliert.

Weitere Informationen zu den von Amazon Rekognition Custom Labels zurückgegebenen Metriken und Filterdimensionen finden Sie unter CloudWatch Metriken für Rekognition.

Rufen Sie `DescribeProjectsVersion` auf und überprüfen Sie das `MaxInferenceUnits`-Feld in der Antwort, um die maximale Anzahl von Inferenzeinheiten zu ermitteln, die Sie für ein Modell angefordert haben. Beispielcode finden Sie unter Beschreibung eines Modells (SDK).

Availability Zones

Amazon Rekognition Custom Labels verteilt Inferenzeinheiten über mehrere Availability Zone (AZ) innerhalb einer AWS Region, um die Verfügbarkeit zu erhöhen. Weitere Informationen finden Sie unter Availability Zones (AZ). Um Ihre Produktionsmodelle vor Ausfällen in der Availability Zone (AZ) und vor Ausfällen von Inferenzeinheiten zu schützen, starten Sie Ihre Produktionsmodelle mit mindestens zwei Inferenzeinheiten.

Bei einem Ausfall der Availability Zone (AZ) sind alle Inferenzeinheiten in der Availability Zone (AZ) nicht verfügbar und die Modellkapazität wird reduziert. Aufrufe von DetectCustomLabels werden auf die verbleibenden Inferenzeinheiten umverteilt. Solche Aufrufe sind erfolgreich, wenn sie die unterstützten Transaktionen pro Sekunde (TPS) der verbleibenden Inferenzeinheiten nicht

überschreiten. Nachdem AWS die Availability Zone (AZ) repariert hat, werden die Inferenzeinheiten neu gestartet und die volle Kapazität wiederhergestellt.

Wenn eine einzelne Inferenzeinheit ausfällt, startet Amazon Rekognition Custom Labels automatisch eine neue Inferenzeinheit in derselben Availability Zone (AZ). Die Modellkapazität wird reduziert, bis die neue Inferenzeinheit gestartet wird.

Starten eines Amazon Rekognition Custom Labels-Modells

Sie können mit der Ausführung eines Amazon Rekognition Custom Labels-Modells beginnen, indem Sie die Konsole oder den Vorgang StartProjectVersion verwenden.

> **Important**
>
> Ihnen werden die Anzahl der Stunden, die Ihr Modell läuft, und die Anzahl der Inferenzeinheiten, die Ihr Modell während des Betriebs verwendet, in Rechnung gestellt. Weitere Informationen finden Sie unter Ausführen eines trainierten Amazon Rekognition Custom Labels-Modells.

Das Starten eines Modells kann einige Minuten dauern. Um den aktuellen Status der Modellreife zu überprüfen, besuchen Sie die Detailseite für das Projekt oder verwenden Sie DescribeProject Versionen.

Nachdem das Modell gestartet wurde, verwenden Sie DetectCustomLabels, um Bilder anhand des Modells zu analysieren. Weitere Informationen finden Sie unter Analysieren eines Bildes mit einem trainierten Modell. Die Konsole bietet auch Beispielcode zum Aufrufen von `DetectCustomLabels`.

Themen

- Starten eines Amazon Rekognition Custom Labels-Modells (Konsole)
- Starten eines Amazon Rekognition Custom Labels-Modells (SDK)

Starten eines Amazon Rekognition Custom Labels-Modells (Konsole)

Gehen Sie wie folgt vor, um ein Amazon Rekognition Custom Labels-Modell mit der Konsole auszuführen. Sie können das Modell direkt von der Konsole aus starten oder den von der Konsole bereitgestellten AWS SDK-Code verwenden.

So starten Sie ein Modell (Konsole)

1. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Benutzerdefinierte Labels verwenden.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie auf der Ressourcenseite Projekte das Projekt aus, das das trainierte Modell enthält, das Sie starten möchten.
6. Wählen Sie im Abschnitt Modelle das Modell aus, das Sie starten möchten.
7. Wählen Sie die Registerkarte Modell verwenden.
8. Führen Sie eine der folgenden Aktionen aus:

   Start model using the console

   Gehen Sie im Abschnitt Modell starten oder stoppen wie folgt vor:

   1. Wählen Sie die Anzahl der Inferenzeinheiten aus, die Sie verwenden möchten. Weitere Informationen finden Sie unter Ausführen eines trainierten Amazon Rekognition Custom Labels-Modells.
   2. Wählen Sie Starten.
   3. Wählen Sie im Dialogfeld Modell starten die Option Starten aus.

   Start model using the AWS SDK

   Gehen Sie im Abschnitt Modell verwenden wie folgt vor:

   1. Wählen Sie API-Code.
   2. Wählen Sie entweder AWS CLI oder Python.
   3. Kopieren Sie in Modell starten den Beispielcode.
   4. Verwenden Sie den Beispielcode, um Ihr Modell zu starten. Weitere Informationen finden Sie unter Starten eines Amazon Rekognition Custom Labels-Modells (SDK).
9. Um zur Projektübersichtsseite zurückzukehren, wählen Sie oben auf der Seite Ihren Projektnamen aus.

10. Überprüfen Sie im Abschnitt Modell den Status des Modells. Wenn der Modellstatus WIRD AUSGEFÜHRT lautet, können Sie das Modell zur Analyse von Bildern verwenden. Weitere Informationen finden Sie unter Analysieren eines Bildes mit einem trainierten Modell.

Starten eines Amazon Rekognition Custom Labels-Modells (SDK)

Sie starten ein Modell, indem Sie die StartProjectVersions-API aufrufen und den Amazon-Ressourcennamen (ARN) des Modells im `ProjectVersionArn` Eingabeparameter übergeben. Sie geben auch die Anzahl der Inferenzeinheiten an, die Sie verwenden möchten. Weitere Informationen finden Sie unter Ausführen eines trainierten Amazon Rekognition Custom Labels-Modells.

Es kann eine Weile dauern, bis ein Modell gestartet wird. Die Python- und Java-Beispiele in diesem Thema verwenden Waiter, um auf den Start des Modells zu warten. Waiter sind Hilfsprogrammmethoden, die einen bestimmten Status abfragen. Alternativ können Sie den aktuellen Status überprüfen, indem Sie DescribeProjectVersions aufrufen.

So starten Sie ein Modell (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Beispielcode, um ein Modell zu starten.

   CLI

   Ändern Sie den Wert von `project-version-arn` in den ARN des Modells, das Sie starten möchten. Ändern Sie den Wert von `--min-inference-units` in die Anzahl der Inferenzeinheiten, die Sie verwenden möchten. Ändern Sie optional `--max-inference-units` in die maximale Anzahl von Inferenzeinheiten, die Amazon Rekognition Custom Labels verwenden kann, um das Modell automatisch zu skalieren.

```
aws rekognition start-project-version  --project-version-arn model_arn \
   --min-inference-units minimum number of units \
   --max-inference-units maximum number of units \
   --profile custom-labels-access
```

Python

Geben Sie die folgenden Befehlszeilenparameter an:

- `project_arn` — der ARN des Projekts, das das Modell enthält, das Sie starten möchten.
- `model_arn` — die ARN des Modells, das Sie starten möchten.
- `min_inference_units` — die Anzahl der Inferenzeinheiten, die Sie verwenden möchten.
- (Optional) `--max_inference_units` Die maximale Anzahl von Inferenzeinheiten, die Amazon Rekognition Custom Labels für die automatische Skalierung des Modells verwenden kann.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to start running an Amazon Lookout for Vision model.
"""

import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def get_model_status(rek_client, project_arn, model_arn):
    """
    Gets the current status of an Amazon Rekognition Custom Labels model
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that you want to use.
    :param model_arn:  The name of the model that you want the status for.
    :return: The model status
    """

    logger.info("Getting status for %s.", model_arn)

    # Extract the model version from the model arn.
    version_name = (model_arn.split("version/", 1)[1]).rpartition('/')[0]
```

```
    models = rek_client.describe_project_versions(ProjectArn=project_arn,
                                                  VersionNames=[version_name])

    for model in models['ProjectVersionDescriptions']:

        logger.info("Status: %s", model['StatusMessage'])
        return model["Status"]

    error_message = f"Model {model_arn} not found."
    logger.exception(error_message)
    raise Exception(error_message)


def start_model(rek_client, project_arn, model_arn, min_inference_units,
 max_inference_units=None):
    """
    Starts the hosting of an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that contains the
    model that you want to start hosting.
    :param min_inference_units: The number of inference units to use for
 hosting.
    :param max_inference_units: The number of inference units to use for auto-
scaling
    the model. If not supplied, auto-scaling does not happen.
    """

    try:
        # Start the model
        logger.info(f"Starting model: {model_arn}. Please wait....")

        if max_inference_units is None:
            rek_client.start_project_version(ProjectVersionArn=model_arn,

 MinInferenceUnits=int(min_inference_units))
        else:
            rek_client.start_project_version(ProjectVersionArn=model_arn,
                                             MinInferenceUnits=int(
                                                 min_inference_units),

 MaxInferenceUnits=int(max_inference_units))

        # Wait for the model to be in the running state
```

```
        version_name = (model_arn.split("version/", 1)[1]).rpartition('/')[0]
        project_version_running_waiter = rek_client.get_waiter(
            'project_version_running')
        project_version_running_waiter.wait(
            ProjectArn=project_arn, VersionNames=[version_name])

        # Get the running status
        return get_model_status(rek_client, project_arn, model_arn)

    except ClientError as err:
        logger.exception("Client error: Problem starting model: %s", err)
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains that the model
 you want to start."
    )
    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to start."
    )
    parser.add_argument(
        "min_inference_units", help="The minimum number of inference units to
 use."
    )
    parser.add_argument(
        "--max_inference_units",  help="The maximum number of inference units to
 use for auto-scaling the model.", required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:
```

```
        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Start the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status = start_model(rekognition_client,
                             args.project_arn, args.model_arn,
                             args.min_inference_units,
                             args.max_inference_units)

        print(f"Finished starting model: {args.model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        error_message = f"Client error: Problem starting model: {err}"
        logger.exception(error_message)
        print(error_message)

    except Exception as err:
        error_message = f"Problem starting model:{err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

Geben Sie die folgenden Befehlszeilenparameter an:

- `project_arn` — der ARN des Projekts, das das Modell enthält, das Sie starten möchten.
- `model_arn` — die ARN des Modells, das Sie starten möchten.
- `min_inference_units` — die Anzahl der Inferenzeinheiten, die Sie verwenden möchten.
- (Optional) `max_inference_units` — Die maximale Anzahl von Inferenzeinheiten, die Amazon Rekognition Custom Labels verwenden kann, um das Modell automatisch zu skalieren. Wenn Sie keinen Wert angeben, erfolgt keine automatische Skalierung.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.ProjectVersionStatus;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.StartProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StartProjectVersionResponse;
import software.amazon.awssdk.services.rekognition.waiters.RekognitionWaiter;

import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;

public class StartModel {

    public static final Logger logger =
 Logger.getLogger(StartModel.class.getName());



    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
```

```
        return start;

    }

    public static void startMyModel(RekognitionClient rekClient, String
projectArn, String modelArn,
            Integer minInferenceUnits, Integer maxInferenceUnits
            ) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Starting model: {0}", modelArn);

            StartProjectVersionRequest startProjectVersionRequest = null;

            if (maxInferenceUnits == null) {
                startProjectVersionRequest =
StartProjectVersionRequest.builder()
                    .projectVersionArn(modelArn)
                    .minInferenceUnits(minInferenceUnits)
                    .build();
            }
            else {
                startProjectVersionRequest =
StartProjectVersionRequest.builder()
                        .projectVersionArn(modelArn)
                        .minInferenceUnits(minInferenceUnits)
                        .maxInferenceUnits(maxInferenceUnits)
                        .build();

            }

            StartProjectVersionResponse response =
rekClient.startProjectVersion(startProjectVersionRequest);

            logger.log(Level.INFO, "Status: {0}", response.statusAsString() );


            // Get the model version

            int start = findForwardSlash(modelArn, 3) + 1;
            int end = findForwardSlash(modelArn, 4);

            String versionName = modelArn.substring(start, end);
```

```
            // wait until model starts

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                    .versionNames(versionName)
                    .projectArn(projectArn)
                    .build();

            RekognitionWaiter waiter = rekClient.waiter();

            WaiterResponse<DescribeProjectVersionsResponse> waiterResponse =
waiter

.waitUntilProjectVersionRunning(describeProjectVersionsRequest);

            Optional<DescribeProjectVersionsResponse> optionalResponse =
waiterResponse.matched().response();

            DescribeProjectVersionsResponse describeProjectVersionsResponse =
optionalResponse.get();

            for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                    .projectVersionDescriptions()) {
                if(projectVersionDescription.status() ==
ProjectVersionStatus.RUNNING) {
                    logger.log(Level.INFO, "Model is running" );

                }
                else {
                    String error = "Model training failed: " +
projectVersionDescription.statusAsString() + " "
                            + projectVersionDescription.statusMessage() + " " +
modelArn;
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);
                }

            }


        } catch (RekognitionException e) {
```

```
            logger.log(Level.SEVERE, "Could not start model: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String modelArn = null;
        String projectArn = null;
        Integer minInferenceUnits = null;
        Integer maxInferenceUnits = null;




        final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>
<min_inference_units> <max_inference_units>\n\n" + "Where:\n"
                + "   project_arn - The ARN of the project that contains the
model that you want to start. \n\n"
                + "   model_arn - The ARN of the model version that you want to
start.\n\n"
                + "   min_inference_units - The number of inference units to
start the model with.\n\n"
                + "   max_inference_units - The maximum number of inference
units that Custom Labels can use to "
                + "   automatically scale the model. If the value is null,
automatic scaling doesn't happen.\n\n";

        if (args.length < 3  || args.length >4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        modelArn = args[1];
        minInferenceUnits=Integer.parseInt(args[2]);

        if (args.length == 4) {
            maxInferenceUnits = Integer.parseInt(args[3]);
        }

        try {
```

```
            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Start the model.
            startMyModel(rekClient, projectArn, modelArn, minInferenceUnits,
 maxInferenceUnits);


            System.out.println(String.format("Model started: %s", modelArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }



    }

}
```

Stoppen eines Amazon Rekognition Custom Labels-Modells

Sie können die Ausführung eines Amazon Rekognition Custom Labels-Modells beenden, indem Sie die Konsole oder den Vorgang StopProjectVersion verwenden.

Themen

- Stoppen eines Amazon Rekognition Custom Labels-Modells (Konsole)
- Stoppen eines Amazon Rekognition Custom Labels-Modells (SDK)

Stoppen eines Amazon Rekognition Custom Labels-Modells (Konsole)

Führen Sie die folgenden Schritte aus, um ein laufendes Amazon Rekognition Custom Labels-Modell mit der Konsole zu stoppen. Sie können das Modell direkt von der Konsole aus beenden oder den von der Konsole bereitgestellten AWS SDK-Code verwenden.

So stoppen Sie ein Modell (Konsole)

1. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Benutzerdefinierte Labels verwenden.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie auf der Seite Projekte das Projekt aus, das das trainierte Modell enthält, das Sie stoppen möchten.
6. Wählen Sie im Abschnitt Modelle das Modell aus, das Sie stoppen möchten.
7. Wählen Sie die Registerkarte Modell verwenden.
8. Stop model using the console

    1. Wählen Sie im Abschnitt Modell starten oder stoppen die Option Stoppen.
    2. Geben Sie im Dialogfeld Modell stoppen Stoppen ein, um zu bestätigen, dass Sie das Modell stoppen möchten.
    3. Wählen Sie Stoppen, um das Modell zu stoppen.

    Stop model using the AWS SDK

    Gehen Sie im Abschnitt Modell verwenden wie folgt vor:

    1. Wählen Sie API-Code.
    2. Wählen Sie entweder AWS CLI oder Python.
    3. Kopieren Sie in Modellen stoppen den Beispielcode.
    4. Verwenden Sie den Beispielcode, um Ihr Modell zu stoppen. Weitere Informationen finden Sie unter Stoppen eines Amazon Rekognition Custom Labels-Modells (SDK).
9. Wählen Sie oben auf der Seite Ihren Projektnamen, um zur Projektübersichtsseite zurückzukehren.

10. Überprüfen Sie im Abschnitt Modell den Status des Modells. Das Modell wurde gestoppt, wenn der Modellstatus GESTOPPT lautet.

Stoppen eines Amazon Rekognition Custom Labels-Modells (SDK)

Sie beenden ein Modell, indem Sie die StopProjectVersions-API aufrufen und den Amazon-Ressourcennamen (ARN) des Modells im `ProjectVersionArn` Eingabeparameter übergeben.

Es kann eine Weile dauern, bis ein Modell gestoppt wird. Um den aktuellen Status zu überprüfen, rufen Sie `DescribeProjectVersions` auf.

So stoppen Sie ein Modell (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Beispielcode, um ein laufendes Modell zu stoppen.

   CLI

   Ändern Sie den Wert von `project-version-arn` in den ARN der Modellversion, die Sie stoppen möchten.

   ```
   aws rekognition stop-project-version --project-version-arn "model arn" \
     --profile custom-labels-access
   ```

   Python

   Im folgenden Beispiel wird ein Modell gestoppt, das bereits ausgeführt wird.

   Geben Sie die folgenden Befehlszeilenparameter an:

   - `project_arn` — der ARN des Projekts, das das Modell enthält, das Sie stoppen möchten.
   - `model_arn` — den ARN des Modells, das Sie stoppen möchten.

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   ```

```
Purpose
Shows how to stop a running Amazon Lookout for Vision model.
"""

import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def get_model_status(rek_client, project_arn, model_arn):
    """
    Gets the current status of an Amazon Rekognition Custom Labels model
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name:  The name of the project that you want to use.
    :param model_arn:  The name of the model that you want the status for.
    """

    logger.info ("Getting status for %s.", model_arn)

    # Extract the model version from the model arn.
    version_name=(model_arn.split("version/",1)[1]).rpartition('/')[0]

    # Get the model status.
    models=rek_client.describe_project_versions(ProjectArn=project_arn,
    VersionNames=[version_name])

    for model in models['ProjectVersionDescriptions']:
        logger.info("Status: %s",model['StatusMessage'])
        return model["Status"]

    # No model found.
    logger.exception("Model %s not found.", model_arn)
    raise Exception("Model %s not found.", model_arn)


def stop_model(rek_client, project_arn, model_arn):
    """
    Stops a running Amazon Rekognition Custom Labels Model.
```

```
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to stop running.
    :param model_arn:  The ARN of the model (ProjectVersion) that you want to
stop running.
    """

    logger.info("Stopping model: %s", model_arn)

    try:
        # Stop the model.
        response=rek_client.stop_project_version(ProjectVersionArn=model_arn)

        logger.info("Status: %s", response['Status'])

        # stops when hosting has stopped or failure.
        status = ""
        finished = False

        while finished is False:

            status=get_model_status(rek_client, project_arn, model_arn)

            if status == "STOPPING":
                logger.info("Model stopping in progress...")
                time.sleep(10)
                continue
            if status == "STOPPED":
                logger.info("Model is not running.")
                finished = True
                continue

            error_message = f"Error stopping model. Unexepected state: {status}"
            logger.exception(error_message)
            raise Exception(error_message)

        logger.info("finished. Status %s", status)
        return status

    except ClientError as err:
        logger.exception("Couldn't stop model - %s: %s",
           model_arn,err.response['Error']['Message'])
        raise
```

```
def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
 you want to stop."
    )
    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to stop."
    )

def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Stop the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status=stop_model(rekognition_client, args.project_arn, args.model_arn)

        print(f"Finished stopping model: {args.model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        logger.exception("Problem stopping model:%s",err)
        print(f"Failed to stop model: {err}")

    except Exception as err:
        logger.exception("Problem stopping model:%s", err)
        print(f"Failed to stop model: {err}")

if __name__ == "__main__":
    main()
```

Java V2

Geben Sie die folgenden Befehlszeilenparameter an:

- `project_arn` — der ARN des Projekts, das das Modell enthält, das Sie stoppen möchten.
- `model_arn` — den ARN des Modells, das Sie stoppen möchten.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.ProjectVersionStatus;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.StopProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.StopProjectVersionResponse;


import java.util.logging.Level;
import java.util.logging.Logger;

public class StopModel {

    public static final Logger logger =
 Logger.getLogger(StopModel.class.getName());

    public static int findForwardSlash(String modelArn, int n) {
```

```
        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;

    }

    public static void stopMyModel(RekognitionClient rekClient, String
projectArn, String modelArn)
            throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Stopping {0}", modelArn);

            StopProjectVersionRequest stopProjectVersionRequest =
StopProjectVersionRequest.builder()
                    .projectVersionArn(modelArn).build();

            StopProjectVersionResponse response =
rekClient.stopProjectVersion(stopProjectVersionRequest);

            logger.log(Level.INFO, "Status: {0}", response.statusAsString());

            // Get the model version

            int start = findForwardSlash(modelArn, 3) + 1;
            int end = findForwardSlash(modelArn, 4);

            String versionName = modelArn.substring(start, end);

            // wait until model stops

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                    .projectArn(projectArn).versionNames(versionName).build();

            boolean stopped = false;

            // Wait until create finishes
```

```
            do {

                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

.describeProjectVersions(describeProjectVersionsRequest);

                for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                        .projectVersionDescriptions()) {

                    ProjectVersionStatus status =
projectVersionDescription.status();

                    logger.log(Level.INFO, "stopping model: {0} ", modelArn);

                    switch (status) {

                    case STOPPED:
                        logger.log(Level.INFO, "Model stopped");
                        stopped = true;
                        break;

                    case STOPPING:
                        Thread.sleep(5000);
                        break;

                    case FAILED:
                        String error = "Model stopping failed: " +
projectVersionDescription.statusAsString() + " "
                                + projectVersionDescription.statusMessage() + "
" + modelArn;
                        logger.log(Level.SEVERE, error);
                        throw new Exception(error);

                    default:
                        String unexpectedError = "Unexpected stopping state: "
                                + projectVersionDescription.statusAsString() + "
"
                                + projectVersionDescription.statusMessage() + "
" + modelArn;
                        logger.log(Level.SEVERE, unexpectedError);
                        throw new Exception(unexpectedError);
                    }
```

```
                }

            } while (stopped == false);

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not stop model: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String modelArn = null;
        String projectArn = null;


        final String USAGE = "\n" + "Usage: " + "<project_name> <version_name>\n
\n" + "Where:\n"
                + "   project_arn - The ARN of the project that contains the
 model that you want to stop. \n\n"
                + "   model_arn - The ARN of the model version that you want to
 stop.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        modelArn = args[1];


        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Stop model
```

```
            stopMyModel(rekClient, projectArn, modelArn);

            System.out.println(String.format("Model stopped: %s", modelArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Laufzeit und verwendete Inferenzeinheiten melden

Wenn Sie Ihr Modell nach August 2022 trainiert und gestartet haben, können Sie anhand der `InServiceInferenceUnits` CloudWatch Amazon-Metrik ermitteln, wie viele Stunden ein Modell gelaufen ist und wie viele Inferenzeinheiten in diesen Stunden verwendet wurden.

> **Note**
>
> Wenn Sie nur über ein Modell in einer AWS Region verfügen, können Sie die Laufzeit des Modells auch ermitteln, indem Sie erfolgreiche An `StartprojectVersion` - und `StopProjectVersion` Abrufe nachverfolgen. CloudWatch Dieser Ansatz funktioniert nicht, wenn Sie mehr als ein Modell in der AWS Region ausführen, da die Metriken keine Informationen über das Modell enthalten.
> Alternativ können Sie es verwenden, AWS CloudTrail um Anrufe an `StartProjectVersion` und zu verfolgen `StopProjectVersion` (was das Modell ARN im `requestParameters` Feld der Ereignishistorie beinhaltet). CloudTrail Ereignisse sind auf 90 Tage begrenzt, aber Sie können Ereignisse für bis zu 7 Jahre in einem CloudTrailLake speichern.

Mit dem folgenden Verfahren werden Diagramme für Folgendes erstellt:

- Die Anzahl der Stunden, während der ein Modell gelaufen ist.
- Die Anzahl der Inferenzeinheiten, die ein Modell verwendet hat.

Sie können einen Zeitraum von bis zu 15 Monaten in der Vergangenheit wählen. Weitere Informationen zur Aufbewahrung von Metriken finden Sie unter Aufbewahrung von Metriken.

So bestimmen Sie die Modelldauer und die für ein Modell verwendeten Inferenzeinheiten

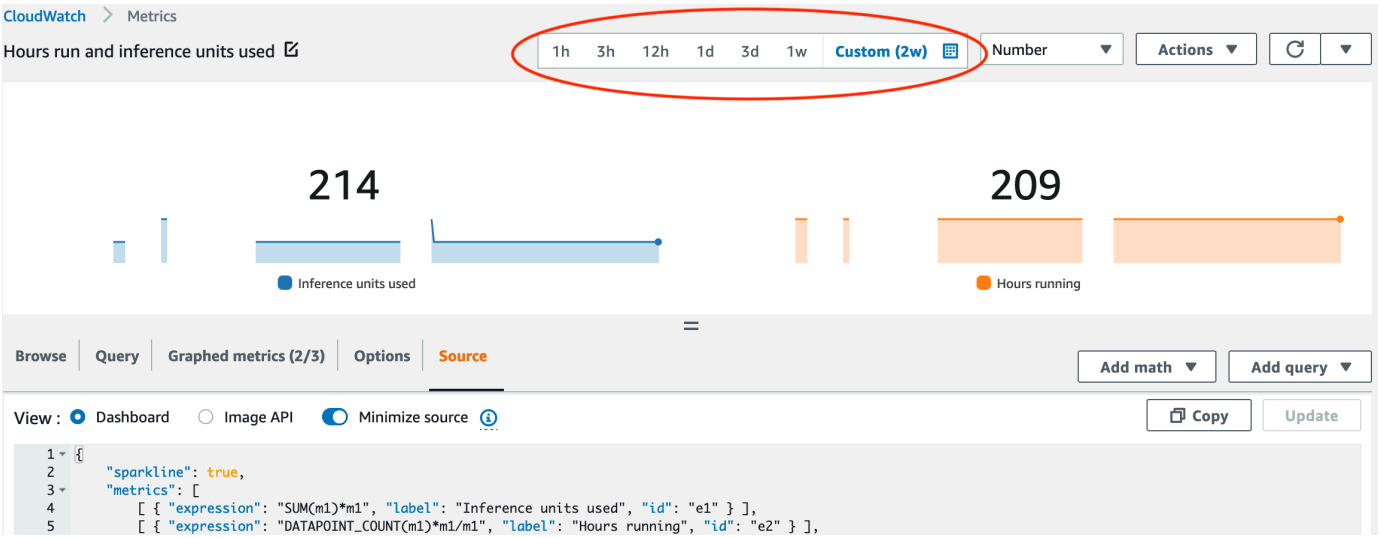
1. Melden Sie sich bei der an AWS Management Console und öffnen Sie die CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im linken Navigationsbereich unter Metriken Alle Metriken aus.
3. Wählen Sie im Bereich „Inhalt“ die Registerkarte Quelle aus.
4. Vergewissern Sie sich, dass die Dashboard-Schaltfläche ausgewählt ist.
5. Ersetzen Sie im Bearbeitungsfeld die vorhandene JSON durch die folgende JSON: Ändern Sie die folgenden Werte:

   - `Project_Name` — Das Projekt, das das Modell enthält, das Sie grafisch darstellen möchten.
   - `Version_Name` — Die Version des Modells, das Sie grafisch darstellen möchten.
   - `AWS_Region`— Die AWS Region, in der das Modell enthalten ist. Stellen Sie sicher, dass sich die CloudWatch Konsole in derselben AWS Region befindet. Überprüfen Sie dazu die Regionsauswahl in der Navigationsleiste oben auf der Seite. Aktualisieren Sie nach Bedarf.

```
{
    "sparkline": true,
    "metrics": [
        [
            {
                "expression": "SUM(m1)*m1",
                "label": "Inference units used",
                "id": "e1"
            }
        ],
        [
            {
                "expression": "DATAPOINT_COUNT(m1)*m1/m1",
                "label": "Hours running",
```

```
                "id": "e2"
            }
        ],
        [
            "AWS/Rekognition",
            "InServiceInferenceUnits",
            "ProjectName",
            "Project_Name",
            "VersionName",
            "Version_Name",
            {
                "id": "m1",
                "visible": false
            }
        ]
    ],
    "view": "singleValue",
    "stacked": false,
    "region": "AWS_Region",
    "stat": "Average",
    "period": 3600,
    "title": "Hours run and inference units used"
}
```

6. Wählen Sie Aktualisieren.
7. Wählen Sie oben auf der Seite eine Zeitleiste aus. Während der Zeitleiste sollten Sie Zahlen für die verwendeten Inferenzeinheiten und die laufenden Stunden sehen. Lücken in der Grafik weisen auf Zeiten hin, in denen das Modell nicht lief. Der Screenshot der Konsole unten zeigt die verwendeten Inferenzeinheiten und die Betriebsstunden über Zeiträume, wobei eine benutzerdefinierte Zeit von 2 Wochen festgelegt ist, wobei die höchsten Werte bei 214 Inferenzeinheiten und 209 Betriebsstunden liegen.



8. (Optional) Um das Diagramm einem Dashboard hinzuzufügen, wählen Sie Aktionen gefolgt von Zum Dashboard hinzufügen - verbessert aus.

Analysieren eines Bildes mit einem trainierten Modell

Um ein Bild mit einem trainierten Amazon Rekognition Custom Labels-Modell zu analysieren, rufen Sie die DetectCustomLabels API auf. Das Ergebnis von `DetectCustomLabels` ist eine Vorhersage, dass das Bild bestimmte Objekte, Szenen oder Konzepte enthält.

Sie müssen Folgendes angeben, um `DetectCustomLabels` aufzurufen:

- Den Amazon-Ressourcenname (ARN) des Amazon Rekognition Custom Labels-Modells, das Sie verwenden möchten.
- Das Bild, mit dem das Modell eine Vorhersage treffen soll. Sie können ein Eingabebild als Bild-Byte-Array (base64-verschlüsselte Bild-Bytes) oder als Amazon-S3-Objekt zur Verfügung stellen. Weitere Informationen finden Sie unter Bild.

Benutzerdefinierte Labels werden in einer Anordnung von Custom Label-Objekten zurückgegeben. Jedes benutzerdefinierte Label steht für ein einzelnes Objekt, eine Szene oder ein einzelnes Konzept im Bild. Ein benutzerdefiniertes Label umfasst:

- Eine Bezeichnung für das Objekt, die Szene oder das Konzept im Bild.
- Einen Begrenzungsrahmen für Objekte, die im Bild gefunden wurden. Der Begrenzungsrahmen zeigen die Position des Objekts auf dem Quellbild an. Die Koordinatenwerte sind ein Verhältnis der gesamten Bildgröße. Weitere Informationen finden Sie unter. BoundingBox `DetectCustomLabels`gibt nur dann Begrenzungsrahmen zurück, wenn das Modell darauf trainiert wurde, Objektpositionen zu erkennen.
- Das Vertrauen, das Amazon Rekognition Custom Labels in die Richtigkeit des Labels und des Begrenzungsrahmens hat.

Um Labels auf der Grundlage der Erkennungssicherheit zu filtern, geben Sie einen Wert für `MinConfidence` an, der Ihrem gewünschten Konfidenzniveau entspricht. Wenn Sie sich der Vorhersage beispielsweise sehr sicher sein müssen, geben Sie einen hohen Wert für `MinConfidence` an. Geben Sie den `MinConfidence` Wert 0 an, um alle Labels unabhängig von der Konfidenz zu erhalten.

Die Leistung Ihres Modells wird teilweise anhand der Rückruf- und Präzisionsmesswerte gemessen, die während des Modelltrainings berechnet wurden. Weitere Informationen finden Sie unter Metriken für die Bewertung Ihres Modells.

Um die Genauigkeit Ihres Modells zu erhöhen, legen Sie einen höheren Wert für `MinConfidence` fest. Weitere Informationen finden Sie unter Reduzierung falsch positiver Ergebnisse (höhere Präzision).

Um die Rückrufrate Ihres Modells zu erhöhen, verwenden Sie einen niedrigeren Wert für `MinConfidence`. Weitere Informationen finden Sie unter Reduzierung falsch negativer Ergebnisse (besseres Erinnerungsvermögen).

Wenn Sie keinen Wert für `MinConfidence` angeben, gibt Amazon Rekognition Custom Labels ein Label zurück, das auf dem angenommenen Schwellenwert für dieses Label basiert. Weitere Informationen finden Sie unter Angenommener Schwellenwert. Sie können den Wert des angenommenen Schwellenwerts für ein Label aus den Trainingsergebnissen des Modells ermitteln. Weitere Informationen finden Sie unter Ein Model trainieren (Konsole).

Mithilfe des `MinConfidence` Eingabeparameters geben Sie einen gewünschten Schwellenwert für den Abruf an. Labels, deren Konfidenzniveau unter dem Wert von `MinConfidence` liegt, werden in der Antwort nicht zurückgegeben. Außerdem wirkt sich der angenommene Schwellenwert für ein Label nicht auf die Aufnahme des Labels in die Antwort aus.

> **Note**
>
> Die Amazon Rekognition Custom Labels-Metriken drücken einen angenommenen Schwellenwert als Gleitkommazahl zwischen 0 und 1 aus. Der Bereich von `MinConfidence` normalisiert den Schwellenwert auf einen Prozentwert (0-100). Konfidenzwerte von DetectCustomLabels werden ebenfalls als Prozentsatz zurückgegeben.

Legen Sie für spezifische Labels ggf. eine Variable fest. Wenn beispielsweise die Genauigkeitsmetrik für Label A akzeptabel ist, aber nicht für Label B. Beachten Sie bei der Angabe eines anderen Schwellenwerts (`MinConfidence`) Folgendes.

- Wenn Sie nur an einem einzigen Label (A) interessiert sind, setzen Sie den Wert von `MinConfidence` auf den gewünschten Schwellenwert. In der Antwort werden Vorhersagen für Label A (zusammen mit anderen Labels) nur zurückgegeben, wenn die Konfidenz größer als ist `MinConfidence`. Sie müssen alle anderen zurückgegebenen Labels herausfiltern.
- Befolgen Sie folgende Schritte, wenn Sie verschiedene Schwellenwerte auf mehrere Labels anwenden möchten:

1. Verwenden Sie den Wert 0 für `MinConfidence`. Ein Wert von 0 stellt sicher, dass alle Labels zurückgegeben werden, unabhängig von der Erkennungssicherheit.
2. Wenden Sie für jedes zurückgegebene Label den gewünschten Schwellenwert an, indem Sie überprüfen, ob die Label-Konfidenz höher ist als der Schwellenwert, den Sie für das Label wünschen.

Weitere Informationen finden Sie unter Verbessern eines geschulten Amazon Rekognition Custom Labels-Modells.

Wenn Sie feststellen, dass die von `DetectCustomLabels` zurückgegebenen Konfidenzwerte zu niedrig sind, sollten Sie das Modell erneut trainieren. Weitere Informationen finden Sie unter Schulung eines Amazon-Rekognition-Custom-Labels-Modells. Sie können die Anzahl der `DetectCustomLabels` zurückgegebenen und benutzerdefinierten Labels einschränken, indem Sie den `MaxResults` Eingabeparameter angeben. Die Ergebnisse werden sortiert von der höchsten bis zur niedrigsten Konfidenz zurückgegeben.

Weitere Beispiele, die `DetectCustomLabels` aufrufen, finden Sie unter Beispiele.

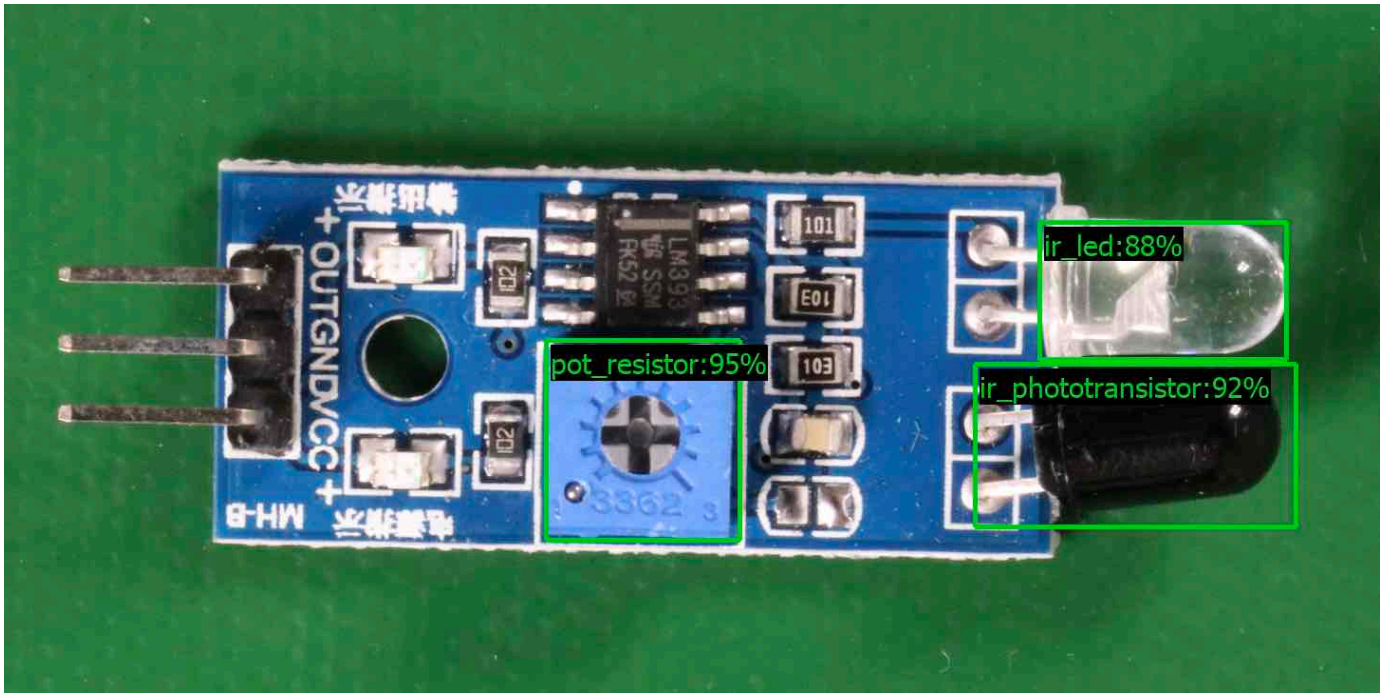
Weitere Informationen zum Sichern von `DetectCustomLabels` finden Sie unter SicherungDetectCustomLabels.

So erkennen Sie benutzerdefinierte Labels (API)

1. Wenn Sie dies noch nicht getan haben:
   a. Stellen Sie sicher, dass Sie Berechtigungen für `DetectCustomLabels` und `AmazonS3ReadOnlyAccess` haben. Weitere Informationen finden Sie unter Einrichten von SDK-Berechtigungen.
   b. Installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Trainieren und implementieren Sie Ihr Modell. Weitere Informationen finden Sie unter Erstellen eines Amazon Rekognition-Custom-Custom-Label-Erstellen eines neuen.
3. Stellen Sie sicher, dass der Benutzer, der `DetectCustomLabels` aufruft, Zugang zu dem Modell, das Sie in Schritt 2 verwendet haben, hat. Weitere Informationen finden Sie unter SicherungDetectCustomLabels.
4. Laden Sie ein Bild hoch, das Sie für ein S3-Bucket analysieren möchten.

Weitere Anleitungen finden Sie unter Upload eines Objekts in Amazon S3 im Benutzerhandbuch für Amazon Simple Storage Service. Die Beispiele für Python, Java und Java 2 zeigen Ihnen auch, wie Sie eine lokale Bilddatei verwenden, um ein Bild mithilfe von Raw-Bytes zu übermitteln. Die Datei muss kleiner als 4 MB sein.

5. Verwenden Sie die folgenden Beispiele zum Aufrufen der `DetectCustomLabels`-Operation. Die Python- und Java-Beispiele zeigen das Bild und überlagern die Analyseergebnisse, ähnlich wie in der folgenden Abbildung. Die folgenden Bilder enthalten Begrenzungsfelder und Beschriftungen für eine Leiterplatte mit einem Potentiometer, einem Infrarot-Phototransistor und LED-Komponenten.



AWS CLI

Dieser AWS CLI Befehl zeigt die JSON-Ausgabe für den `DetectCustomLabels` CLI-Vorgang an. Ändern Sie die Werte der folgenden Eingabeparameter.

- `bucket` durch den Namen des Amazon-S3-Buckets, den Sie in Schritt 4 verwendet haben.
- `image` durch den Namen der Eingabebilddatei, die Sie in Schritt 4 hochgeladen haben.
- `projectVersionArn` durch den ARN des Modells, das Sie verwenden möchten.

```
aws rekognition detect-custom-labels --project-version-arn model_arn \
   --image '{"S3Object":{"Bucket":"bucket","Name":"image"}}' \
```

```
    --min-confidence 70 \
    --profile custom-labels-access
```

Python

Der folgende Beispielcode zeigt Begrenzungsrahmen und Labels auf Bildebene eines Bildes an.

Um ein lokales Bild zu analysieren, führen Sie das Programm aus und geben Sie die folgenden Befehlszeilenargumente ein:

- Der ARN des Modells, mit dem Sie das Bild analysieren möchten.
- Der Name und der Speicherort einer lokalen Bilddatei.

Um ein in einem Amazon-S3-Bucket gespeichertes Bild zu analysieren, führen Sie das Programm aus und geben Sie die folgenden Befehlszeilenargumente ein:

- Der ARN des Modells, mit dem Sie das Bild analysieren möchten.
- Der Name und Speicherort eines Bildes innerhalb des Amazon-S3-Buckets, den Sie in Schritt 4 verwendet haben.
- `--bucket`*`Bucket-Name`* — Der Amazon-S3-Bucket, den Sie in Schritt 4 verwendet haben.

Beachten Sie, dass in diesem Beispiel davon ausgegangen wird, dass Ihre Version von Pillow >= 8.0.0 ist.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Amazon Rekognition Custom Labels detection example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/detecting-custom-
labels.html
Shows how to detect custom labels by using an Amazon Rekognition Custom Labels
 model.
The image can be stored on your local computer or in an Amazon S3 bucket.
"""
```

```
import io
import logging
import argparse
import boto3
from PIL import Image, ImageDraw, ImageFont

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def analyze_local_image(rek_client, model, photo, min_confidence):
    """
    Analyzes an image stored as a local file.
    :param rek_client: The Amazon Rekognition Boto3 client.
    :param s3_connection: The Amazon S3 Boto3 S3 connection object.
    :param model: The ARN of the Amazon Rekognition Custom Labels model that you
 want to use.
    :param photo: The name and file path of the photo that you want to analyze.
    :param min_confidence: The desired threshold/confidence for the call.
    """

    try:
        logger.info("Analyzing local file: %s", photo)
        image = Image.open(photo)
        image_type = Image.MIME[image.format]

        if (image_type == "image/jpeg" or image_type == "image/png") is False:
            logger.error("Invalid image type for %s", photo)
            raise ValueError(
                f"Invalid file format. Supply a jpeg or png format file:
 {photo}"
            )

        # get images bytes for call to detect_anomalies
        image_bytes = io.BytesIO()
        image.save(image_bytes, format=image.format)
        image_bytes = image_bytes.getvalue()

        response = rek_client.detect_custom_labels(Image={'Bytes': image_bytes},
                                                   MinConfidence=min_confidence,
                                                   ProjectVersionArn=model)


        show_image(image, response)
```

```
        return len(response['CustomLabels'])

    except ClientError as client_err:
        logger.error(format(client_err))
        raise
    except FileNotFoundError as file_error:
        logger.error(format(file_error))
        raise


def analyze_s3_image(rek_client, s3_connection, model, bucket, photo,
 min_confidence):
    """
    Analyzes an image stored in the specified S3 bucket.
    :param rek_client: The Amazon Rekognition Boto3 client.
    :param s3_connection: The Amazon S3 Boto3 S3 connection object.
    :param model: The ARN of the Amazon Rekognition Custom Labels model that you
 want to use.
    :param bucket: The name of the S3 bucket that contains the image that you
 want to analyze.
    :param photo: The name of the photo that you want to analyze.
    :param min_confidence: The desired threshold/confidence for the call.
    """

    try:
        # Get image from S3 bucket.

        logger.info("analyzing bucket: %s image: %s", bucket, photo)
        s3_object = s3_connection.Object(bucket, photo)
        s3_response = s3_object.get()

        stream = io.BytesIO(s3_response['Body'].read())
        image = Image.open(stream)

        image_type = Image.MIME[image.format]

        if (image_type == "image/jpeg" or image_type == "image/png") is False:
            logger.error("Invalid image type for %s", photo)
            raise ValueError(
                f"Invalid file format. Supply a jpeg or png format file:
 {photo}")

        ImageDraw.Draw(image)
```

```
        # Call DetectCustomLabels.
        response = rek_client.detect_custom_labels(
            Image={'S3Object': {'Bucket': bucket, 'Name': photo}},
            MinConfidence=min_confidence,
            ProjectVersionArn=model)

        show_image(image, response)
        return len(response['CustomLabels'])

    except ClientError as err:
        logger.error(format(err))
        raise


def show_image(image, response):
    """
    Displays the analyzed image and overlays analysis results
    :param image: The analyzed image
    :param response: the response from DetectCustomLabels
    """
    try:
        font_size = 40
        line_width = 5

        img_width, img_height = image.size
        draw = ImageDraw.Draw(image)

        # Calculate and display bounding boxes for each detected custom label.
        image_level_label_height = 0

        for custom_label in response['CustomLabels']:
            confidence = int(round(custom_label['Confidence'], 0))
            label_text = f"{custom_label['Name']}:{confidence}%"
            fnt = ImageFont.truetype('Tahoma.ttf', font_size)
            text_left, text_top, text_right, text_bottom = draw.textbbox((0, 0),
 label_text, fnt)
            text_width, text_height = text_right - text_left, text_bottom -
 text_top

            logger.info("Label: %s", custom_label['Name'])
            logger.info("Confidence:  %s", confidence)

            # Draw bounding boxes, if present
            if 'Geometry' in custom_label:
```

```
                box = custom_label['Geometry']['BoundingBox']
                left = img_width * box['Left']
                top = img_height * box['Top']
                width = img_width * box['Width']
                height = img_height * box['Height']

                logger.info("Bounding box")
                logger.info("\tLeft: {0:.0f}".format(left))
                logger.info("\tTop: {0:.0f}".format(top))
                logger.info("\tLabel Width: {0:.0f}".format(width))
                logger.info("\tLabel Height: {0:.0f}".format(height))

                points = (
                    (left, top),
                    (left + width, top),
                    (left + width, top + height),
                    (left, top + height),
                    (left, top))
                # Draw bounding box and label text
                draw.line(points, fill="limegreen", width=line_width)
                draw.rectangle([(left + line_width, top+line_width),
                                (left + text_width + line_width, top +
 line_width + text_height)], fill="black")
                draw.text((left + line_width, top + line_width),
                          label_text, fill="limegreen", font=fnt)

            # draw image-level label text.
            else:
                draw.rectangle([(10, image_level_label_height),
                                (text_width + 10, image_level_label_height
+text_height)], fill="black")
                draw.text((10, image_level_label_height),
                          label_text, fill="limegreen", font=fnt)

                image_level_label_height += text_height

        image.show()

    except Exception as err:
        logger.error(format(err))
        raise


def add_arguments(parser):
```

```
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "model_arn", help="The ARN of the model that you want to use."
    )

    parser.add_argument(
        "image", help="The path and file name of the image that you want to
 analyze"
    )
    parser.add_argument(
        "--bucket",  help="The bucket that contains the image. If not supplied,
 image is assumed to be a local file.", required=False
    )


def main():

    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        label_count = 0
        min_confidence = 50

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        if args.bucket is None:
            # Analyze local image.
            label_count = analyze_local_image(rekognition_client,
                                              args.model_arn,
                                              args.image,
                                              min_confidence)
        else:
            # Analyze image in S3 bucket.
```

```
            s3_connection = session.resource('s3')
            label_count = analyze_s3_image(rekognition_client,
                                           s3_connection,
                                           args.model_arn,
                                           args.bucket,
                                           args.image,
                                           min_confidence)

        print(f"Custom labels detected: {label_count}")

    except ClientError as client_err:
        print("A service client error occurred: " +
              format(client_err.response["Error"]["Message"]))

    except ValueError as value_err:
        print("A value error occurred: " + format(value_err))

    except FileNotFoundError as file_error:
        print("File not found error: " + format(file_error))

    except Exception as err:
        print("An error occurred: " + format(err))


if __name__ == "__main__":
    main()
```

Java

Der folgende Beispielcode zeigt Begrenzungsrahmen und Labels auf Bildebene eines Bildes an.

Um ein lokales Bild zu analysieren, führen Sie das Programm aus und geben Sie die folgenden Befehlszeilenargumente ein:

- Der ARN des Modells, mit dem Sie das Bild analysieren möchten.
- Der Name und der Speicherort einer lokalen Bilddatei.

Um ein in einem Amazon-S3-Bucket gespeichertes Bild zu analysieren, führen Sie das Programm aus und geben Sie die folgenden Befehlszeilenargumente ein:

- Der ARN des Modells, mit dem Sie das Bild analysieren möchten.

- Der Name und Speicherort eines Bildes innerhalb des Amazon-S3-Buckets, den Sie in Schritt 4 verwendet haben.
- Der Amazon-S3-Bucket mit dem Bild, das Sie in Schritt 4 verwendet haben.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.amazonaws.samples;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.util.List;
import javax.imageio.ImageIO;
import javax.swing.*;
import java.io.FileNotFoundException;
import java.awt.font.FontRenderContext;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.nio.ByteBuffer;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;

import com.amazonaws.auth.AWSCredentialsProvider;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.rekognition.AmazonRekognition;
import com.amazonaws.services.rekognition.AmazonRekognitionClientBuilder;

import com.amazonaws.services.rekognition.model.BoundingBox;
import com.amazonaws.services.rekognition.model.CustomLabel;
import com.amazonaws.services.rekognition.model.DetectCustomLabelsRequest;
import com.amazonaws.services.rekognition.model.DetectCustomLabelsResult;
import com.amazonaws.services.rekognition.model.Image;
import com.amazonaws.services.rekognition.model.S3Object;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
```

```
import com.amazonaws.services.s3.model.S3ObjectInputStream;

import com.amazonaws.services.rekognition.model.AmazonRekognitionException;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.util.IOUtils;

// Calls DetectCustomLabels and displays a bounding box around each detected
 image.
public class DetectCustomLabels extends JPanel {

    private transient DetectCustomLabelsResult response;
    private transient Dimension dimension;
    private transient BufferedImage image;

    public static final Logger logger =
 Logger.getLogger(DetectCustomLabels.class.getName());

    // Finds custom labels in an image stored in an S3 bucket.
    public DetectCustomLabels(AmazonRekognition rekClient,
            AmazonS3 s3client,
            String projectVersionArn,
            String bucket,
            String key,
            Float minConfidence) throws AmazonRekognitionException,
 AmazonS3Exception, IOException {

        logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
 Object[] { bucket, key });

        // Get image from S3 bucket and create BufferedImage
        com.amazonaws.services.s3.model.S3Object s3object =
 s3client.getObject(bucket, key);
        S3ObjectInputStream inputStream = s3object.getObjectContent();
        image = ImageIO.read(inputStream);

        // Set image size
        setWindowDimensions();

        DetectCustomLabelsRequest request = new DetectCustomLabelsRequest()
                .withProjectVersionArn(projectVersionArn)
                .withImage(new Image().withS3Object(new
 S3Object().withName(key).withBucket(bucket)))
                .withMinConfidence(minConfidence);
```

```
        // Call DetectCustomLabels

        response = rekClient.detectCustomLabels(request);
        logFoundLabels(response.getCustomLabels());
        drawLabels();

    }

    // Finds custom label in a local image file.
    public DetectCustomLabels(AmazonRekognition rekClient,
            String projectVersionArn,
            String photo,
            Float minConfidence)
            throws IOException, AmazonRekognitionException {

        logger.log(Level.INFO, "Processing local file: {0}", photo);

        // Get image bytes and buffered image
        ByteBuffer imageBytes;
        try (InputStream inputStream = new FileInputStream(new File(photo))) {
            imageBytes = ByteBuffer.wrap(IOUtils.toByteArray(inputStream));
        }

        // Get image for display
        InputStream imageBytesStream;
        imageBytesStream = new ByteArrayInputStream(imageBytes.array());

        ByteArrayOutputStream baos = new ByteArrayOutputStream();
        image = ImageIO.read(imageBytesStream);
        ImageIO.write(image, "jpg", baos);

        // Set image size
        setWindowDimensions();

        // Analyze image
        DetectCustomLabelsRequest request = new DetectCustomLabelsRequest()
                .withProjectVersionArn(projectVersionArn)
                .withImage(new Image()
                        .withBytes(imageBytes))
                .withMinConfidence(minConfidence);

        response = rekClient.detectCustomLabels(request);

        logFoundLabels(response.getCustomLabels());
```

```
        drawLabels();

    }

    // Log the labels found by DetectCustomLabels
    private void logFoundLabels(List<CustomLabel> customLabels) {
        logger.info("Custom labels found");
        if (customLabels.isEmpty()) {
            logger.log(Level.INFO, "No Custom Labels found. Consider lowering
min confidence.");
        } else {
            for (CustomLabel customLabel : customLabels) {
                logger.log(Level.INFO, " Label: {0} Confidence: {1}",
                        new Object[] { customLabel.getName(),
customLabel.getConfidence() });
            }

        }
    }

    // Sets window dimensions to 1/2 screen size, unless image is smaller
    public void setWindowDimensions() {
        dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

        dimension.width = (int) dimension.getWidth() / 2;
        if (image.getWidth() < dimension.width) {
            dimension.width = image.getWidth();
        }
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getHeight() < dimension.height) {
            dimension.height = image.getHeight();
        }

        setPreferredSize(dimension);

    }

    // Draws the image containing the bounding boxes and labels.
    @Override
    public void paintComponent(Graphics g) {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.
```

```
        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

    public void drawLabels() {
        // Draws bounding boxes (if present) and label text.

        int boundingBoxBorderWidth = 5;
        int imageHeight = image.getHeight(this);
        int imageWidth = image.getWidth(this);

        // Set up drawing
        Graphics2D g2d = image.createGraphics();
        g2d.setColor(Color.GREEN);
        g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
        Font font = g2d.getFont();
        FontRenderContext frc = g2d.getFontRenderContext();
        g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

        List<CustomLabel> customLabels = response.getCustomLabels();

        int imageLevelLabelHeight = 0;
        for (CustomLabel customLabel : customLabels) {

            String label = customLabel.getName();

            int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
            int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());

            // Draw bounding box, if present
            if (customLabel.getGeometry() != null) {

                BoundingBox box = customLabel.getGeometry().getBoundingBox();
                float left = imageWidth * box.getLeft();
                float top = imageHeight * box.getTop();

                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
```

```
                        textWidth + boundingBoxBorderWidth, textHeight +
 boundingBoxBorderWidth);

                // Write label onto black rectangle
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, left + boundingBoxBorderWidth, (top +
 textHeight));

                // Draw bounding box around label location
                g2d.drawRect(Math.round(left), Math.round(top),
 Math.round((imageWidth * box.getWidth())),
                        Math.round((imageHeight * box.getHeight())));
            }
            // Draw image level labels.
            else {
                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
 textHeight);
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

                imageLevelLabelHeight += textHeight;
            }

        }
        g2d.dispose();

    }

    public static void main(String args[]) throws Exception {

        String photo = null;
        String bucket = null;
        String projectVersionArn = null;
        float minConfidence = 50;

        final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n
\n" + "Where:\n"
                + "   model_arn - The ARN of the model that you want to use. \n
\n"
                + "   image - The location of the image on your local file
 system or within an S3 bucket.\n\n"
```

```
                + "   bucket - The S3 bucket that contains the image. Don't
specify if image is local.\n\n";

       // Collect the arguments. If 3 arguments are present, the image is
assumed to be
       // in an S3 bucket.

       if (args.length < 2 || args.length > 3) {
           System.out.println(USAGE);
           System.exit(1);
       }

       projectVersionArn = args[0];
       photo = args[1];

       if (args.length == 3) {
           bucket = args[2];
       }

       DetectCustomLabels panel = null;

       try {

           AWSCredentialsProvider provider =new
ProfileCredentialsProvider("custom-labels-access");

           AmazonRekognition rekClient =
AmazonRekognitionClientBuilder.standard()
                   .withCredentials(provider)
                   .withRegion(Regions.US_WEST_2)
                   .build();


           AmazonS3 s3client = AmazonS3ClientBuilder.standard()
           .withCredentials(provider)
           .withRegion(Regions.US_WEST_2)
           .build();

           // Create frame and panel.
           JFrame frame = new JFrame("Custom Labels");
           frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

           if (args.length == 2) {
               // Analyze local image
```

```
                panel = new DetectCustomLabels(rekClient, projectVersionArn,
 photo, minConfidence);
            } else {
                // Analyze image in S3 bucket
                panel = new DetectCustomLabels(rekClient, s3client,
 projectVersionArn, bucket, photo, minConfidence);
            }

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);

        } catch (AmazonRekognitionException rekError) {
            String errorMessage = "Rekognition client error: " +
 rekError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (FileNotFoundException fileError) {
            String errorMessage = "File not found: " + photo;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (IOException fileError) {
            String errorMessage = "Input output exception: " +
 fileError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (AmazonS3Exception s3Error) {
            String errorMessage = "S3 error: " + s3Error.getErrorMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        }

    }
}
```

Java V2

Der folgende Beispielcode zeigt Begrenzungsrahmen und Labels auf Bildebene eines Bildes an.

Um ein lokales Bild zu analysieren, führen Sie das Programm aus und geben Sie die folgenden Befehlszeilenargumente ein:

- `projectVersionArn` — Der ARN des Modells, mit dem Sie das Bild analysieren möchten.
- `photo` — Der Name und der Speicherort einer lokalen Bilddatei.

Um ein in einem S3-Bucket gespeichertes Bild zu analysieren, führen Sie das Programm aus und geben Sie die folgenden Befehlszeilenargumente ein:

- Der ARN des Modells, mit dem Sie das Bild analysieren möchten.
- Der Name und Speicherort eines Bildes innerhalb des Amazon-S3-Buckets, den Sie in Schritt 4 verwendet haben.
- Der Amazon-S3-Bucket mit dem Bild, das Sie in Schritt 4 verwendet haben.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.core.sync.ResponseTransformer;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.S3Object;
import software.amazon.awssdk.services.rekognition.model.Image;
import
 software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DetectCustomLabelsResponse;
import software.amazon.awssdk.services.rekognition.model.CustomLabel;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import software.amazon.awssdk.services.rekognition.model.BoundingBox;

import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
```

```
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.NoSuchBucketException;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.List;

import java.awt.*;
import java.awt.font.FontRenderContext;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import javax.swing.*;

import java.util.logging.Level;
import java.util.logging.Logger;

// Calls DetectCustomLabels on an image. Displays bounding boxes or
// image level labels found in the image.
public class ShowCustomLabels extends JPanel {

    private transient BufferedImage image;
    private transient DetectCustomLabelsResponse response;
    private transient Dimension dimension;
    public static final Logger logger =
 Logger.getLogger(ShowCustomLabels.class.getName());

    // Finds custom labels in an image stored in an S3 bucket.
    public ShowCustomLabels(RekognitionClient rekClient,
            S3Client s3client,
            String projectVersionArn,
            String bucket,
            String key,
            Float minConfidence) throws RekognitionException,
 NoSuchBucketException, NoSuchKeyException, IOException {

        logger.log(Level.INFO, "Processing S3 bucket: {0} image {1}", new
 Object[] { bucket, key });
        // Get image from S3 bucket and create BufferedImage
```

```
        GetObjectRequest requestObject =
GetObjectRequest.builder().bucket(bucket).key(key).build();
        ResponseBytes<GetObjectResponse> result =
s3client.getObject(requestObject, ResponseTransformer.toBytes());
        ByteArrayInputStream bis = new
ByteArrayInputStream(result.asByteArray());
        image = ImageIO.read(bis);

        // Set image size
        setWindowDimensions();

        // Construct request parameter for DetectCustomLabels
        S3Object s3Object = S3Object.builder().bucket(bucket).name(key).build();

        Image s3Image = Image.builder().s3Object(s3Object).build();

        DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(s3Image)

.projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

        response = rekClient.detectCustomLabels(request);
        logFoundLabels(response.customLabels());
        drawLabels();

    }

    // Finds custom label in a local image file.
    public ShowCustomLabels(RekognitionClient rekClient,
            String projectVersionArn,
            String photo,
            Float minConfidence)
            throws IOException, RekognitionException {

        logger.log(Level.INFO, "Processing local file: {0}", photo);
        // Get image bytes and buffered image
        InputStream sourceStream = new FileInputStream(new File(photo));
        SdkBytes imageBytes = SdkBytes.fromInputStream(sourceStream);
        ByteArrayInputStream inputStream = new
ByteArrayInputStream(imageBytes.asByteArray());
        image = ImageIO.read(inputStream);

        setWindowDimensions();
```

```
        // Construct request parameter for DetectCustomLabels
        Image localImageBytes = Image.builder().bytes(imageBytes).build();

        DetectCustomLabelsRequest request =
DetectCustomLabelsRequest.builder().image(localImageBytes)

.projectVersionArn(projectVersionArn).minConfidence(minConfidence).build();

        response = rekClient.detectCustomLabels(request);

        logFoundLabels(response.customLabels());
        drawLabels();

    }

    // Sets window dimensions to 1/2 screen size, unless image is smaller
    public void setWindowDimensions() {
        dimension = java.awt.Toolkit.getDefaultToolkit().getScreenSize();

        dimension.width = (int) dimension.getWidth() / 2;
        if (image.getWidth() < dimension.width) {
            dimension.width = image.getWidth();
        }
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getHeight() < dimension.height) {
            dimension.height = image.getHeight();
        }

        setPreferredSize(dimension);

    }

    // Draws bounding boxes (if present) and label text.
    public void drawLabels() {

        int boundingBoxBorderWidth = 5;
        int imageHeight = image.getHeight(this);
        int imageWidth = image.getWidth(this);

        // Set up drawing
        Graphics2D g2d = image.createGraphics();
        g2d.setColor(Color.GREEN);
        g2d.setFont(new Font("Tahoma", Font.PLAIN, 50));
```

```
        Font font = g2d.getFont();
        FontRenderContext frc = g2d.getFontRenderContext();
        g2d.setStroke(new BasicStroke(boundingBoxBorderWidth));

        List<CustomLabel> customLabels = response.customLabels();

        int imageLevelLabelHeight = 0;
        for (CustomLabel customLabel : customLabels) {

            String label = customLabel.name();

            int textWidth = (int) (font.getStringBounds(label, frc).getWidth());
            int textHeight = (int) (font.getStringBounds(label,
frc).getHeight());

            // Draw bounding box, if present
            if (customLabel.geometry() != null) {

                BoundingBox box = customLabel.geometry().boundingBox();
                float left = imageWidth * box.left();
                float top = imageHeight * box.top();

                // Draw black rectangle
                g2d.setColor(Color.BLACK);
                g2d.fillRect(Math.round(left + (boundingBoxBorderWidth)),
Math.round(top + (boundingBoxBorderWidth)),
                        textWidth + boundingBoxBorderWidth, textHeight +
boundingBoxBorderWidth);

                // Write label onto black rectangle
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, left + boundingBoxBorderWidth, (top +
textHeight));

                // Draw bounding box around label location
                g2d.drawRect(Math.round(left), Math.round(top),
Math.round((imageWidth * box.width())),
                        Math.round((imageHeight * box.height())));
            }
            // Draw image level labels.
            else {
                // Draw black rectangle
                g2d.setColor(Color.BLACK);
```

```
                g2d.fillRect(10, 10 + imageLevelLabelHeight, textWidth,
textHeight);
                g2d.setColor(Color.GREEN);
                g2d.drawString(label, 10, textHeight + imageLevelLabelHeight);

                imageLevelLabelHeight += textHeight;
            }

        }
        g2d.dispose();

    }

    // Log the labels found by DetectCustomLabels
    private void logFoundLabels(List<CustomLabel> customLabels) {
        logger.info("Custom labels found:");
        if (customLabels.isEmpty()) {
            logger.log(Level.INFO, "No Custom Labels found. Consider lowering
min confidence.");

        }
        else {
        for (CustomLabel customLabel : customLabels) {
            logger.log(Level.INFO, " Label: {0} Confidence: {1}",
                    new Object[] { customLabel.name(),
customLabel.confidence() } );
            }
        }
    }

    // Draws the image containing the bounding boxes and labels.
    @Override
    public void paintComponent(Graphics g) {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

    public static void main(String args[]) throws Exception {

        String photo = null;
```

```
        String bucket = null;
        String projectVersionArn = null;

        final String USAGE = "\n" + "Usage: " + "<model_arn> <image> <bucket>\n
\n" + "Where:\n"
                + "   model_arn - The ARN of the model that you want to use. \n
\n"
                + "   image - The location of the image on your local file
 system or within an S3 bucket.\n\n"
                + "   bucket - The S3 bucket that contains the image. Don't
 specify if image is local.\n\n";

        // Collect the arguments. If 3 arguments are present, the image is
 assumed to be
        // in an S3 bucket.

        if (args.length < 2 || args.length > 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectVersionArn = args[0];
        photo = args[1];

        if (args.length == 3) {
            bucket = args[2];
        }

        float minConfidence = 50;

        ShowCustomLabels panel = null;

        try {
            // Get the Rekognition client


            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            S3Client s3Client = S3Client.builder()
```

```
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Create frame and panel.
            JFrame frame = new JFrame("Custom Labels");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            if (args.length == 2) {
                // Analyze local image
                panel = new ShowCustomLabels(rekClient, projectVersionArn,
 photo, minConfidence);
            } else {
                // Analyze image in S3 bucket
                panel = new ShowCustomLabels(rekClient, s3Client,
 projectVersionArn, bucket, photo, minConfidence);
            }

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);

        } catch (RekognitionException rekError) {

            String errorMessage = "Rekognition client error: " +
 rekError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (FileNotFoundException fileError) {
            String errorMessage = "File not found: " + photo;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (IOException fileError) {
            String errorMessage = "Input output exception: " +
 fileError.getMessage();
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (NoSuchKeyException bucketError) {
```

```
            String errorMessage = String.format("Image not found: %s in bucket
 %s.", photo, bucket);
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        } catch (NoSuchBucketException bucketError) {
            String errorMessage = "Bucket not found: " + bucket;
            logger.log(Level.SEVERE, errorMessage);
            System.out.println(errorMessage);
            System.exit(1);
        }

    }
}
```

DetectCustomLabels Operationsanfrage

In der `DetectCustomLabels`-Operation stellen Sie ein Eingabebild bereit, entweder als base64-codiertes Byte-Array oder als Bild, das in einem Amazon-S3-Bucket gespeichert ist. Das folgende Beispiel einer JSON-Anforderung zeigt das aus einem Amazon-S3-Bucket geladene Bild.

```
{
    "ProjectVersionArn": "string",
     "Image":{
        "S3Object":{
            "Bucket":"string",
            "Name":"string",
            "Version":"string"
         }
    },
    "MinConfidence": 90,
    "MaxLabels": 10,
}
```

DetectCustomLabels Antwort auf die Operation

Die folgende JSON-Antwort der `DetectCustomLabels`-Operation zeigt die benutzerdefinierten Labels, die im nachfolgenden Bild erkannt wurden.

```
{
```

```
    "CustomLabels": [
        {
            "Name": "MyLogo",
            "Confidence": 77.7729721069336,
            "Geometry": {
                "BoundingBox": {
                    "Width": 0.198987677693367,
                    "Height": 0.31296101212501526,
                    "Left": 0.07924537360668182,
                    "Top": 0.4037395715713501
                }
            }
        }
    ]
}
```

Amazon Rekognition Custom Labels

Dieser Abschnitt gibt Ihnen einen Überblick über den Arbeitsablauf, den Sie verwenden, um ein Amazon Rekognition Custom Labels-Modell zu trainieren und zu verwenden. Ebenfalls enthalten sind Übersichtsinformationen zur Verwendung desAWS SDK zum Trainieren und Verwenden eines Modells.

Verwalten eines Amazon Rekognition Custom Labels-Projekts

In Amazon Rekognition Custom Labels verwenden Sie ein Projekt, um die Modelle zu verwalten, die Sie für einen bestimmten Anwendungsfall erstellen. Ein Projekt verwaltet Datensätze, Modelltraining, Modellversionen, Modellbewertung und das Ausführen der Modelle Ihres Projekts.

Themen

- Löschen eines Amazon Rekognition Custom Labels-Projekts
- Beschreibung eines Projekts (SDK)
- Ein Projekt erstellen mit AWS CloudFormation

Löschen eines Amazon Rekognition Custom Labels-Projekts

Sie können ein Projekt löschen, indem Sie die Amazon Rekognition Rekognition-Konsole verwenden oder die DeleteProjectAPI aufrufen. Um ein Projekt zu löschen, müssen Sie zunächst alle zugehörigen Modelle löschen. Ein gelöschtes Projekt oder Modell kann nicht wiederhergestellt werden.

Themen

- Löschen eines Amazon Rekognition Custom Labels-Projekts (Konsole)
- Löschen eines Amazon Rekognition Custom Labels-Projekts (SDK)

Löschen eines Amazon Rekognition Custom Labels-Projekts (Konsole)

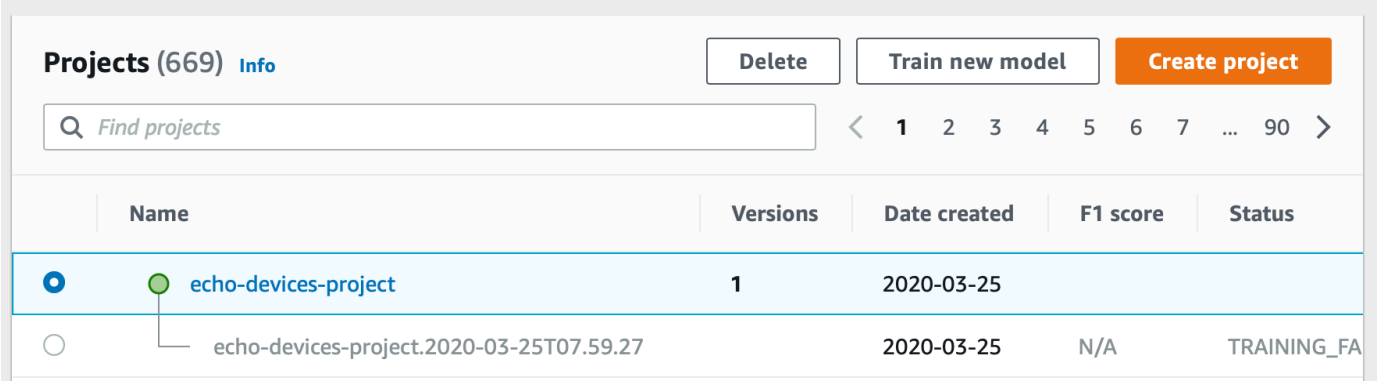
Sie können ein Projekt von der Projektseite oder von der Detailseite eines Projekts löschen. Das folgende Verfahren zeigt Ihnen, wie Sie Projekt löschen, indem Sie die Projektseite verwenden.

Die Amazon Rekognition Custom Labels-Konsole löscht beim Löschen von Projekten die zugehörigen Modelle und Datensätze für Sie. Sie können ein Projekt nicht löschen, wenn eines seiner

Modelle läuft oder trainiert wird. Informationen zum Stoppen eines laufenden Modells finden Sie unter Stoppen eines Amazon Rekognition Custom Labels-Modells (SDK). Wenn ein Modell trainiert wird, warten Sie, bis es fertig ist, bevor Sie das Projekt löschen.

Löschen eines Projekts (Konsole)

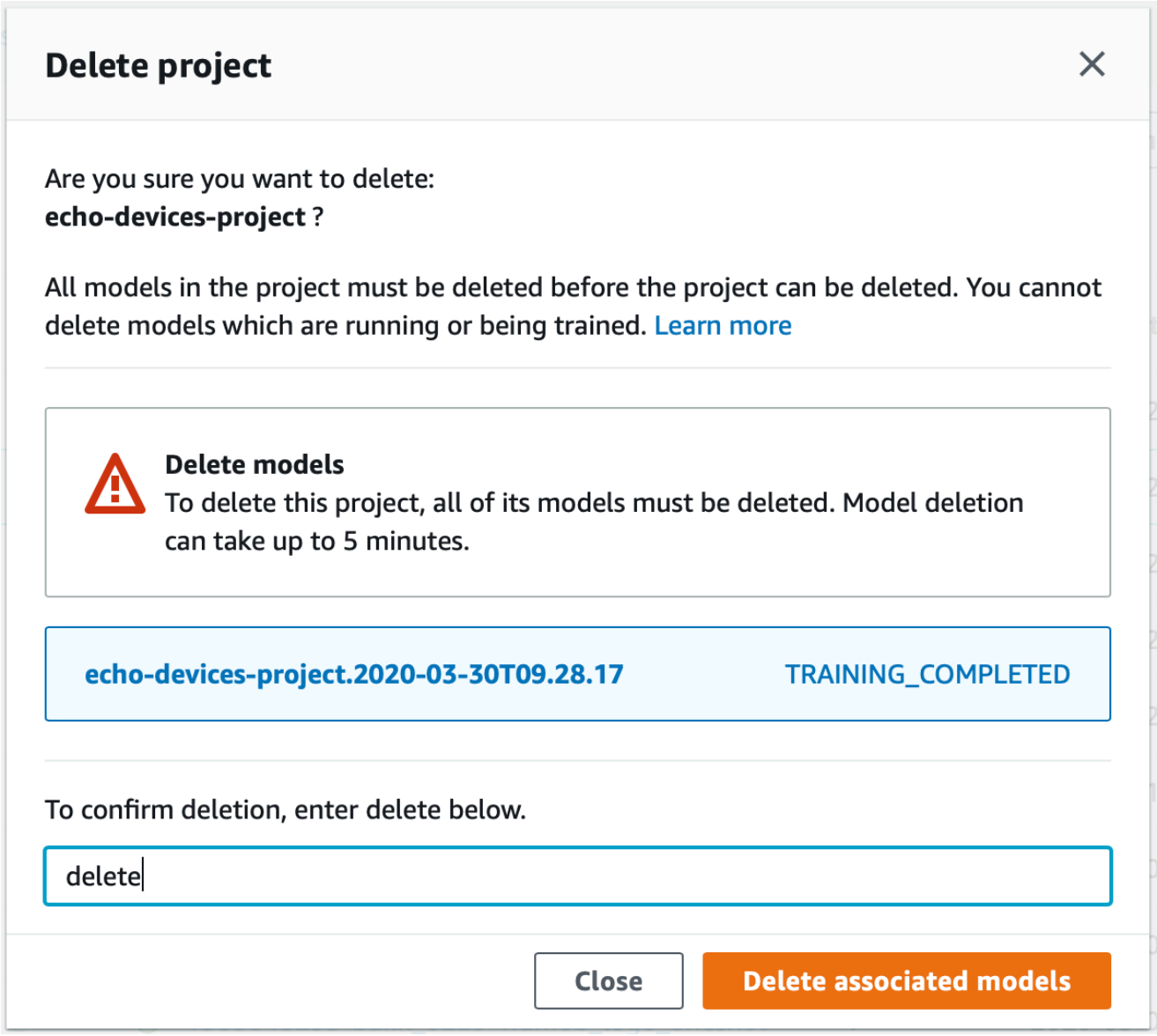
1. Öffnen Sie die Amazon Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Benutzerdefinierte Labels verwenden.
3. Wählen Sie Erste Schritte.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Klicken Sie auf der Seite Projekte auf die Schaltfläche neben dem Projekt, das Sie löschen möchten. Die Projektliste wird angezeigt echo-devices-project, mit einer Version, die am 25.03.2020 erstellt wurde, und den Optionen Löschen, Neues Modell trainieren oder Projekt erstellen.



6. Wählen Sie oben auf der Seite Löschen. Das Dialogfeld Projekt löschen wird angezeigt.
7. Wenn dem Projekt keine Modelle zugeordnet sind:
   a. Geben Sie Löschen ein, um das Projekt zu löschen.
   b. Wählen Sie Löschen, um das Projekt zu löschen.
8. Wenn dem Projekt Modelle oder Datensätze zugeordnet sind:
   a. Geben Sie Löschen ein, um zu bestätigen, dass Sie die Modelle und Datensätze löschen möchten.
   b. Wählen Sie entweder Zugeordnete Modelle löschen oder Zugeordnete Datensätze löschen oder Zugeordnete Datensätze und Modelle löschen, je nachdem, ob das Modell Datensätze, Modelle oder beides enthält. Das Löschen des Modells kann eine Weile dauern.

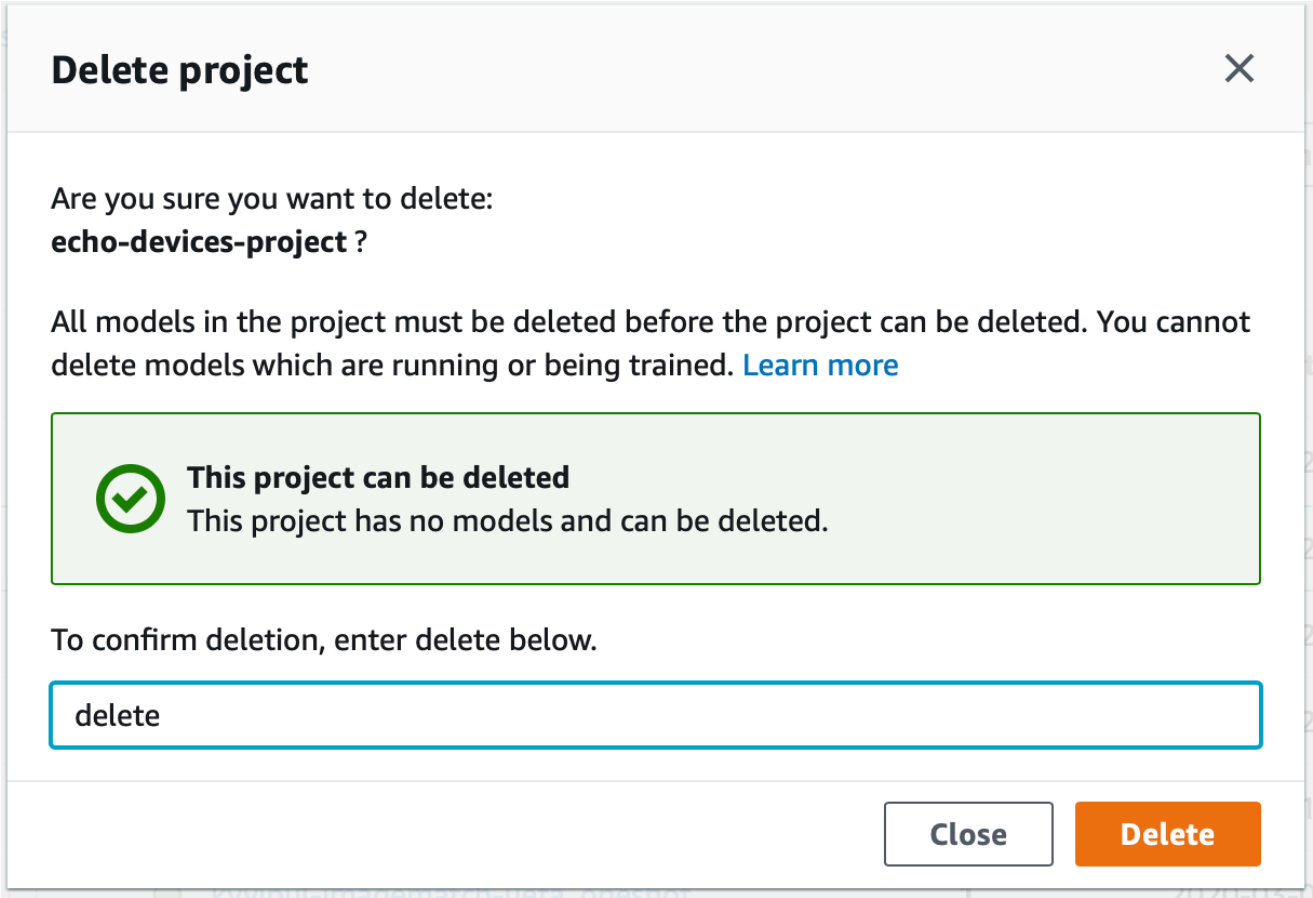
> **Note**
>
> Die Konsole kann keine Modelle löschen, die gerade trainiert werden oder ausgeführt werden. Versuchen Sie es erneut, nachdem Sie alle laufenden Modelle, die in der Liste aufgeführt sind, gestoppt haben, und warten Sie, bis die Modelle, die als Training aufgeführt sind, gestoppt wurden.
> Wenn Sie das Dialogfeld während des Löschens des Modells schließen, werden die Modelle trotzdem gelöscht. Später können Sie das Projekt löschen, indem Sie diesen Vorgang wiederholen.

Das Fenster zum Löschen eines Modells enthält explizite Anweisungen zum Löschen der zugehörigen Modelle.



c. Geben Sie Löschen ein, um zu bestätigen, dass Sie das Projekt löschen möchten.

d. Wählen Sie Löschen, um das Projekt zu löschen.



Löschen eines Amazon Rekognition Custom Labels-Projekts (SDK)

Sie löschen ein Amazon Rekognition Custom Labels-Projekt, indem Sie den Amazon-Ressourcennamen (ARN) des Projekts, das Sie löschen möchten, aufrufen DeleteProjectund angeben. Rufen Sie an, um die ARNs der Projekte in Ihrem AWS Konto zu erhalten. DescribeProjects Die Antwort umfasst eine Reihe von ProjectDescriptionObjekten. Das Projekt ARN ist das `ProjectArn` Feld. Sie können den Projektnamen verwenden, um den ARN des Projekts zu identifizieren. z. B. `arn:aws:rekognition:us-east-1:123456789010:project/`*`project name`*`/1234567890123`.

Bevor Sie ein Projekt löschen können, müssen Sie zunächst alle Modelle und Datensätze im Projekt löschen. Weitere Informationen finden Sie unter Löschen eines Amazon-Rekognition-Custom-Labels-Modells (SDK) und Löschen eines Dataset.

Es kann einige Momente dauern, das Projekt zu löschen. Während dieser Zeit lautet der Status des Projekts `DELETING`. Das Projekt wird gelöscht, wenn ein nachfolgender Aufruf das von Ihnen gelöschte Projekt DescribeProjectsnicht enthält.

So löschen Sie ein Projekt (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Code, um ein Projekt zu löschen.

   AWS CLI

   Ändern Sie den Wert von `project-arn` in den Namen des Projekts, das Sie löschen möchten.

   ```
   aws rekognition delete-project --project-arn project_arn \
     --profile custom-labels-access
   ```

   Python

   Verwenden Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

   - `project_arn`— der ARN des Projekts, das Sie löschen möchten.

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   Purpose
   Amazon Rekognition Custom Labels project example used in the service
    documentation:
   https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/mp-delete-
   project.html
   Shows how to delete an existing Amazon Rekognition Custom Labels project.
   You must first delete any models and datasets that belong to the project.
   """

   import argparse
   import logging
   import time
   import boto3
   ```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def find_forward_slash(input_string, n):
    """
    Returns the location of '/' after n number of occurences.
    :param input_string: The string you want to search
    : n: the occurence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position


def delete_project(rek_client, project_arn):
    """
    Deletes an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project that you want to delete.
    """

    try:
        # Delete the project
        logger.info("Deleting project: %s", project_arn)

        response = rek_client.delete_project(ProjectArn=project_arn)

        logger.info("project status: %s",response['Status'])

        deleted = False

        logger.info("waiting for project deletion: %s", project_arn)

        # Get the project name
        start = find_forward_slash(project_arn, 1) + 1
        end = find_forward_slash(project_arn, 2)
        project_name = project_arn[start:end]

        project_names = [project_name]
```

```
        while deleted is False:

            project_descriptions = rek_client.describe_projects(
                ProjectNames=project_names)['ProjectDescriptions']

            if len(project_descriptions) == 0:
                deleted = True

            else:
                time.sleep(5)

        logger.info("project deleted: %s",project_arn)

        return True

    except ClientError as err:
        logger.exception(
            "Couldn't delete project - %s: %s",
            project_arn, err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that you want to delete."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()
```

```
        print(f"Deleting project: {args.project_arn}")

        # Delete the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        delete_project(rekognition_client,
                       args.project_arn)

        print(f"Finished deleting project: {args.project_arn}")

    except ClientError as err:
        error_message = f"Problem deleting project: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

Verwenden Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- `project_arn`— der ARN des Projekts, das Sie löschen möchten.

```
/*
Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.List;
import java.util.Objects;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
```

```
import software.amazon.awssdk.services.rekognition.model.DeleteProjectRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteProjectResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteProject {

    public static final Logger logger =
 Logger.getLogger(DeleteProject.class.getName());

    public static void deleteMyProject(RekognitionClient rekClient, String
 projectArn) throws InterruptedException {

        try {

            logger.log(Level.INFO, "Deleting project: {0}", projectArn);

            // Delete the project

            DeleteProjectRequest deleteProjectRequest =
 DeleteProjectRequest.builder().projectArn(projectArn).build();
            DeleteProjectResponse response =
 rekClient.deleteProject(deleteProjectRequest);

            logger.log(Level.INFO, "Status: {0}", response.status());

            // Wait until deletion finishes

            Boolean deleted = false;

            do {

        DescribeProjectsRequest describeProjectsRequest =
 DescribeProjectsRequest.builder().build();
                    DescribeProjectsResponse describeResponse =
 rekClient.describeProjects(describeProjectsRequest);
                    List<ProjectDescription> projectDescriptions =
 describeResponse.projectDescriptions();

                    deleted = true;
```

```
                    for (ProjectDescription projectDescription :
 projectDescriptions) {

                        if (Objects.equals(projectDescription.projectArn(),
 projectArn)) {
                            deleted = false;
                            logger.log(Level.INFO, "Not deleted: {0}",
 projectDescription.projectArn());
                            Thread.sleep(5000);
                            break;
                        }
                    }

            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Project deleted: {0} ", projectArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        final String USAGE = "\n" + "Usage: " + "<project_arn>\n\n" + "Where:\n"
             + "   project_arn - The ARN of the project that you want to delete.
\n\n";

        if (args.length != 1) {
             System.out.println(USAGE);
          System.exit(1);
        }

        String projectArn = args[0];

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
```

```
                .region(Region.US_WEST_2)
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .build();

            // Delete the project.
            deleteMyProject(rekClient, projectArn);

            System.out.println(String.format("Project deleted: %s",
 projectArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }
    }

}
```

Beschreibung eines Projekts (SDK)

Sie können die `DescribeProjects` API verwenden, um Informationen über Ihre Projekte abzurufen.

So beschreiben Sie ein Projekt (SDK)

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie die AWS CLI und die AWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Beispielcode, um ein Projekt zu beschreiben. Ersetzen Sie `project_name` durch den Namen des Projekts, das Sie beschreiben möchten. Wenn Sie `--project-names` nicht angeben, werden Beschreibungen für alle Projekte zurückgegeben.

AWS CLI

```
aws rekognition describe-projects --project-names project_name \
  --profile custom-labels-access
```

Python

Verwenden Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- project_name — der Name des Projekts, das Sie beschreiben möchten. Wenn Sie keinen Namen angeben, werden Beschreibungen für alle Projekte zurückgegeben.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to describe an Amazon Rekognition Custom Labels project.
"""
import argparse
import logging
import json
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def display_project_info(project):
    """
    Displays information about a Custom Labels project.
    :param project: The project that you want to display information about.
    """
    print(f"Arn: {project['ProjectArn']}")
    print(f"Status: {project['Status']}")

    if len(project['Datasets']) == 0:
        print("Datasets: None")
    else:
        print("Datasets:")
```

```
    for dataset in project['Datasets']:
        print(f"\tCreated: {str(dataset['CreationTimestamp'])}")
        print(f"\tType: {dataset['DatasetType']}")
        print(f"\tARN: {dataset['DatasetArn']}")
        print(f"\tStatus: {dataset['Status']}")
        print(f"\tStatus message: {dataset['StatusMessage']}")
        print(f"\tStatus code: {dataset['StatusMessageCode']}")
        print()
    print()


def describe_projects(rek_client, project_name):
    """
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_name: The project you want to describe. Pass None to describe
 all projects.
    """

    try:
        # Describe the project
        if project_name is None:
            logger.info("Describing all projects.")
        else:
            logger.info("Describing project: %s.",project_name)

        if project_name is None:
            response = rek_client.describe_projects()
        else:
            project_names = json.loads('["' + project_name + '"]')
            response = rek_client.describe_projects(ProjectNames=project_names)

        print('Projects\n--------')
        if len(response['ProjectDescriptions']) == 0:
            print("Project(s) not found.")
        else:
            for project in response['ProjectDescriptions']:
                display_project_info(project)

        logger.info("Finished project description.")

    except ClientError as err:
        logger.exception(
            "Couldn't describe project - %s: %s",
```

```
            project_name,err.response['Error']['Message'] )
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "--project_name",  help="The name of the project that you want to
 describe.", required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)

        args = parser.parse_args()

        print(f"Describing projects: {args.project_name}")

        # Describe the project.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_projects(rekognition_client,
                          args.project_name)

        if args.project_name is None:
            print("Finished describing all projects.")
        else:
            print("Finished describing project %s.", args.project_name)

    except ClientError as err:
```

```
        error_message = f"Problem describing project: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

Verwenden Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- `project_name` — die ARN des Projekts, das Sie beschreiben möchten. Wenn Sie keinen Namen angeben, werden Beschreibungen für alle Projekte zurückgegeben.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetMetadata;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectsResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DescribeProjects {

    public static final Logger logger =
 Logger.getLogger(DescribeProjects.class.getName());
```

```
   public static void describeMyProjects(RekognitionClient rekClient, String
projectName) {

       DescribeProjectsRequest descProjects = null;

       // If a single project name is supplied, build projectNames argument

       List<String> projectNames = new ArrayList<String>();

       if (projectName == null) {
           descProjects = DescribeProjectsRequest.builder().build();
       } else {
           projectNames.add(projectName);
           descProjects =
DescribeProjectsRequest.builder().projectNames(projectNames).build();
       }

       // Display useful information for each project.

       DescribeProjectsResponse resp =
rekClient.describeProjects(descProjects);

       for (ProjectDescription projectDescription : resp.projectDescriptions())
{

           System.out.println("ARN: " + projectDescription.projectArn());
           System.out.println("Status: " +
projectDescription.statusAsString());
           if (projectDescription.hasDatasets()) {
               for (DatasetMetadata datasetDescription :
projectDescription.datasets()) {
                   System.out.println("\tdataset Type: " +
datasetDescription.datasetTypeAsString());
                   System.out.println("\tdataset ARN: " +
datasetDescription.datasetArn());
                   System.out.println("\tdataset Status: " +
datasetDescription.statusAsString());
               }
           }
           System.out.println();
       }

   }
```

```
    public static void main(String[] args) {

        String projectArn = null;

        // Get command line arguments

        final String USAGE = "\n" + "Usage: " + "<project_name>\n\n" + "Where:
\n"
                + "   project_name - (Optional) The name of the project that you
 want to describe. If not specified, all projects "
                + "are described.\n\n";

        if (args.length > 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        if (args.length == 1) {
            projectArn = args[0];
        }

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Describe projects

            describeMyProjects(rekClient, projectArn);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }
```

```
}
```

Ein Projekt erstellen mit AWS CloudFormation

Amazon Rekognition Custom Labels ist integriert AWS CloudFormation, ein Service, der Sie bei der Modellierung und Einrichtung Ihrer AWS Ressourcen unterstützt, sodass Sie weniger Zeit mit der Erstellung und Verwaltung Ihrer Ressourcen und Infrastruktur verbringen müssen. Sie erstellen eine Vorlage, die alle gewünschten AWS Ressourcen beschreibt und sich um die Bereitstellung und Konfiguration dieser Ressourcen für Sie AWS CloudFormation kümmert.

Sie können AWS CloudFormation es verwenden, um Amazon Rekognition Custom Labels-Projekte bereitzustellen und zu konfigurieren.

Wenn Sie es verwenden AWS CloudFormation, können Sie Ihre Vorlage wiederverwenden, um Ihre Amazon Rekognition Custom Labels-Projekte konsistent und wiederholt einzurichten. Beschreiben Sie Ihre Projekte einfach einmal und stellen Sie dann dieselben Projekte immer wieder in mehreren AWS Konten und Regionen bereit.

Amazon Rekognition Benutzerdefinierte Labels und Vorlagen AWS CloudFormation

Um Ressourcen für Amazon Rekognition Custom Labels und verwandte Dienstleistungen bereitzustellen und zu konfigurieren, müssen Sie AWS CloudFormation -Vorlagen kennen und verstehen. Vorlagen sind formatierte Textdateien in JSON oder YAML. Diese Vorlagen beschreiben die Ressourcen, die Sie in Ihren AWS CloudFormation Stacks bereitstellen möchten. Wenn Sie mit JSON oder YAML nicht vertraut sind, können Sie AWS CloudFormation Designer verwenden, um Ihnen die ersten Schritte mit Vorlagen zu erleichtern. AWS CloudFormation Weitere Informationen finden Sie unter Was ist AWS CloudFormation -Designer? im AWS CloudFormation - Benutzerhandbuch.

Referenzinformationen zu Amazon Rekognition Custom Labels-Projekten, einschließlich Beispiele für JSON- und YAML-Vorlagen, finden Sie in der Referenz zum Rekognition-Ressourcentyp.

Erfahren Sie mehr über AWS CloudFormation

Weitere Informationen AWS CloudFormation dazu finden Sie in den folgenden Ressourcen:

- AWS CloudFormation
- AWS CloudFormation Benutzerhandbuch

- AWS CloudFormation API Reference
- AWS CloudFormation Benutzerhandbuch für die Befehlszeilenschnittstelle

Verwalten von Datensätzen

Ein Datensatz enthält die Bilder und zugewiesenen Bezeichnungen, die Sie zum Trainieren oder Testen eines Modells verwenden. Die Themen in diesem Abschnitt zeigen, wie Sie einen Datensatz mit der Amazon Rekognition Custom Labels-Konsole und demAWS SDK verwalten können.

Themen

- Hinzufügen eines Datensatzes zu einem Projekt
- Hinzufügen weiterer Bilder zu einem Datensatz
- Erstellen eines Datensatzes unter Verwendung eines vorhandenen Datensatzes (SDK)
- Beschreibung eines Datensatzes (SDK)
- Datensatzeinträge auflisten (SDK)
- Verteilen eines Trainingsdatensatzes (SDK)
- Löschen eines Dataset

Hinzufügen eines Datensatzes zu einem Projekt

Sie können einem vorhandenen Projekt einen Trainingsdatensatz oder einen Testdatensatz hinzufügen. Wenn Sie einen vorhandenen Datensatz ersetzen möchten, löschen Sie zuerst den vorhandenen Datensatz. Weitere Informationen finden Sie unter Löschen eines Dataset. Fügen Sie dann den neuen Datensatz hinzu.

Themen

- Hinzufügen eines Datensatzes zu einem Projekt (Konsole)
- Hinzufügen eines Datensatzes zu einem Projekt (SDK)

Hinzufügen eines Datensatzes zu einem Projekt (Konsole)

Sie können einem Projekt einen Trainings- oder Testdatensatz hinzufügen, indem Sie die Amazon Rekognition Custom Labels-Konsole verwenden.

Einen Datensatz zu einem Projekt ein Dataset hinzufügen

1. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie im linken Bereich die Option „Benutzerdefinierte Bezeichnungen“ aus. Die Amazon Rekognition Custom Labels Landingpage wird angezeigt.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus. Die Projektansicht wird angezeigt.
4. Wählen Sie das Projekt aus, dem Sie ein Dataset hinzufügen möchten.
5. Wählen Sie im linken Navigationsbereich unter dem Projektnamen die Option Dataset aus.
6. Wenn das Projekt keinen vorhandenen Datensatz hat, wird die Seite Datensatz erstellen angezeigt. Gehen Sie wie folgt vor:

   a. Geben Sie auf der Seite Datensatz erstellen die Bildquelleninformationen ein. Weitere Informationen finden Sie unter the section called “Erstellen von Datensätzen mit Bildern”.

   b. Wählen Sie Datensatz erstellen, um den Datensatz zu erstellen.
7. Wenn das Projekt über einen vorhandenen Datensatz (Training oder Test) verfügt, wird die Seite mit den Projektdetails angezeigt. Gehen Sie wie folgt vor:

   a. Wählen Sie auf der Seite mit den Projektdetails die Option Aktionen aus.

   b. Wenn Sie einen Trainingsdatensatz hinzufügen möchten, wählen Sie Trainingsdatensatz erstellen.

   c. Wenn Sie einen Testdatensatz hinzufügen möchten, wählen Sie Testdatensatz erstellen.

   d. Geben Sie auf der Seite Datensatz erstellen die Bildquelleninformationen ein. Weitere Informationen finden Sie unter the section called “Erstellen von Datensätzen mit Bildern”.

   e. Wählen Sie Datensatz erstellen, um den Datensatz zu erstellen.
8. Fügen Sie Ihrem Datensatz Bilder hinzu. Weitere Informationen finden Sie unter Weitere Bilder hinzufügen (Konsole).
9. Fügen Sie Ihrem Datensatz Labels hinzu. Weitere Informationen finden Sie unter Neue Labels hinzufügen (Konsole).
10. Füge Labels zu deinen Bildern hinzu. Informationen zum Hinzufügen von Bezeichnungen auf Bildebene finden Sie unterthe section called “Einem Bild Labels auf Bildebene zuweisen”. Informationen zum Hinzufügen von Begrenzungsrahmen finden Sie unterObjekte mit Begrenzungsrahmen mit Labels versehen. Weitere Informationen finden Sie unter Datensätzen einen Zweck geben.

Hinzufügen eines Datensatzes zu einem Projekt (SDK)

Sie können ein Zug- oder TestDataset zu einem bestehenden Projekt auf folgende Weise hinzufügen:

- Erstellen Sie ein Dataset unter Verwendung einer Manifestdatei. Weitere Informationen finden Sie unter Einen Datensatz mit einer SageMaker Ground Truth Manifest-Datei (SDK) erstellen.
- Erstellen Sie einen leeren Datensatz und füllen Sie den Datensatz anschließend auf. Im folgenden Beispiel wird gezeigt, wie ein leeres Dataset erstellt wird. Informationen zum Hinzufügen von Einträgen, nachdem Sie einen leeren Datensatz erstellt haben, finden Sie unterHinzufügen weiterer Bilder zu einem Datensatz.

So fügen Sie einem Projekt einen Datensatz hinzu (SDK)

1. Falls noch nicht erfolgt, installieren und konfigurieren Sie dieAWS CLI und dieAWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie die folgenden Beispiele, um JSON-Zeilen zu einem Datensatz hinzuzufügen.

   CLI

   `project_arn`Ersetzen Sie das Dataset, dem Sie den Datensatz hinzufügen möchten. Ersetzen Sie`dataset_type` durch,`TRAIN` um einen Trainingsdatensatz oder einen Testdatensatz`TEST` zu erstellen.

   ```
   aws rekognition create-dataset --project-arn project_arn \
     --dataset-type dataset_type \
     --profile custom-labels-access
   ```

   Python

   Verwenden Sie folgenden Code, um ein Dataset zu erstellen. Geben Sie die folgenden Befehlszeilenoptionen an:

   - `project_arn`— den ARN des Projekts, dem Sie den Testdataset hinzufügen möchten.
   - `type`— die Art des Datensatzes, den Sie erstellen möchten (trainieren oder testen)

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0
   ```

```
import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

def create_empty_dataset(rek_client, project_arn, dataset_type):
    """
    Creates an empty Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
 dataset.
    :param dataset_type: The type of the dataset that you want to create (train
 or test).
    """

    try:
        #Create the dataset.
        logger.info("Creating empty %s dataset for project %s",
            dataset_type, project_arn)

        dataset_type=dataset_type.upper()

        response = rek_client.create_dataset(
            ProjectArn=project_arn, DatasetType=dataset_type
        )

        dataset_arn=response['DatasetArn']

        logger.info("dataset ARN: %s", dataset_arn)

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']

            if status == "CREATE_IN_PROGRESS":

                logger.info(("Creating dataset: %s ", dataset_arn))
```

```
                time.sleep(5)
                continue

            if status == "CREATE_COMPLETE":
                logger.info("Dataset created: %s", dataset_arn)
                finished=True
                continue

            if status == "CREATE_FAILED":
                error_message = f"Dataset creation failed: {status} :
 {dataset_arn}"
                logger.exception(error_message)
                raise Exception(error_message)

            error_message = f"Failed. Unexpected state for dataset creation:
 {status} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        return dataset_arn

    except ClientError as err:
        logger.exception("Couldn't create dataset: %s", err.response['Error']
['Message'])
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
 the empty dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the empty dataset that you want to
 create (train or test)."
    )


def main():
```

```
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Creating empty {args.dataset_type} dataset for project
 {args.project_arn}")

        # Create the empty dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn=create_empty_dataset(rekognition_client,
            args.project_arn,
            args.dataset_type.lower())

        print(f"Finished creating empty dataset: {dataset_arn}")


    except ClientError as err:
        logger.exception("Problem creating empty dataset: %s", err)
        print(f"Problem creating empty dataset: {err}")
    except Exception as err:
        logger.exception("Problem creating empty dataset: %s", err)
        print(f"Problem creating empty dataset: {err}")


if __name__ == "__main__":
    main()
```

Java V2

Verwenden Sie folgenden Code, um ein Dataset zu erstellen. Geben Sie die folgenden Befehlszeilenoptionen an:

- `project_arn`— den ARN des Projekts, dem Sie den Testdataset hinzufügen möchten.
- `type`— die Art des Datensatzes, den Sie erstellen möchten (trainieren oder testen)

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateEmptyDataset {

    public static final Logger logger =
 Logger.getLogger(CreateEmptyDataset.class.getName());

    public static String createMyEmptyDataset(RekognitionClient rekClient,
 String projectArn, String datasetType)
            throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating empty {0} dataset for project :
 {1}",
                    new Object[] { datasetType.toString(), projectArn });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
```

```
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
datasetType);
                throw new Exception("Unrecognized dataset type: " +
datasetType);

            }

            CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)
                    .datasetType(requestDatasetType).build();

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            //Wait until updates finishes

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(response.datasetArn()).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

                switch (status) {

                case CREATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset created");
```

```
                    created = true;
                    break;

                case CREATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;

                case CREATE_FAILED:
                    String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (created == false);

            return response.datasetArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
            throw e;
        }

    }


    public static void main(String args[]) {

        String datasetType = null;
        String datasetArn = null;
        String projectArn = null;
```

```
        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>\n
\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
 copy the datast to.\n\n"
                + "   dataset_type - the type of the empty dataset that you want
 to create (train or test).\n\n";


        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        projectArn = args[0];
        datasetType = args[1];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Create the dataset
            datasetArn = createMyEmptyDataset(rekClient, projectArn,
 datasetType);

            System.out.println(String.format("Created dataset: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }
```

```
}
```

3. Fügen Sie Bilder zum Dataset ein. Weitere Informationen finden Sie unter Weitere Bilder hinzufügen (SDK).

Hinzufügen weiterer Bilder zu einem Datensatz

Sie können Ihren Datensätzen weitere Bilder hinzufügen, indem Sie die Amazon Rekognition Custom Labels-Konsole verwenden oder die`UpdateDatasetEntries` API aufrufen.

Themen

- Weitere Bilder hinzufügen (Konsole)
- Weitere Bilder hinzufügen (SDK)

Weitere Bilder hinzufügen (Konsole)

Wenn Sie die Amazon Rekognition Custom Labels-Konsole verwenden, laden Sie Bilder von Ihrem lokalen Computer hoch. Die Bilder werden dem Amazon S3 S3-Bucket-Speicherort (Konsole oder extern) hinzugefügt, in dem die zur Erstellung des Datensatzes verwendeten Bilder gespeichert sind.

Um Ihrem Datensatz weitere Bilder hinzuzufügen (Konsole)

1. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie im linken Bereich die Option „Benutzerdefinierte Bezeichnungen“ aus. Die Amazon Rekognition Custom Labels Landingpage wird angezeigt.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus. Die Projektansicht wird angezeigt.
4. Wählen Sie das Projekt aus, das Sie verwenden möchten.
5. Wählen Sie im linken Navigationsbereich unter dem Projektnamen die Option Dataset aus.
6. Wählen Sie die Option Aktionen aus und wählen Sie den Datensatz aus, dem Sie Bilder hinzufügen möchten.
7. Wählen Sie die Bilder aus, die Sie in den Datensatz hochladen möchten. Sie können die Bilder ziehen oder die Bilder auswählen, die Sie von Ihrem lokalen Computer hochladen möchten. Sie können bis zu 30 Bilder gleichzeitig hochladen.

8. Wählen Sie Bilder hochladen.
9. Wählen Sie Save Changes (Änderungen speichern).
10. Beschriften Sie die Bilder. Weitere Informationen finden Sie unter Labeling von Bildern.

Weitere Bilder hinzufügen (SDK)

`UpdateDatasetEntries`aktualisiert JSON-Zeilen oder fügt sie zu einer Manifestdatei hinzu. Sie übergeben die JSON-Zeilen als Byte64-kodiertes Datenobjekt in das`GroundTruth` Feld. Wenn Sie einAWS SDK zum Aufrufen verwenden`UpdateDatasetEntries`, kodiert das SDK die Daten für Sie. Jede JSON-Zeile enthält Informationen für ein einzelnes Bild, z. B. zugewiesene Beschriftungen oder Bounding-Box-Informationen. Beispiel:

```
{"source-ref":"s3://bucket/image","BB":{"annotations":
[{"left":1849,"top":1039,"width":422,"height":283,"class_id":0},
{"left":1849,"top":1340,"width":443,"height":415,"class_id":1},
{"left":2637,"top":1380,"width":676,"height":338,"class_id":2},
{"left":2634,"top":1051,"width":673,"height":338,"class_id":3}],"image_size":
[{"width":4000,"height":2667,"depth":3}]},"BB-metadata":{"job-name":"labeling-job/
BB","class-map":
{"0":"comparator","1":"pot_resistor","2":"ir_phototransistor","3":"ir_led"},"human-
annotated":"yes","objects":[{"confidence":1},{"confidence":1},{"confidence":1},
{"confidence":1}],"creation-date":"2021-06-22T10:11:18.006Z","type":"groundtruth/
object-detection"}}
```

Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

Verwenden Sie`source-ref` das Feld als Schlüssel, um die Bilder zu aktualisieren. Wenn der Datensatz keinen passenden`source-ref` Feldwert enthält, wird die JSON-Zeile als neues Bild hinzugefügt.

Um mehr Bilder zu einem Datensatz hinzuzufügen (SDK)

1. Falls noch nicht erfolgt, installieren und konfigurieren Sie dieAWS CLI und dieAWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie die folgenden Beispiele, um JSON-Zeilen zu einem Datensatz hinzuzufügen.

   CLI

   Ersetzen Sie den Wert von`GroundTruth` durch die JSON-Zeilen, die Sie verwenden möchten. Sie müssen alle Sonderzeichen innerhalb der JSON-Zeile maskieren.

```
aws rekognition update-dataset-entries\
  --dataset-arn dataset_arn \
  --changes '{"GroundTruth" : "{\"source-ref\":\"s3://your_bucket/your_image
\",\"BB\":{\"annotations\":[{\"left\":1776,\"top\":1017,\"width\":458,\"height
\":317,\"class_id\":0},{\"left\":1797,\"top\":1334,\"width\":418,\"height
\":415,\"class_id\":1},{\"left\":2597,\"top\":1361,\"width\":655,\"height
\":329,\"class_id\":2},{\"left\":2581,\"top\":1020,\"width\":689,\"height
\":338,\"class_id\":3}],\"image_size\":[{\"width\":4000,\"height\":2667,
\"depth\":3}]},\"BB-metadata\":{\"job-name\":\"labeling-job/BB\",\"class-map
\":{\"0\":\"comparator\",\"1\":\"pot_resistor\",\"2\":\"ir_phototransistor\",
\"3\":\"ir_led\"},\"human-annotated\":\"yes\",\"objects\":[{\"confidence\":1},
{\"confidence\":1},{\"confidence\":1},{\"confidence\":1}],\"creation-date\":
\"2021-06-22T10:10:48.492Z\",\"type\":\"groundtruth/object-detection\"}}" }' \
  --cli-binary-format raw-in-base64-out \
  --profile custom-labels-access
```

Python

Verwenden Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- dataset_arn — der ARN des Datensatzes, den Sie aktualisieren möchten.
- updates_file — die Datei, die die Aktualisierungen der JSON-Zeile enthält.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to add entries to an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
def update_dataset_entries(rek_client, dataset_arn, updates_file):
    """
    Adds dataset entries to an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that yuo want to update.
    :param updates_file: The manifest file of JSON Lines that contains the
 updates.
    """

    try:
        status=""
        status_message=""

        # Update dataset entries.
        logger.info("Updating dataset %s", dataset_arn)


        with open(updates_file) as f:
            manifest_file = f.read()


        changes=json.loads('{ "GroundTruth" : ' +
            json.dumps(manifest_file) +
            '}')

        rek_client.update_dataset_entries(
            Changes=changes, DatasetArn=dataset_arn
        )

        finished=False
        while finished is False:

            dataset=rek_client.describe_dataset(DatasetArn=dataset_arn)

            status=dataset['DatasetDescription']['Status']
            status_message=dataset['DatasetDescription']['StatusMessage']

            if status == "UPDATE_IN_PROGRESS":

                logger.info("Updating dataset: %s ", dataset_arn)
                time.sleep(5)
                continue

            if status == "UPDATE_COMPLETE":
```

```
                logger.info("Dataset updated: %s : %s : %s",
                    status, status_message, dataset_arn)
                finished=True
                continue

            if status == "UPDATE_FAILED":
                error_message = f"Dataset update failed: {status} :
 {status_message} : {dataset_arn}"
                logger.exception(error_message)
                raise Exception (error_message)

            error_message = f"Failed. Unexpected state for dataset update:
 {status} : {status_message} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        logger.info("Added entries to dataset")

        return status, status_message


    except ClientError as err:
        logger.exception("Couldn't update dataset: %s", err.response['Error']
['Message'])
        raise

def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to update."
    )

    parser.add_argument(
        "updates_file", help="The manifest file of JSON Lines that contains the
 updates."
    )

def main():

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
```

```
    try:

        #get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Updating dataset {args.dataset_arn} with entries from
 {args.updates_file}.")

        # Update the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        status, status_message=update_dataset_entries(rekognition_client,
            args.dataset_arn,
            args.updates_file)

        print(f"Finished updates dataset: {status} : {status_message}")


    except ClientError as err:
        logger.exception("Problem updating dataset: %s", err)
        print(f"Problem updating dataset: {err}")

    except Exception as err:
        logger.exception("Problem updating dataset: %s", err)
        print(f"Problem updating dataset: {err}")


if __name__ == "__main__":
    main()
```

Java V2

- dataset_arn — der ARN des Datensatzes, den Sie aktualisieren möchten.
- update_file — die Datei, die die Aktualisierungen der JSON-Zeile enthält.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetChanges;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.model.UpdateDatasetEntriesRequest;
import
 software.amazon.awssdk.services.rekognition.model.UpdateDatasetEntriesResponse;

import java.io.FileInputStream;
import java.io.InputStream;
import java.util.logging.Level;
import java.util.logging.Logger;

public class UpdateDatasetEntries {

    public static final Logger logger =
 Logger.getLogger(UpdateDatasetEntries.class.getName());

    public static String updateMyDataset(RekognitionClient rekClient, String
 datasetArn,
            String updateFile
            ) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Updating dataset {0}",
                    new Object[] { datasetArn});


            InputStream sourceStream = new FileInputStream(updateFile);
            SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```

```
            DatasetChanges datasetChanges = DatasetChanges.builder()
                    .groundTruth(sourceBytes).build();

            UpdateDatasetEntriesRequest updateDatasetEntriesRequest =
UpdateDatasetEntriesRequest.builder()
                    .changes(datasetChanges)
                    .datasetArn(datasetArn)
                    .build();

            UpdateDatasetEntriesResponse response =
rekClient.updateDatasetEntries(updateDatasetEntriesRequest);

            boolean updated = false;

            //Wait until update completes

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(datasetArn).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, " dataset ARN: {0} ", datasetArn);

                switch (status) {

                case UPDATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset updated");
                    updated = true;
                    break;

                case UPDATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;

                case UPDATE_FAILED:
```

```
                    String error = "Dataset update failed: " +
 datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
 datasetArn;
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected update state: " +
 datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
 datasetArn;
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (updated == false);

            return datasetArn;

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not update dataset: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        String updatesFile = null;
        String datasetArn = null;


        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_arn>
 <updates_file>\n\n" + "Where:\n"
                + "   dataset_arn - the ARN of the dataset that you want to
 update.\n\n"
                + "   update_file - The file that includes in JSON Line updates.
\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
```

```
        }

        datasetArn = args[0];
        updatesFile = args[1];


        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

             // Update the dataset
            datasetArn = updateMyDataset(rekClient, datasetArn, updatesFile);

            System.out.println(String.format("Dataset updated: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Erstellen eines Datensatzes unter Verwendung eines vorhandenen Datensatzes (SDK)

Das folgende Verfahren zeigt, wie Sie einen Datensatz aus einem vorhandenen Datensatzes erstellen CreateDatasetkönnen.

1. Falls noch nicht erfolgt, installieren und konfigurieren Sie dieAWS CLI und dieAWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Beispielcode, um einen Datensatz zu erstellen, indem Sie einen anderen Datensatz kopieren.

   AWS CLI

   Verwenden Sie den folgenden Code, um den Datensatz zu erstellen. Ersetzen Sie Folgendes:

   - `project_arn`— den ARN des Projekts, dem Sie den Datensatz hinzufügen möchten.
   - `dataset_type`— mit dem Typ des Datensatzes (`TRAINoderTEST`), den Sie im Projekt erstellen möchten.
   - `dataset_arn`— mit dem ARN des Datensatzes, den Sie kopieren möchten.

   ```
   aws rekognition create-dataset --project-arn project_arn \
     --dataset-type dataset_type \
     --dataset-source '{ "DatasetArn" : "dataset_arn" }' \
     --profile custom-labels-access
   ```

   Python

   Das folgende Beispiel erstellt einen Datensatz unter Verwendung eines vorhandenen Datensatzes und zeigt seinen ARN an.

   Geben Sie die folgenden Befehlszeilenargumente an, um das Programm auszuführen:

   - `project_arn`— den ARN des Projekts, das Sie verwenden möchten.
   - `dataset_type`— der Typ des Projektdatensatzes, den Sie erstellen möchten (`trainodertest`).
   - `dataset_arn`— den ARN des Datensatzes, aus dem Sie den Datensatz erstellen möchten.

   ```
   # Copyright 2023 Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # PDX-License-Identifier: MIT-0 (For details, see https://github.com/
   awsdocs/amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
   SAMPLECODE.)
   ```

```
import argparse
import logging
import time
import json
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def create_dataset_from_existing_dataset(rek_client, project_arn, dataset_type,
 dataset_arn):
    """
    Creates an Amazon Rekognition Custom Labels dataset using an existing
 dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the project in which you want to create a
 dataset.
    :param dataset_type: The type of the dataset that you want to create (train
 or test).
    :param dataset_arn: The ARN of the existing dataset that you want to use.
    """

    try:
        # Create the dataset

        dataset_type=dataset_type.upper()

        logger.info(
            "Creating %s dataset for project %s from dataset %s.",
                dataset_type,project_arn, dataset_arn)

        dataset_source = json.loads(
            '{ "DatasetArn": "' + dataset_arn + '"}'
        )

        response = rek_client.create_dataset(
            ProjectArn=project_arn, DatasetType=dataset_type,
 DatasetSource=dataset_source
        )

        dataset_arn = response['DatasetArn']
```

```
        logger.info("New dataset ARN: %s", dataset_arn)

        finished = False
        while finished is False:

            dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

            status = dataset['DatasetDescription']['Status']

            if status == "CREATE_IN_PROGRESS":

                logger.info(("Creating dataset: %s ", dataset_arn))
                time.sleep(5)
                continue

            if status == "CREATE_COMPLETE":
                logger.info("Dataset created: %s", dataset_arn)
                finished = True
                continue

            if status == "CREATE_FAILED":
                error_message = f"Dataset creation failed: {status} :
 {dataset_arn}"
                logger.exception(error_message)
                raise Exception(error_message)

            error_message = f"Failed. Unexpected state for dataset creation:
 {status} : {dataset_arn}"
            logger.exception(error_message)
            raise Exception(error_message)

        return dataset_arn

    except ClientError as err:
        logger.exception(
            "Couldn't create dataset: %s",err.response['Error']['Message'] )
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
```

```
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which you want to create
 the dataset."
    )

    parser.add_argument(
        "dataset_type", help="The type of the dataset that you want to create
 (train or test)."
    )

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to copy from."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Creating {args.dataset_type} dataset for project
 {args.project_arn}")

        # Create the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        dataset_arn = create_dataset_from_existing_dataset(rekognition_client,
                                    args.project_arn,
                                    args.dataset_type,
                                    args.dataset_arn)

        print(f"Finished creating dataset: {dataset_arn}")
```

```
    except ClientError as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")
    except Exception as err:
        logger.exception("Problem creating dataset: %s", err)
        print(f"Problem creating dataset: {err}")


if __name__ == "__main__":
    main()
```

Java V2

Das folgende Beispiel erstellt einen Datensatz unter Verwendung eines vorhandenen Datensatzes und zeigt seinen ARN an.

Geben Sie die folgenden Befehlszeilenargumente an, um das Programm auszuführen:

- `project_arn`— den ARN des Projekts, das Sie verwenden möchten.
- `dataset_type`— der Typ des Projektdatensatzes, den Sie erstellen möchten (`train`oder`test`).
- `dataset_arn`— den ARN des Datensatzes, aus dem Sie den Datensatz erstellen möchten.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.CreateDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetSource;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DatasetType;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
```

```
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CreateDatasetExisting {

    public static final Logger logger =
 Logger.getLogger(CreateDatasetExisting.class.getName());

    public static String createMyDataset(RekognitionClient rekClient, String
 projectArn, String datasetType,
            String existingDatasetArn) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Creating {0} dataset for project : {1} from
 dataset {2} ",
                    new Object[] { datasetType.toString(), projectArn,
 existingDatasetArn });

            DatasetType requestDatasetType = null;

            switch (datasetType) {
            case "train":
                requestDatasetType = DatasetType.TRAIN;
                break;
            case "test":
                requestDatasetType = DatasetType.TEST;
                break;
            default:
                logger.log(Level.SEVERE, "Unrecognized dataset type: {0}",
 datasetType);
                throw new Exception("Unrecognized dataset type: " +
 datasetType);

            }

            DatasetSource datasetSource =
 DatasetSource.builder().datasetArn(existingDatasetArn).build();
```

```
            CreateDatasetRequest createDatasetRequest =
CreateDatasetRequest.builder().projectArn(projectArn)

.datasetType(requestDatasetType).datasetSource(datasetSource).build();

            CreateDatasetResponse response =
rekClient.createDataset(createDatasetRequest);

            boolean created = false;

            //Wait until create finishes

            do {

                DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                        .datasetArn(response.datasetArn()).build();
                DescribeDatasetResponse describeDatasetResponse =
rekClient.describeDataset(describeDatasetRequest);

                DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

                DatasetStatus status = datasetDescription.status();

                logger.log(Level.INFO, "Creating dataset ARN: {0} ",
response.datasetArn());

                switch (status) {

                case CREATE_COMPLETE:
                    logger.log(Level.INFO, "Dataset created");
                    created = true;
                    break;

                case CREATE_IN_PROGRESS:
                    Thread.sleep(5000);
                    break;

                case CREATE_FAILED:
                    String error = "Dataset creation failed: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
```

```
                    logger.log(Level.SEVERE, error);
                    throw new Exception(error);

                default:
                    String unexpectedError = "Unexpected creation state: " +
datasetDescription.statusAsString() + " "
                            + datasetDescription.statusMessage() + " " +
response.datasetArn();
                    logger.log(Level.SEVERE, unexpectedError);
                    throw new Exception(unexpectedError);
                }

            } while (created == false);

            return response.datasetArn();

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not create dataset: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String datasetType = null;
        String datasetArn = null;
        String projectArn = null;
        String datasetSourceArn = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_type>
<dataset_arn>\n\n" + "Where:\n"
                + "   project_arn - the ARN of the project that you want to add
copy the datast to.\n\n"
                + "   dataset_type - the type of the dataset that you want to
create (train or test).\n\n"
                + "   dataset_arn - the ARN of the dataset that you want to copy
from.\n\n";

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }
```

```
        projectArn = args[0];
        datasetType = args[1];
        datasetSourceArn = args[2];

        try {

            // Get the Rekognition client
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Create the dataset
            datasetArn = createMyDataset(rekClient, projectArn, datasetType,
 datasetSourceArn);

            System.out.println(String.format("Created dataset: %s",
 datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Beschreibung eines Datensatzes (SDK)

Sie können die`DescribeDataset` -API verwenden, um Informationen über ein Dataset abzurufen.

Um einen Datensatz zu beschreiben (SDK)

1. Falls noch erfolgt, noch erfolgtAWS CLI undAWS konfigurieren. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Beispielcode, um einen Datensatz zu beschreiben.

   AWS CLI

   Ändern Sie den Wert von`dataset-arn` in den ARN des Datensatzes, den Sie beschreiben möchten.

   ```
   aws rekognition describe-dataset --dataset-arn dataset_arn \
     --profile custom-labels-access
   ```

   Python

   Verwenden Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

   - dataset_arn — der ARN des Datensatzes, den Sie beschreiben möchten.

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   Purpose
   Shows how to describe an Amazon Rekognition Custom Labels dataset.
   """

   import argparse
   import logging
   import boto3

   from botocore.exceptions import ClientError

   logger = logging.getLogger(__name__)


   def describe_dataset(rek_client, dataset_arn):
       """
       Describes an Amazon Rekognition Custom Labels dataset.
       :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
   ```

```
    :param dataset_arn: The ARN of the dataset that you want to describe.

    """

    try:
        # Describe the dataset
        logger.info("Describing dataset %s", dataset_arn)

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

        description = dataset['DatasetDescription']

        print(f"Created: {str(description['CreationTimestamp'])}")
        print(f"Updated: {str(description['LastUpdatedTimestamp'])}")
        print(f"Status: {description['Status']}")
        print(f"Status message: {description['StatusMessage']}")
        print(f"Status code: {description['StatusMessageCode']}")
        print("Stats:")
        print(
            f"\tLabeled entries: {description['DatasetStats']
['LabeledEntries']}")
        print(
            f"\tTotal entries: {description['DatasetStats']['TotalEntries']}")
        print(f"\tTotal labels: {description['DatasetStats']['TotalLabels']}")

    except ClientError as err:
        logger.exception("Couldn't describe dataset: %s",
                         err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to describe."
    )


def main():
```

```
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Describing dataset {args.dataset_arn}")

        # Describe the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_dataset(rekognition_client, args.dataset_arn)

        print(f"Finished describing dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message=f"Problem describing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem describing dataset: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

- dataset_arn — der ARN des Datensatzes, den Sie beschreiben möchten.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
```

```
package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStats;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class DescribeDataset {

    public static final Logger logger =
 Logger.getLogger(DescribeDataset.class.getName());

    public static void describeMyDataset(RekognitionClient rekClient, String
 datasetArn) {

        try {

            DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder().datasetArn(datasetArn)
                    .build();
            DescribeDatasetResponse describeDatasetResponse =
 rekClient.describeDataset(describeDatasetRequest);

            DatasetDescription datasetDescription =
 describeDatasetResponse.datasetDescription();
            DatasetStats datasetStats = datasetDescription.datasetStats();

            System.out.println("ARN: " + datasetArn);
            System.out.println("Created: " +
 datasetDescription.creationTimestamp().toString());
            System.out.println("Updated: " +
 datasetDescription.lastUpdatedTimestamp().toString());
            System.out.println("Status: " +
 datasetDescription.statusAsString());
            System.out.println("Message: " +
 datasetDescription.statusMessage());
```

```
            System.out.println("Total Labels: " +
 datasetStats.totalLabels().toString());
            System.out.println("Total entries: " +
 datasetStats.totalEntries().toString());
            System.out.println("Entries with labels: " +
 datasetStats.labeledEntries().toString());
            System.out.println("Entries with at least 1 error: " +
 datasetStats.errorEntries().toString());

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            throw rekError;
        }

    }

    public static void main(String[] args) {

        final String USAGE = "\n" + "Usage: " + "<dataset_arn>\n\n" + "Where:\n"
                + "   dataset_arn - The ARN of the dataset that you want to
 describe.\n\n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String datasetArn = args[0];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

             // Describe the dataset.
            describeMyDataset(rekClient, datasetArn);

            rekClient.close();
```

```
        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Datensatzeinträge auflisten (SDK)

Sie können die`ListDatasetEntries` API verwenden, um die JSON-Zeilen für jedes Bild in einem Datensatz aufzulisten. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei.

Um Datensatzeinträge aufzulisten (SDK)

1. Falls noch erfolgt, noch erfolgtAWS CLI undAWS konfigurieren. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Beispielcode, um die Einträge in einem Datensatz aufzulisten

   AWS CLI

   Ändern Sie den Wert von`dataset-arn` in den ARN des Datensatzes, den Sie auflisten möchten.

   ```
   aws rekognition list-dataset-entries --dataset-arn dataset_arn \
     --profile custom-labels-access
   ```

   Um nur JSON-Zeilen mit Fehlern aufzulisten, geben Sie an`has-errors`.

   ```
   aws rekognition list-dataset-entries --dataset-arn dataset_arn \
     --has-errors \
     --profile custom-labels-access
   ```

   Python

   Verwenden Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

   - dataset_arn — der ARN des Datensatzes, den Sie auflisten möchten.

- show_errors_only — geben Sie an,`true` ob Sie nur Fehler sehen möchten. `false`sonst.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to list the entries in an Amazon Rekognition Custom Labels dataset.
"""

import argparse
import logging
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def list_dataset_entries(rek_client, dataset_arn, show_errors):
    """
    Lists the entries in an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataet that you want to use.
    """

    try:
        # List the entries.
        logger.info("Listing dataset entries for the dataset %s.", dataset_arn)

        finished = False
        count = 0
        next_token = ""
        show_errors_only = False

        if show_errors.lower() == "true":
            show_errors_only = True

        while finished is False:

            response = rek_client.list_dataset_entries(
                DatasetArn=dataset_arn,
```

```
                HasErrors=show_errors_only,
                MaxResults=100,
                NextToken=next_token)

            count += len(response['DatasetEntries'])

            for entry in response['DatasetEntries']:
                print(entry)

            if 'NextToken' not in response:
                finished = True
                logger.info("No more entries. Total:%s", count)
            else:
                next_token = next_token = response['NextToken']
                logger.info("Getting more entries. Total so far :%s", count)

    except ClientError as err:
        logger.exception(
            "Couldn't list dataset: %s",
             err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to list."

    )

    parser.add_argument(
        "show_errors_only", help="true if you want to see errors only. false
 otherwise."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")
```

```
    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Listing entries for  dataset {args.dataset_arn}")

        # List the dataset entries.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        list_dataset_entries(rekognition_client,
                             args.dataset_arn,
                             args.show_errors_only)

        print(f"Finished listing entries for dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message = f"Problem listing dataset: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem listing dataset: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

Verwenden Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- dataset_arn — der ARN des Datensatzes, den Sie auflisten möchten.
- show_errors_only — geben Sie an,`true` ob Sie nur Fehler sehen möchten. `false`sonst.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.ListDatasetEntriesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;
import
 software.amazon.awssdk.services.rekognition.paginators.ListDatasetEntriesIterable;


import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ListDatasetEntries {

    public static final Logger logger =
 Logger.getLogger(ListDatasetEntries.class.getName());

    public static void listMyDatasetEntries(RekognitionClient rekClient, String
 datasetArn, boolean showErrorsOnly)
            throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Listing dataset {0}", new Object[]
 { datasetArn });

            ListDatasetEntriesRequest listDatasetEntriesRequest =
 ListDatasetEntriesRequest.builder()

 .hasErrors(showErrorsOnly).datasetArn(datasetArn).maxResults(1).build();

            ListDatasetEntriesIterable datasetEntriesList = rekClient
                    .listDatasetEntriesPaginator(listDatasetEntriesRequest);

            datasetEntriesList.stream().flatMap(r ->
 r.datasetEntries().stream())
```

```
                    .forEach(datasetEntry ->
 System.out.println(datasetEntry.toString()));

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not update dataset: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        boolean showErrorsOnly = false;
        String datasetArn = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <dataset_arn>
 <updates_file>\n\n" + "Where:\n"
                + "   dataset_arn - the ARN of the dataset that you want to
 update.\n\n"
                + "   show_errors_only - true to show only errors. false
 otherwise.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        datasetArn = args[0];
        if (args[1].toLowerCase().equals("true")) {

            showErrorsOnly = true;
        }

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

             // list the dataset entries.
```

```
            listMyDatasetEntries(rekClient, datasetArn, showErrorsOnly);

            System.out.println(String.format("Finished listing entries for :
 %s", datasetArn));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Verteilen eines Trainingsdatensatzes (SDK)

Amazon Rekognition Custom Labels benötigt einen Trainingsdatensatz und einen Testdatensatz, um Ihr Modell zu trainieren.

Wenn Sie die API verwenden, können Sie die API verwenden, um 20% des Trainingsdatensatzes in einen leeren Testdatensatz zu verteilen. DistributeDatasetEntries Das Verteilen des Trainingsdatensatzes kann nützlich sein, wenn nur eine einzige Manifestdatei verfügbar ist. Verwenden Sie die einzelne Manifestdatei, um Ihren Trainingsdatensatz zu erstellen. Erstellen Sie dann einen leeren Testdatensatz und verwenden Sie ihn`DistributeDatasetEntries`, um den Testdatensatz zu füllen.

> **Note**
>
> Wenn Sie die Amazon Rekognition Custom Labels-Konsole verwenden und mit einem einzelnen Datensatzprojekt beginnen, teilt (verteilt) Amazon Rekognition Custom Labels den Trainingsdatensatz während des Trainings, um einen Testdatensatz zu erstellen. 20% der Trainingsdatensatzeinträge werden in den Testdatensatz verschoben.

Um einen Trainingsdatensatz zu verteilen (SDK)

1. Falls noch erfolgt, noch erfolgtAWS CLI undAWS konfigurieren. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Erstellen eines Projekts Weitere Informationen finden Sie unter Erstellen eines Amazon-Rekognition-Custom-Labels-Projekts (SDK).
3. Erstellen Sie Ihren Trainingsdatensatz. Hinweise zu Datensätzen finden Sie unterTrainings- und Testdatensätze erstellen.
4. Erstellen Sie ein leeres TestDataset.
5. Verwenden Sie den folgenden Beispielcode, um 20% der Trainingsdatensatzeinträge in den Testdatensatz zu verteilen. Sie können die Amazon Resource Names (ARN) für die Datensätze eines Projekts abrufen, indem Sie anrufen DescribeProjects. Beispielcode finden Sie unter Beschreibung eines Projekts (SDK).

   AWS CLI

   Ändern Sie den Wert von`training_dataset-arn` und`test_dataset_arn` mit den ARNS der Datensätze, die Sie verwenden möchten.

   ```
   aws rekognition distribute-dataset-entries --datasets ['{"Arn":
    "training_dataset_arn"}, {"Arn": "test_dataset_arn"}'] \
     --profile custom-labels-access
   ```

   Python

   Verwenden Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

   - training_dataset_arn — der ARN des Trainingsdatensatzes, aus dem Sie Einträge verteilen.
   - test_dataset_arn — der ARN des Testdatensatzes, an den Sie Einträge verteilen.

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   import argparse
   import logging
   import time
   import json
   import boto3
   ```

```
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def check_dataset_status(rek_client, dataset_arn):
    """
    Checks the current status of a dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The dataset that you want to check.
    :return: The dataset status and status message.
    """
    finished = False
    status = ""
    status_message = ""

    while finished is False:

        dataset = rek_client.describe_dataset(DatasetArn=dataset_arn)

        status = dataset['DatasetDescription']['Status']
        status_message = dataset['DatasetDescription']['StatusMessage']

        if status == "UPDATE_IN_PROGRESS":

            logger.info("Distributing dataset: %s ", dataset_arn)
            time.sleep(5)
            continue

        if status == "UPDATE_COMPLETE":
            logger.info(
                "Dataset distribution complete: %s : %s : %s",
                    status, status_message, dataset_arn)
            finished = True
            continue

        if status == "UPDATE_FAILED":
            logger.exception(
                "Dataset distribution failed: %s : %s : %s",
                    status, status_message, dataset_arn)
            finished = True
            break
```

```
        logger.exception(
            "Failed. Unexpected state for dataset distribution: %s : %s : %s",
            status, status_message, dataset_arn)
        finished = True
        status_message = "An unexpected error occurred while distributing the
 dataset"
        break

    return status, status_message


def distribute_dataset_entries(rek_client, training_dataset_arn,
 test_dataset_arn):
    """
    Distributes 20% of the supplied training dataset into the supplied test
 dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param training_dataset_arn: The ARN of the training dataset that you
 distribute entries from.
    :param test_dataset_arn: The ARN of the test dataset that you distribute
 entries to.
    """

    try:
        # List dataset labels.
        logger.info("Distributing training dataset entries (%s) into test
 dataset (%s).",
            training_dataset_arn,test_dataset_arn)


        datasets = json.loads(
            '[{"Arn" : "' + str(training_dataset_arn) + '"},{"Arn" : "' +
 str(test_dataset_arn) + '"}]')

        rek_client.distribute_dataset_entries(
            Datasets=datasets
        )

        training_dataset_status, training_dataset_status_message =
 check_dataset_status(
            rek_client, training_dataset_arn)
        test_dataset_status, test_dataset_status_message = check_dataset_status(
            rek_client, test_dataset_arn)
```

```
        if training_dataset_status == 'UPDATE_COMPLETE' and test_dataset_status
 == "UPDATE_COMPLETE":
            print("Distribution complete")
        else:
            print("Distribution failed:")
            print(
                f"\ttraining dataset: {training_dataset_status} :
 {training_dataset_status_message}")
            print(
                f"\ttest dataset: {test_dataset_status} :
 {test_dataset_status_message}")

    except ClientError as err:
        logger.exception(
            "Couldn't distribute dataset: %s",err.response['Error']['Message'] )
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "training_dataset_arn", help="The ARN of the training dataset that you
 want to distribute from."
    )

    parser.add_argument(
        "test_dataset_arn", help="The ARN of the test dataset that you want to
 distribute to."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
```

```
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Distributing training dataset entries
 ({args.training_dataset_arn}) "\
            f"into test dataset ({args.test_dataset_arn}).")

        # Distribute the datasets.

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        distribute_dataset_entries(rekognition_client,
                                   args.training_dataset_arn,
                                   args.test_dataset_arn)

        print("Finished distributing datasets.")

    except ClientError as err:
        logger.exception("Problem distributing datasets: %s", err)
        print(f"Problem listing dataset labels: {err}")
    except Exception as err:
        logger.exception("Problem distributing datasets: %s", err)
        print(f"Problem distributing datasets: {err}")


if __name__ == "__main__":
    main()
```

Java V2

Verwenden Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- training_dataset_arn — der ARN des Trainingsdatensatzes, aus dem Sie Einträge verteilen.
- test_dataset_arn — der ARN des Testdatensatzes, an den Sie Einträge verteilen.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;
```

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DatasetDescription;
import software.amazon.awssdk.services.rekognition.model.DatasetStatus;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DistributeDataset;
import
 software.amazon.awssdk.services.rekognition.model.DistributeDatasetEntriesRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

public class DistributeDatasetEntries {

    public static final Logger logger =
 Logger.getLogger(DistributeDatasetEntries.class.getName());

    public static DatasetStatus checkDatasetStatus(RekognitionClient rekClient,
 String datasetArn)
            throws Exception, RekognitionException {

        boolean distributed = false;
        DatasetStatus status = null;

        // Wait until distribution completes

        do {

            DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder().datasetArn(datasetArn)
                    .build();
            DescribeDatasetResponse describeDatasetResponse =
 rekClient.describeDataset(describeDatasetRequest);

            DatasetDescription datasetDescription =
 describeDatasetResponse.datasetDescription();

            status = datasetDescription.status();
```

```
            logger.log(Level.INFO, " dataset ARN: {0} ", datasetArn);

            switch (status) {

            case UPDATE_COMPLETE:
                logger.log(Level.INFO, "Dataset updated");
                distributed = true;
                break;

            case UPDATE_IN_PROGRESS:
                Thread.sleep(5000);
                break;

            case UPDATE_FAILED:
                String error = "Dataset distribution failed: " +
datasetDescription.statusAsString() + " "
                        + datasetDescription.statusMessage() + " " + datasetArn;
                logger.log(Level.SEVERE, error);
                break;

            default:
                String unexpectedError = "Unexpected distribution state: " +
datasetDescription.statusAsString() + " "
                        + datasetDescription.statusMessage() + " " + datasetArn;
                logger.log(Level.SEVERE, unexpectedError);

            }

        } while (distributed == false);

        return status;

    }

    public static void distributeMyDatasetEntries(RekognitionClient rekClient,
String trainingDatasetArn,
            String testDatasetArn) throws Exception, RekognitionException {

        try {

            logger.log(Level.INFO, "Distributing {0} dataset to {1} ",
                    new Object[] { trainingDatasetArn, testDatasetArn });
```

```
            DistributeDataset distributeTrainingDataset =
DistributeDataset.builder().arn(trainingDatasetArn).build();

            DistributeDataset distributeTestDataset =
DistributeDataset.builder().arn(testDatasetArn).build();

            ArrayList<DistributeDataset> datasets = new ArrayList();

            datasets.add(distributeTrainingDataset);
            datasets.add(distributeTestDataset);

            DistributeDatasetEntriesRequest distributeDatasetEntriesRequest =
DistributeDatasetEntriesRequest.builder()
                    .datasets(datasets).build();

            rekClient.distributeDatasetEntries(distributeDatasetEntriesRequest);

            DatasetStatus trainingStatus = checkDatasetStatus(rekClient,
trainingDatasetArn);
            DatasetStatus testStatus = checkDatasetStatus(rekClient,
testDatasetArn);

            if (trainingStatus == DatasetStatus.UPDATE_COMPLETE && testStatus ==
DatasetStatus.UPDATE_COMPLETE) {
                logger.log(Level.INFO, "Successfully distributed dataset: {0}",
trainingDatasetArn);

            } else {

                throw new Exception("Failed to distribute dataset: " +
trainingDatasetArn);
            }

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not distribute dataset: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String[] args) {

        String trainingDatasetArn = null;
```

```
        String testDatasetArn = null;

        final String USAGE = "\n" + "Usage: " + "<training_dataset_arn>
 <test_dataset_arn>\n\n" + "Where:\n"
                + "   training_dataset_arn - the ARN of the dataset that you
 want to distribute from.\n\n"
                + "   test_dataset_arn - the ARN of the dataset that you want to
 distribute to.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        trainingDatasetArn = args[0];
        testDatasetArn = args[1];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();

            // Distribute the dataset
            distributeMyDatasetEntries(rekClient, trainingDatasetArn,
 testDatasetArn);

            System.out.println("Datasets distributed.");

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (Exception rekError) {
            logger.log(Level.SEVERE, "Error: {0}", rekError.getMessage());
            System.exit(1);
        }

    }
```

```
}
```

Löschen eines Dataset

Sie können die Trainings- und Testdatensätze aus einem Projekt löschen.

Themen

- Löschen eines Dataset (Konsole)
- Löschen eines Amazon-Rekognition-Custom-Labels-Datensatzes (SDK)

Löschen eines Dataset (Konsole)

Gehen Sie wie folgt vor, um ein Dataset zu löschen. Wenn das Projekt noch einen Datensatz übrig hat (Zug oder Test), wird anschließend die Seite mit den Projektdetails angezeigt. Wenn das Projekt keine weiteren Datensätze enthält, wird die Seite Datensatz erstellen angezeigt.

Wenn Sie den Trainingsdatensatz löschen, müssen Sie einen neuen Trainingsdatensatz für das Projekt erstellen, bevor Sie ein Modell trainieren können. Weitere Informationen finden Sie unter Erstellen von Trainings- und Testdatensätzen mit Bildern.

Wenn Sie den Testdatensatz löschen, können Sie ein Modell trainieren, ohne einen neuen Testdatensatz zu erstellen. Während des Trainings wird der Trainingsdatensatz aufgeteilt, um einen neuen Testdatensatz für das Projekt zu erstellen. Durch die Aufteilung des Trainingsdatensatzes wird die Anzahl der für das Training verfügbaren Bilder reduziert. Um die Qualität aufrechtzuerhalten, empfehlen wir, vor dem Training eines Modells einen neuen Testdatensatz zu erstellen. Weitere Informationen finden Sie unter Hinzufügen eines Datensatzes zu einem Projekt.

Löschen Sie ein Dataset

1. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie im linken Bereich die Option „Benutzerdefinierte Bezeichnungen“ aus. Die Amazon Rekognition Custom Labels Landingpage wird angezeigt.
3. Wählen Sie im linken Navigationsbereich die Option Projekte aus. Die Projektansicht wird angezeigt.
4. Wählen Sie das Projekt aus, das Sie löschen möchten.

5. Wählen Sie im linken Navigationsbereich unter dem Projektnamen die Option Dataset aus
6. Wählen Sie Aktionen
7. Um den Trainingsdatensatz zu löschen, wählen Sie Trainingsdatensatz löschen.
8. Um den Testdatensatz zu löschen, wählen Sie Testdatensatz löschen.
9. Geben Sie im Dialogfeld Zug- oder Testdatensatz löschen den Befehl delete ein, um zu bestätigen, dass Sie den Datensatz löschen möchten.
10. Wählen Sie Zug- oder Testdatensatz löschen, um den Datensatz zu löschen.

Löschen eines Amazon-Rekognition-Custom-Labels-Datensatzes (SDK)

Sie löschen Amazon Rekognition Amazon-Ressourcennamen (ARN) des Datensatzes, den Sie löschen möchten. DeleteDataset Rufen Sie an, um die ARNs der Trainings- und Testdatensätze innerhalb eines Projekts zu erhalten DescribeProjects. Die Antwort umfasst eine Reihe von ProjectDescriptionObjekten. Die Datensatz-ARNs (`DatasetArn`) und Datensatztypen (`DatasetType`) sind in der`Datasets` Liste enthalten.

Wenn Sie den Trainingsdatensatz löschen, müssen Sie einen neuen Trainingsdatensatz für das Projekt erstellen, bevor Sie ein Modell trainieren können. Wenn Sie den Testdatensatz löschen, müssen Sie einen neuen Testdatensatz erstellen, bevor Sie das Modell trainieren können. Weitere Informationen finden Sie unter Hinzufügen eines Datensatzes zu einem Projekt (SDK).

Um einen Datensatz zu löschen (SDK)

1. Falls noch erfolgt, noch erfolgtAWS CLI undAWS konfigurieren. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie folgenden Code, um ein Dataset zu löschen.

   AWS CLI

   Ändern Sie den Wert von`dataset-arn` mit dem ARN des Datensatzes, den Sie löschen möchten.

   ```
   aws rekognition delete-dataset --dataset-arn dataset-arn \
     --profile custom-labels-access
   ```

   Python

   Verwenden Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- dataset_arn — der ARN des Datensatzes, den Sie löschen möchten.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to delete an Amazon Rekognition Custom Labels dataset.
"""
import argparse
import logging
import time
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def delete_dataset(rek_client, dataset_arn):
    """
    Deletes an Amazon Rekognition Custom Labels dataset.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param dataset_arn: The ARN of the dataset that you want to delete.
    """

    try:
        # Delete the dataset,
        logger.info("Deleting dataset: %s", dataset_arn)

        rek_client.delete_dataset(DatasetArn=dataset_arn)

        deleted = False

        logger.info("waiting for dataset deletion %s", dataset_arn)

        # Dataset might not be deleted yet, so wait.
        while deleted is False:
            try:
                rek_client.describe_dataset(DatasetArn=dataset_arn)
                time.sleep(5)
            except ClientError as err:
```

```
                if err.response['Error']['Code'] == 'ResourceNotFoundException':
                    logger.info("dataset deleted: %s", dataset_arn)
                    deleted = True
                else:
                    raise

        logger.info("dataset deleted: %s", dataset_arn)

        return True

    except ClientError as err:
        logger.exception("Couldn't delete dataset - %s: %s",
                         dataset_arn, err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "dataset_arn", help="The ARN of the dataset that you want to delete."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Deleting dataset: {args.dataset_arn}")

        # Delete the dataset.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")
```

```
        delete_dataset(rekognition_client,
                       args.dataset_arn)

        print(f"Finished deleting dataset: {args.dataset_arn}")

    except ClientError as err:
        error_message = f"Problem deleting dataset: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

Verwenden Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- dataset_arn — der ARN des Datensatzes, den Sie löschen möchten.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/
package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import software.amazon.awssdk.services.rekognition.model.DeleteDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.DeleteDatasetResponse;
import software.amazon.awssdk.services.rekognition.model.DescribeDatasetRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteDataset {

    public static final Logger logger =
 Logger.getLogger(DeleteDataset.class.getName());
```

```
   public static void deleteMyDataset(RekognitionClient rekClient, String
datasetArn) throws InterruptedException {

       try {

           logger.log(Level.INFO, "Deleting dataset: {0}", datasetArn);

           // Delete the dataset

           DeleteDatasetRequest deleteDatasetRequest =
DeleteDatasetRequest.builder().datasetArn(datasetArn).build();

           DeleteDatasetResponse response =
rekClient.deleteDataset(deleteDatasetRequest);

           // Wait until deletion finishes

           DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder().datasetArn(datasetArn)
                   .build();

           Boolean deleted = false;

           do {

               try {

                   rekClient.describeDataset(describeDatasetRequest);
                   Thread.sleep(5000);
               } catch (RekognitionException e) {
                   String errorCode = e.awsErrorDetails().errorCode();
                   if (errorCode.equals("ResourceNotFoundException")) {
                       logger.log(Level.INFO, "Dataset deleted: {0}",
datasetArn);
                       deleted = true;
                   } else {
                       logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
                       throw e;
                   }

               }
```

```
            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Dataset deleted: {0} ", datasetArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
 e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: " + "<dataset_arn>\n\n" + "Where:\n"
                + "   dataset_arn - The ARN of the dataset that you want to
 delete.\n\n";

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String datasetArn = args[0];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
                .region(Region.US_WEST_2)
                .build();


            // Delete the dataset
            deleteMyDataset(rekClient, datasetArn);

            System.out.println(String.format("Dataset deleted: %s",
 datasetArn));

            rekClient.close();
```

```
        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

Verwaltung eines Amazon-Rekognition-Custom-Labels-Modells

Ein Amazon Rekognition Custom Labels-Modell ist ein mathematisches Modell, das das Vorhandensein von Objekten, Szenen und Konzepten in neuen Bildern vorhersagt. Dazu findet es Muster in Bildern, die zum Trainieren des Modells verwendet werden. In diesem Abschnitt erfahren Sie, wie Sie ein Modell trainieren, seine Leistung bewerten und Verbesserungen vornehmen. Es zeigt Ihnen auch, wie Sie ein Modell zur Verwendung verfügbar machen und wie Sie ein Modell löschen, wenn Sie es nicht mehr benötigen.

Themen

- Löschen eines Amazon-Rekognition-Custom-Labels-Modells
- Markieren eines Modells
- Beschreibung eines Modells (SDK)
- Kopieren eines Amazon-Rekognition-Custom-Labels-Modells (SDK)

Löschen eines Amazon-Rekognition-Custom-Labels-Modells

Sie können ein Modell löschen, indem Sie die Amazon Rekognition Custom Labels-Konsole oder die DeleteProjectVersionAPI verwenden. Sie können ein Modell nicht löschen, wenn es läuft oder trainiert. Verwenden Sie die StopProjectVersionAPI, um ein laufendes Modell zu stoppen. Weitere

Informationen finden Sie unter Stoppen eines Amazon Rekognition Custom Labels-Modells (SDK). Wenn ein Modell trainiert, warten Sie, bis es fertig ist, bevor Sie das Modell löschen.

Ein gelöschtes Modell kann nicht wiederhergestellt werden.

Themen

- Löschen eines Amazon-Rekognition-Custom-Labels-Modells (Konsole)
- Löschen eines Amazon-Rekognition-Custom-Labels-Modells (SDK)

Löschen eines Amazon-Rekognition-Custom-Labels-Modells (Konsole)

Im folgenden Verfahren wird das Löschen eines Modells von einer Projektdetailseite erläutert. Sie können ein Modell auch von der Detailseite eines Modells löschen.

So löschen Sie ein Modell (Konsole)

1. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Benutzerdefinierte Labels verwenden.
3. Wählen Sie Get started (Erste Schritte) aus.
4. Wählen Sie im linken Navigationsbereich die Option Projekte aus.
5. Wählen Sie das Projekt aus, das Modell enthält, das Sie löschen möchten. Die Projektseite Details wird geöffnet.
6. Wählen Sie im Abschnitt Modelle die Modelle aus, die Sie löschen möchten.

   > **Note**
   >
   > Wenn das Modell nicht ausgewählt werden kann, läuft es oder trainiert es und kann nicht gelöscht werden. Überprüfen Sie das Feld Status und versuchen Sie es erneut, nachdem Sie das laufende Modell gestoppt haben, oder warten Sie, bis das Training beendet ist.

7. Wählen Sie Modell löschen und das Dialogfeld Modell löschen wird angezeigt.
8. Geben Sie löschen ein, um den Löschvorgang zu bestätigen.
9. Wählen Sie Löschen, um das Modell zu löschen. Das Löschen des Modells kann eine Weile dauern.

> **Note**
>
> Wenn Sie das Dialogfeld beim Löschen des Modells schließen, werden die Modelle trotzdem gelöscht.

Löschen eines Amazon-Rekognition-Custom-Labels-Modells (SDK)

Sie löschen ein Amazon Rekognition-Custom-Labels-Modell, indem Sie den Amazon-Ressourcennamen (ARN) des Modells aufrufen DeleteProjectVersionund angeben, das Sie löschen möchten. Sie können den Modell-ARN im Abschnitt Verwenden Sie Ihr Modell auf der Seite mit den Modelldetails in der Amazon Rekognition Custom Labels-Konsole abrufen. Rufen Sie alternativ an DescribeProjectVersionsund geben Sie Folgendes an.

- Der ARN des Projekts (`ProjectArn`), mit dem das Modell verknüpft ist.
- Der Versionsname (`VersionNames`) des Modells.

Der Modell-ARN ist das`ProjectVersionArn` Feld im ProjectVersionDescriptionObjekt aus der`DescribeProjectVersions` Antwort.

Sie können ein Modell nicht löschen, wenn es läuft oder trainiert. Um festzustellen, ob das Modell läuft oder trainiert, rufen Sie das`Status` Feld des ProjectVersionDescriptionModellobjekts auf DescribeProjectVersionsund überprüfen Sie es. Verwenden Sie die StopProjectVersionAPI, um ein laufendes Modell zu stoppen. Weitere Informationen finden Sie unter Stoppen eines Amazon Rekognition Custom Labels-Modells (SDK). Sie müssen warten, bis ein Modell das Training abgeschlossen hat, bevor Sie es löschen können.

Um ein Modell zu löschen (SDK)

1. Falls noch nicht getan haben, installieren und konfigurieren Sie dieAWS CLIAWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Löschen Sie folgenden Code, um ein Modell zu löschen.

   AWS CLI

   Ändern Sie den Wert von`project-version-arn` in den Namen des Projekts, das Sie löschen möchten.

```
aws rekognition delete-project-version --project-version-arn model_arn \
  --profile custom-labels-access
```

Python

Geben Sie folgenden Befehlszeilenparameter Sie folgenden Befehlszeilenparameter

- `project_arn`— der ARN des Projekts, das Sie löschen möchten.
- `model_arn`— der ARN der Modellversion, die Sie löschen möchten.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to delete an existing Amazon Rekognition Custom Labels model.
"""


import argparse
import logging
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def find_forward_slash(input_string, n):
    """
    Returns the location of '/' after n number of occurences.
    :param input_string: The string you want to search
    : n: the occurence that you want to find.
    """
    position = input_string.find('/')
    while position >= 0 and n > 1:
        position = input_string.find('/', position + 1)
        n -= 1
    return position
```

```
def delete_model(rek_client, project_arn, model_arn):
    """
    Deletes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param model_arn: The ARN of the model version that you want to delete.
    """

    try:
        # Delete the model
        logger.info("Deleting dataset: {%s}", model_arn)

        rek_client.delete_project_version(ProjectVersionArn=model_arn)

        # Get the model version name
        start = find_forward_slash(model_arn, 3) + 1
        end = find_forward_slash(model_arn, 4)
        version_name = model_arn[start:end]

        deleted = False

        # model might not be deleted yet, so wait deletion finishes.
        while deleted is False:
            describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
            if len(describe_response['ProjectVersionDescriptions']) == 0:
                deleted = True
            else:
                logger.info("Waiting for model deletion %s", model_arn)
                time.sleep(5)

        logger.info("model deleted: %s", model_arn)

        return True

    except ClientError as err:
        logger.exception("Couldn't delete model - %s: %s",
                         model_arn, err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
```

```
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project that contains the model that
 you want to delete."
    )

    parser.add_argument(
        "model_arn", help="The ARN of the model version that you want to
 delete."
    )


def confirm_model_deletion(model_arn):
    """
    Confirms deletion of the model. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(f"Are you sure you wany to delete model {model_arn} ?\n", model_arn)

    start = input("Enter delete to delete your model: ")
    if start == "delete":
        return True
    else:
        return False


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        if confirm_model_deletion(args.model_arn) is True:
            print(f"Deleting model: {args.model_arn}")
```

```
            # Delete the model.
            session = boto3.Session(profile_name='custom-labels-access')
            rekognition_client = session.client("rekognition")

            delete_model(rekognition_client,
                         args.project_arn,
                         args.model_arn)

            print(f"Finished deleting model: {args.model_arn}")
        else:
            print(f"Not deleting model {args.model_arn}")

    except ClientError as err:
        print(f"Problem deleting model: {err}")


if __name__ == "__main__":
    main()
```

Java V2

- `project_arn`— der ARN des Projekts, das Sie löschen möchten.
- `model_arn`— der ARN der Modellversion, die Sie löschen möchten.

```
//Copyright 2021 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//PDX-License-Identifier: MIT-0 (For details, see https://github.com/
awsdocs/amazon-rekognition-custom-labels-developer-guide/blob/master/LICENSE-
SAMPLECODE.)

import java.net.URI;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.services.rekognition.RekognitionClient;

import
 software.amazon.awssdk.services.rekognition.model.DeleteProjectVersionRequest;
import
 software.amazon.awssdk.services.rekognition.model.DeleteProjectVersionResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
```

```
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteModel {

    public static final Logger logger =
 Logger.getLogger(DeleteModel.class.getName());

    public static int findForwardSlash(String modelArn, int n) {

        int start = modelArn.indexOf('/');
        while (start >= 0 && n > 1) {
            start = modelArn.indexOf('/', start + 1);
            n -= 1;
        }
        return start;

    }

    public static void deleteMyModel(RekognitionClient rekClient, String
 projectArn, String modelArn)
            throws InterruptedException {

        try {

            logger.log(Level.INFO, "Deleting model: {0}", projectArn);

            // Delete the model

            DeleteProjectVersionRequest deleteProjectVersionRequest =
 DeleteProjectVersionRequest.builder()
                    .projectVersionArn(modelArn).build();

            DeleteProjectVersionResponse response =
                    rekClient.deleteProjectVersion(deleteProjectVersionRequest);

            logger.log(Level.INFO, "Status: {0}", response.status());

            // Get the model version

            int start = findForwardSlash(modelArn, 3) + 1;
            int end = findForwardSlash(modelArn, 4);
```

```
            String versionName = modelArn.substring(start, end);

            Boolean deleted = false;

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                    .projectArn(projectArn).versionNames(versionName).build();

            // Wait until model is deleted.

            do {

                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

.describeProjectVersions(describeProjectVersionsRequest);

                if
(describeProjectVersionsResponse.projectVersionDescriptions().size()==0) {
                    logger.log(Level.INFO, "Waiting for model deletion: {0}",
modelArn);
                    Thread.sleep(5000);
                } else {
                    deleted = true;
                    logger.log(Level.INFO, "Model deleted: {0}", modelArn);
                }

            } while (Boolean.FALSE.equals(deleted));

            logger.log(Level.INFO, "Model deleted: {0}", modelArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {
```

```
        final String USAGE = "\n" + "Usage: " + "<project_arn> <model_arn>\n\n"
 + "Where:\n"
                + "   project_arn - The ARN of the project that contains the
 model that you want to delete.\n\n"
                + "   model_version - The ARN of the model that you want to
 delete.\n\n";

        if (args.length != 2) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];
        String modelVersion = args[1];

        try {

            RekognitionClient rekClient = RekognitionClient.builder().build();

            // Delete the model
            deleteMyModel(rekClient, projectArn, modelVersion);

            System.out.println(String.format("model deleted: %s",
 modelVersion));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

Markieren eines Modells

Sie können Ihre Amazon Rekognition Rekognition-Modelle mithilfe von Tags identifizieren, organisieren, nach ihnen suchen und filtern. Jedes Tag ist ein Label, das aus einem benutzerdefinierten Schlüssel und Wert besteht. Um beispielsweise die Abrechnung für Ihre Modelle zu ermitteln, kennzeichnen Sie Ihre Modelle mit einem`Cost center` Schlüssel und fügen Sie die entsprechende Kostenstellennummer als Wert hinzu. Weitere Informationen finden Sie unter Markieren von AWS-Ressourcen.

Verwenden Sie Tags für:

- Verfolgen Sie die Abrechnung für ein Modell mithilfe von Kennzeichnungen für die Kostenzuweisung. Weitere Informationen finden Sie unter Verwendung von Kostenzuordnungs-Tags.
- Steuern Sie den Zugriff auf ein Modell mithilfe von Identity and Access Management (IAM). Weitere Informationen finden Sie unter Steuern des Zugriffs aufAWS -Ressourcen mit Ressourcen-Tags.
- Automatisieren Sie das Modellmanagement. Sie können beispielsweise automatische Start- oder Stoppskripts ausführen, die Entwicklungsmodelle außerhalb der Geschäftszeiten ausschalten, um die Kosten zu senken. Weitere Informationen finden Sie unter Ausführen eines trainierten Amazon Rekognition Custom Labels-Modells.

Sie können Modelle mithilfe der Amazon Rekognition Rekognition-Konsole oder mithilfe derAWS SDKs taggen.

Themen

- Markieren von Modellen (Konsole)
- Anzeigen von Modell-Tags
- Modelle taggen (SDK)

Markieren von Modellen (Konsole)

Sie können die Rekognition-Konsole verwenden, um Tags zu Modellen hinzuzufügen, die an ein Modell angehängten Tags anzuzeigen und Tags zu entfernen.

Hinzufügen oder Entfernen von Tags

In diesem Verfahren wird erklärt, wie Sie einem vorhandenen Modell Tags hinzufügen oder aus einem vorhandenen Modell entfernen. Sie können einem neuen Modell auch Tags (Markierungen) hinzufügen. Weitere Informationen finden Sie unter Schulung eines Amazon-Rekognition-Custom-Labels-Modells.

Hinzufügen von Tags zu einem vorhandenen Modell oder Entfernen von Tags aus einem vorhandenen Modell mithilfe der Konsole

1. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Get started (Erste Schritte) aus.
3. Klicken Sie im Navigationsbereich auf Projects (Projekte).
4. Wählen Sie auf der Seite Projektressourcen das Projekt aus, das das Modell enthält, das Sie taggen möchten.
5. Wählen Sie im Navigationsbereich unter dem zuvor ausgewählten Projekt die Option Modelle aus.
6. Wählen Sie im Abschnitt Modelle das Modell aus, dem Sie ein Tag hinzufügen möchten.
7. Wählen Sie auf der Detailseite des Modells die Registerkarte Tags aus.
8. Wählen Sie im Abschnitt Tags (Markierungen) die Option Manage tags (Tags (Markierungen) verwalten).
9. Wählen Sie auf der Seite „Schlagworte verwalten“ die Option Neues Tag hinzufügen aus.
10. Geben Sie einen Schlüssel und einen Wert ein.
    a. Geben Sie unter Key (Schlüssel) einen Namen für den Schlüssel ein.
    b. Geben Sie unter Value (Wert) einen Wert ein.
11. Um weitere Tags hinzuzufügen, wiederholen Sie die Schritte 9 und 10.
12. (Optional) Um ein Tag zu entfernen, wählen Sie Remove (Entfernen) neben dem Tag, das Sie entfernen möchten. Wenn Sie ein zuvor gespeichertes Tag entfernen, wird es entfernt, wenn Sie Ihre Änderungen speichern.
13. Wählen Sie Save changes (Änderungen speichern) aus, um die Änderungen zu speichern.

Anzeigen von Modell-Tags

Sie können die Amazon-Rekognition-Konsole verwenden, um die mit einem Modell verknüpften Tags anzuzeigen.

Um die an alle Modelle innerhalb eines Projekts angehängten Tags anzuzeigen, müssen Sie das AWS SDK verwenden. Weitere Informationen finden Sie unter Auflisten von Modell-Tags.

So sehen Sie die einem Modell angefügten Tags

1. Öffnen Sie die Amazon Rekognition Rekognition-Konsole unter https://console.aws.amazon.com/rekognition/.
2. Wählen Sie Get started (Erste Schritte) aus.
3. Klicken Sie im Navigationsbereich auf Projects (Projekte).
4. Wählen Sie auf der Seite Projektressourcen das Projekt aus, das das Modell enthält, dessen Tag Sie anzeigen möchten.
5. Wählen Sie im Navigationsbereich unter dem zuvor ausgewählten Projekt die Option Modelle aus.
6. Wählen Sie im Abschnitt Modelle das Modell aus, dessen Tag Sie anzeigen möchten.
7. Wählen Sie auf der Detailseite des Modells die Registerkarte Tags aus. Die Tags werden im Abschnitt Tags angezeigt.

Modelle taggen (SDK)

Sie können dasAWS SDK verwenden, um:

- Hinzufügen von Tags zu einem neuen Modell
- Hinzufügen von Tags zu einem vorhandenen Modell
- Auflisten Sie einem Modell angefügten Tags auf
- Entfernen von Tags von einem Modell entfernen

Die Tags in den folgendenAWS CLI Beispielen haben das folgende Format.

```
--tags '{"key1":"value1","key2":"value2"}'
```

Alternativ Sie können dieses Format verwenden Sie auch verwenden Sie auch.

```
--tags key1=value1,key2=value2
```

Falls Sie die nicht installiert habenAWS CLI, finden Sie weitere Informationen unterSchritt 4: Richten Sie die AWS CLI und SDKs ein AWS.

Hinzufügen von Tags zu einem neuen Modell

Sie können einem Modell Tags hinzufügen, wenn Sie es mit der CreateProjectVersionOperation erstellen. Geben Sie ein oder mehrere Tags im`Tags` Array-Eingabeparameter an.

```
aws rekognition create-project-version --project-arn project arn \
  --version-name version_name \
  --output-config '{ "S3Location": { "Bucket": "output bucket", "Prefix":  "output
 folder" } }' \
  --tags '{"key1":"value1","key2":"value2"}' \
  --profile custom-labels-access
```

Weitere Informationen zum Erstellen und Trainieren eines Modells finden Sie unterEin Modell trainieren (SDK)

Hinzufügen von Tags zu einem vorhandenen Modell

Führen Sie zum Hinzufügen eines oder mehrerer Tags zu einem vorhandenen Modell den TagResourceVorgang in der aus. Geben Sie den Amazon-Ressourcennamen (ARN) (`ResourceArn`) des Modells sowie die Tags (`Tags`) an, die Sie hinzufügen möchten. Das folgende Beispiel zeigt, wie Sie zwei Tags hinzufügen.

```
aws rekognition tag-resource --resource-arn resource-arn \
  --tags '{"key1":"value1","key2":"value2"}' \
  --profile custom-labels-access
```

Den ARN für ein Modell erhalten Sie telefonisch CreateProjectVersion.

Auflisten von Modell-Tags

Um die an ein Modell angehängten Tags aufzulisten, verwenden Sie die ListTagsForResourceOperation und geben Sie den ARN des Modells an (`ResourceArn`). Die Antwort ist eine Zuordnung von Tag-Schlüsseln und -Werten, die mit dem angegebenen Modell verknüpft sind.

```
aws rekognition list-tags-for-resource --resource-arn resource-arn \
```

```
  --profile custom-labels-access
```

Die Ausgabe zeigt eine Liste der dem Modell angefügten Tags an.

```
{
    "Tags": {
        "Dept": "Engineering",
        "Name": "Ana Silva Carolina",
        "Role": "Developer"
    }
}
```

Um zu sehen, welche Modelle in einem Projekt einen bestimmten Tag haben, rufen Sie an,`DescribeProjectVersions` um eine Liste der Modelle zu erhalten. Rufen Sie dann`ListTagsForResource` für jedes Modell in der Antwort von an`DescribeProjectVersions`. Prüfen Sie die Antwort von`ListTagsForResource`, um festzustellen, ob das erforderliche Tag vorhanden ist.

Das folgende Python 3-Beispiel zeigt Ihnen, wie Sie alle Ihre Projekte nach einem bestimmten Tag-Schlüssel und -Wert durchsuchen. Die Ausgabe enthält den Projekt-ARN und den Modell-ARN, in dem ein passender Schlüssel gefunden wurde.

So suchen Sie nach einem Tag-Wert

1. Speichern Sie folgenden Code in einer Datei mit dem Namen`find_tag.py`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Purpose
Shows how to find a tag value that's associated with models within
your Amazon Rekognition Custom Labels projects.
"""
import logging
import argparse
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)
```

```
def find_tag_in_projects(rekognition_client, key, value):
    """
    Finds Amazon Rekognition Custom Label models tagged with the supplied key and
 key value.
    :param rekognition_client: An Amazon Rekognition boto3 client.
    :param key: The tag key to find.
    :param value: The value of the tag that you want to find.
    return: A list of matching model versions (and model projects) that were found.
    """
    try:

        found_tags = []
        found = False

        projects = rekognition_client.describe_projects()
        # Iterate through each project and models within a project.
        for project in projects["ProjectDescriptions"]:
            logger.info("Searching project: %s ...", project["ProjectArn"])

            models = rekognition_client.describe_project_versions(
                ProjectArn=(project["ProjectArn"])
            )

            for model in models["ProjectVersionDescriptions"]:
                logger.info("Searching model %s", model["ProjectVersionArn"])

                tags = rekognition_client.list_tags_for_resource(
                    ResourceArn=model["ProjectVersionArn"]
                )

                logger.info(
                    "\tSearching model: %s for tag: %s value: %s.",
                    model["ProjectVersionArn"],
                    key,
                    value,
                )
                # Check if tag exists.

                if key in tags["Tags"]:
                    if tags["Tags"][key] == value:
                        found = True
                        logger.info(
                            "\t\tMATCH: Project: %s: model version %s",
```

```
                            project["ProjectArn"],
                            model["ProjectVersionArn"],
                        )
                        found_tags.append(
                            {
                                "Project": project["ProjectArn"],
                                "ModelVersion": model["ProjectVersionArn"],
                            }
                        )

            if found is False:
                logger.info("No match for Tag %s with value %s.", key, value)
            return found_tags
        except ClientError as err:
            logger.info("Problem finding tags: %s. ", format(err))
            raise


def main():
    """
    Entry point for example.
    """
    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    # Set up command line arguments.
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)

    parser.add_argument("tag", help="The tag that you want to find.")
    parser.add_argument("value", help="The tag value that you want to find.")

    args = parser.parse_args()
    key = args.tag
    value = args.value

    print(f"Searching your models for tag: {key} with value: {value}.")


    session = boto3.Session(profile_name='custom-labels-access')
    rekognition_client = session.client("rekognition")

    # Get tagged models for all projects.
    tagged_models = find_tag_in_projects(rekognition_client, key, value)
```

```
    print("Matched models\n--------------")
    if len(tagged_models) > 0:
        for model in tagged_models:
            print(
                "Project: {project}\nModel version: {version}\n".format(
                    project=model["Project"], version=model["ModelVersion"]
                )
            )

    else:
        print("No matches found.")

    print("Done.")


if __name__ == "__main__":
    main()
```

2. Geben Sie in der Eingabeaufforderung Folgendes ein. Ersetzen Sie *Schlüssel* und *Wert* durch den Schlüsselnamen und den Schlüsselwert, den Sie suchen möchten.

```
python find_tag.py key value
```

Löschen von Tags aus einem Modell

Führen Sie zum Entfernen eines oder mehrerer Tags aus, um ein oder mehrere Tags aus, um eine oder mehrere Tags aus, die UntagResourceSie verwenden. Geben Sie den ARN des Modells (`ResourceArn`) und die Tag-Schlüssel (`Tag-Keys`) an, die Sie entfernen möchten.

```
aws rekognition untag-resource --resource-arn resource-arn \
  --tag-keys '["key1","key2"]' \
  --profile custom-labels-access
```

Alternativ können Sie`tag-keys` in diesem Format angeben.

```
--tag-keys key1,key2
```

Beschreibung eines Modells (SDK)

Sie können die`DescribeProjectVersions` API verwenden, um Informationen über eine Version eines Modells abzurufen. Wenn Sie nichts angeben`VersionName`, werden Beschreibungen für alle Modellversionen im Projekt`DescribeProjectVersions` zurückgegeben.

Um ein Modell zu beschreiben (SDK)

1. Falls noch nicht getan haben, installieren und konfigurieren Sie dieAWS CLIAWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Beispielcode, um eine Version eines Modells zu beschreiben.

   AWS CLI

   Ändern Sie den Wert von`project-arn` in den ARN des Projekts, das Sie beschreiben möchten. Ändern Sie`version-name` den Wert von auf die Sie beschreiben möchten.

   ```
   aws rekognition describe-project-versions --project-arn project_arn \
     --version-names version_name \
     --profile custom-labels-access
   ```

   Python

   Verwenden Sie auch Sie folgenden Code Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

   - project_arn — der ARN des Modells, das Sie beschreiben möchten.
   - model_version — die Version des Modells, das Sie beschreiben möchten.

   Beispiel: `python describe_model.py project_arn model_version`

   ```
   # Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   # SPDX-License-Identifier: Apache-2.0

   """
   Purpose
   Shows how to describe an Amazon Rekognition Custom Labels model.
   """
   import argparse
   import logging
   ```

```
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def describe_model(rek_client, project_arn, version_name):
    """
    Describes an Amazon Rekognition Custom Labels model.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The ARN of the prject that contains the model.
    :param version_name: The version name of the model that you want to
 describe.
    """

    try:
        # Describe the model
        logger.info("Describing model: %s for project %s",
                    version_name, project_arn)

        describe_response =
 rek_client.describe_project_versions(ProjectArn=project_arn,

 VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            print(f"Created: {str(model['CreationTimestamp'])} ")
            print(f"ARN: {str(model['ProjectVersionArn'])} ")
            if 'BillableTrainingTimeInSeconds' in model:
                print(
                    f"Billing training time (minutes):
 {str(model['BillableTrainingTimeInSeconds']/60)} ")
            print("Evaluation results: ")
            if 'EvaluationResult' in model:
                evaluation_results = model['EvaluationResult']
                print(f"\tF1 score: {str(evaluation_results['F1Score'])}")
                print(
                    f"\tSummary location: s3://{evaluation_results['Summary']
['S3Object']['Bucket']}/{evaluation_results['Summary']['S3Object']['Name']}")

            if 'ManifestSummary' in model:
                print(
                    f"Manifest summary location: s3://{model['ManifestSummary']
['S3Object']['Bucket']}/{model['ManifestSummary']['S3Object']['Name']}")
```

```
            if 'OutputConfig' in model:
                print(
                    f"Training output location: s3://{model['OutputConfig']
['S3Bucket']}/{model['OutputConfig']['S3KeyPrefix']}")
            if 'MinInferenceUnits' in model:
                print(
                    f"Minimum inference units:
 {str(model['MinInferenceUnits'])}")
            if 'MaxInferenceUnits' in model:
                print(
                    f"Maximum Inference units:
 {str(model['MaxInferenceUnits'])}")

            print("Status: " + model['Status'])
            print("Message: " + model['StatusMessage'])

    except ClientError as err:
        logger.exception(
            "Couldn't describe model: %s", err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn", help="The ARN of the project in which the model resides."
    )
    parser.add_argument(
        "version_name", help="The version of the model that you want to
 describe."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:
```

```
        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Describing model: {args.version_name} for project
 {args.project_arn}.")

        # Describe the model.
        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        describe_model(rekognition_client, args.project_arn,
                       args.version_name)

        print(
            f"Finished describing model: {args.version_name} for project
 {args.project_arn}.")

    except ClientError as err:
        error_message = f"Problem describing model: {err}"
        logger.exception(error_message)
        print(error_message)
    except Exception as err:
        error_message = f"Problem describing model: {err}"
        logger.exception(error_message)
        print(error_message)


if __name__ == "__main__":
    main()
```

Java V2

Verwenden Sie auch Sie folgenden Code Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- project_arn — der ARN des Modells, das Sie beschreiben möchten.
- model_version — die Version des Modells, das Sie beschreiben möchten.

```
/*
```

```
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.EvaluationResult;
import software.amazon.awssdk.services.rekognition.model.GroundTruthManifest;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class DescribeModel {

    public static final Logger logger =
 Logger.getLogger(DescribeModel.class.getName());

    public static void describeMyModel(RekognitionClient rekClient, String
 projectArn, String versionName) {

        try {

            // If a single version name is supplied, build request argument

            DescribeProjectVersionsRequest describeProjectVersionsRequest =
 null;

            if (versionName == null) {
                describeProjectVersionsRequest =
 DescribeProjectVersionsRequest.builder().projectArn(projectArn)
                        .build();
            } else {
```

```
                describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder().projectArn(projectArn)
                        .versionNames(versionName).build();
            }

            DescribeProjectVersionsResponse describeProjectVersionsResponse =
rekClient
                    .describeProjectVersions(describeProjectVersionsRequest);

            for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                    .projectVersionDescriptions()) {

                System.out.println("ARN: " +
projectVersionDescription.projectVersionArn());
                System.out.println("Status: " +
projectVersionDescription.statusAsString());
                System.out.println("Message: " +
projectVersionDescription.statusMessage());

                if (projectVersionDescription.billableTrainingTimeInSeconds() !=
null) {
                    System.out.println(
                            "Billable minutes: " +
(projectVersionDescription.billableTrainingTimeInSeconds() / 60));
                }

                if (projectVersionDescription.evaluationResult() != null) {
                    EvaluationResult evaluationResult =
projectVersionDescription.evaluationResult();

                    System.out.println("F1 Score: " +
evaluationResult.f1Score());
                    System.out.println("Summary location: s3://" +
evaluationResult.summary().s3Object().bucket() + "/"
                            + evaluationResult.summary().s3Object().name());
                }

                if (projectVersionDescription.manifestSummary() != null) {
                    GroundTruthManifest manifestSummary =
projectVersionDescription.manifestSummary();
                    System.out.println("Manifest summary location: s3://" +
manifestSummary.s3Object().bucket() + "/"
                            + manifestSummary.s3Object().name());
```

```
                }

                if (projectVersionDescription.outputConfig() != null) {
                    OutputConfig outputConfig =
 projectVersionDescription.outputConfig();
                    System.out.println(
                            "Training output: s3://" + outputConfig.s3Bucket() +
 "/" + outputConfig.s3KeyPrefix());
                }

                if (projectVersionDescription.minInferenceUnits() != null) {
                    System.out.println("Min inference units: " +
 projectVersionDescription.minInferenceUnits());
                }

                System.out.println();

            }

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            throw rekError;
        }

    }

    public static void main(String args[]) {

        String projectArn = null;
        String versionName = null;

        final String USAGE = "\n" + "Usage: " + "<project_arn> <version_name>\n
\n" + "Where:\n"
                + "   project_arn - The ARN of the project that contains the
 models you want to describe.\n\n"
                + "   version_name - (optional) The version name of the model
 that you want to describe. \n\n"
                + "                  If you don't specify a value, all model
 versions are described.\n\n";

        if (args.length > 2 || args.length == 0) {
            System.out.println(USAGE);
```

```
            System.exit(1);
        }

        projectArn = args[0];

        if (args.length == 2) {
            versionName = args[1];
        }

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

             // Describe the model
            describeMyModel(rekClient, projectArn, versionName);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Kopieren eines Amazon-Rekognition-Custom-Labels-Modells (SDK)

Sie können den CopyProjectVersionVorgang verwenden, um eine Amazon Rekognition Custom Labels-Modellversion von einem Amazon Rekognition Custom Labels-Quellprojekt in ein Zielprojekt zu kopieren. Das Zielprojekt kann sich in einem anderenAWS Konto oder in demselbenAWS Konto befinden. Ein typisches Szenario ist das Kopieren eines getesteten Modells von einemAWS Entwicklungskonto auf einAWS Produktionskonto.

Alternativ können Sie das Modell im Zielkonto mit dem Quelldatensatz trainieren. Die`CopyProjectVersion` Verwendung Sie folgenden Vorteile:

- Das Verhalten des Modells ist konsistent. Das Modelltraining ist nicht deterministisch und es ist nicht garantiert, dass zwei Modelle, die mit demselben Datensatz trainiert wurden, dieselben Vorhersagen treffen. Durch das Kopieren des Modells mit wird`CopyProjectVersion` sichergestellt, dass das Verhalten des kopierten Modells mit dem Quellmodell übereinstimmt, und Sie müssen das Modell nicht erneut testen.
- Eine Modellausbildung ist nicht erforderlich. Das spart Ihnen Geld, da Ihnen jedes erfolgreiche Training eines Models in Rechnung gestellt wird.

Um ein Modell in ein anderesAWS Konto zu kopieren, müssen Sie ein Amazon Rekognition Custom Labels-Projekt imAWS Zielkonto haben. Informationen zum Erstellen eines Projekts finden Sie unterErstellen eines eines Projekts. Stellen Sie sicher, dass Sie das Projekt imAWS Zielkonto erstellen.

Eine Projektrichtlinie ist eine ressourcenbasierte Richtlinie, die Kopierberechtigungen für die Modellversion festlegt, die Sie kopieren möchten. Sie müssen eine Projektrichtlinie verwenden, wenn sich das Zielprojekt in einem anderenAWS Konto als das Quellprojekt befindet.

Sie müssen keine Projektrichtlinie verwenden, wenn Sie Modellversionen innerhalb desselben Kontos kopieren. Sie können sich jedoch dafür entscheiden, eine Projektrichtlinie für Projekte mit mehreren Konten zu verwenden, wenn Sie mehr Kontrolle über diese Ressourcen wünschen.

Sie hängen die Projektrichtlinie an das Quellprojekt an, indem Sie den PutProjectPolicyVorgang aufrufen.

Sie können es nicht verwenden`CopyProjectVersion`, um ein Modell in ein Projekt in einer anderenAWS Region zu kopieren. Außerdem können Sie ein Modell nicht mit der Amazon Rekognition Custom Labels-Konsole kopieren. In diesen Fällen können Sie das Modell im Zielprojekt mit den Datensätzen trainieren, die zum Trainieren des Quellmodells verwendet wurden. Weitere Informationen finden Sie unter Schulung eines Amazon-Rekognition-Custom-Labels-Modells.

Gehen Sie wie folgt vor, um ein Modell von einem Quellprojekt in ein Zielprojekt zu kopieren:

So kopieren Sie ein Modell

1. Erstellen Sie ein Dokument mit den Projektrichtlinien.
2. Hängen Sie die Projektrichtlinie an das Quellprojekt an.

3. Kopieren Sie das Modell mit der`CopyProjectVersion` Operation.

Um eine Projektrichtlinie aus einem Projekt zu entfernen, rufen Sie an DeleteProjectPolicy. Rufen Sie an, um eine Liste der einem Projekt beigefügten Projektrichtlinien zu erhalten ListProjectPolicies.

Themen

- Erstellen eines Projektrichtliniendokuments
- Eine Projektrichtlinie anhängen (SDK)
- Modell kopieren (SDK)
- Auflistung der Projektrichtlinien (SDK)
- Löschen einer Projektrichtlinie (SDK)

Erstellen eines Projektrichtliniendokuments

Rekognition Custom Labels verwendet eine ressourcenbasierte Richtlinie, die als Projektrichtlinie bezeichnet wird, um die Kopierberechtigungen für eine Modellversion zu verwalten. Eine Projektrichtlinie ist ein Dokument im JSON-Format.

Eine Projektrichtlinie erlaubt oder verweigert einem Principal die Erlaubnis, eine Modellversion von einem Quellprojekt in ein Zielprojekt zu kopieren. Sie benötigen eine Projektrichtlinie, wenn sich das Zielprojekt in einem anderenAWS Konto befindet. Das gilt auch, wenn sich das Zielprojekt im selbenAWS Konto wie das Quellprojekt befindet und Sie den Zugriff auf bestimmte Modellversionen einschränken möchten. Beispielsweise können Sie festlegen, dass die Kopierberechtigungen für eine bestimmte IAM-Rolle innerhalb einesAWS Kontos verweigert werden.

Das folgende Beispiel ermöglicht es dem Principal`arn:aws:iam::111111111111:role/Admin`, die Modellversion zu kopieren`arn:aws:rekognition:us-east-1:123456789012:project/my_project/version/test_1/1627045542080`.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::111111111111:role/Admin"
      },
```

```
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"arn:aws:rekognition:us-east-1:111111111111:project/my_project/
version/test_1/1627045542080"
    }
  ]
}
```

> **Note**
>
> `Action`,`ResourcePrincipal`, und`Effect` sind Pflichtfelder in einem Dokument mit Projektrichtlinien.
> Die einzige unterstützte`action` ist`rekognition:CopyProjectVersion`.
> `NotAction`,`NotResource`, und`NotPrincipal` sind verbotene Felder und dürfen nicht im Dokument mit den Projektrichtlinien enthalten sein.

Wenn Sie keine Projektrichtlinie angeben, kann ein Principal im gleichen AWS Konto wie das Quellprojekt trotzdem ein Modell kopieren, wenn der Auftraggeber eine identitätsbasierte Richtlinie hat, die z. B. `AmazonRekognitionCustomLabelsFullAccess` die Berechtigung zum Anruf erteilt`CopyProjectVersion`.

Das folgende Verfahren erstellt eine Projektrichtliniendokumentdatei, die Sie mit dem Python-Beispiel in verwenden können Eine Projektrichtlinie anhängen (SDK). Wenn Sie den `put-project-policy` AWS CLI Befehl verwenden, geben Sie die Projektrichtlinie als JSON-Zeichenfolge an.

So erstellen Sie ein Dokument mit den Projektrichtlinien

1. Erstellen Sie in einem Texteditor das folgende Dokument. Ändern Sie die folgenden Werte:

   - Effekt — Geben Sie `ALLOW` an, ob die Kopierberechtigung erteilt werden soll. Geben Sie `DENY` an, ob die Kopierberechtigung verweigert werden soll.
   - Principal — An den Principal, in dem Sie den Zugriff auf die von Ihnen angegebenen Modellversionen zulassen oder verweigern möchten `Resource`. Sie können beispielsweise den AWS-Kontoprinzipal für ein anderes AWS Konto angeben. Wir schränken die Prinzipien, die Sie verwenden können, nicht ein. Weitere Informationen finden Sie unter Angeben eines Prinzipals.
   - -Ressource — Der Amazon-Ressourcenname (ARN) der Modellversion, für die Sie die Kopierberechtigungen angeben möchten. Wenn Sie allen Modellversionen innerhalb des Quellprojekts Berechtigungen gewähren möchten, verwenden Sie das folgende

Formatarn:aws:rekognition:*region*:*account*:project/*source project*/version/*

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"ALLOW or DENY",
      "Principal":{
        "AWS":"principal"
      },
      "Action":"rekognition:CopyProjectVersion",
      "Resource":"Model version ARN"
    }
  ]
}
```

2. Speichern Sie die Projektrichtlinie auf Ihrem Computer ab.
3. Hängen Sie die Projektrichtlinie an das Quellprojekt an, indem Sie den Anweisungen unter folgenEine Projektrichtlinie anhängen (SDK).

Eine Projektrichtlinie anhängen (SDK)

Sie fügen einem Amazon Rekognition Custom Labels-Projekt eine Projektrichtlinie hinzu, indem Sie den PutProjectpolicyVorgang aufrufen.

Fügen Sie einem Projekt mehrere Projektrichtlinien hinzu, indem Sie`PutProjectPolicy` jede Projektrichtlinie, die Sie hinzufügen möchten, aufrufen. Sie können einem Projekt bis zu fünf Projektrichtlinien zuordnen. Wenn Sie weitere Projektrichtlinien hinzufügen müssen, können Sie eine Limit-Erhöhung beantragen.

Wenn Sie einem Projekt zum ersten Mal eine eindeutige Projektrichtlinie zuordnen, geben Sie im`PolicyRevisionId` Eingabeparameter keine Revisions-ID an. Die Antwort von`PutProjectPolicy` ist eine Revisions-ID für die Projektrichtlinie, die Amazon Rekognition Custom Labels für Sie erstellt. Sie können die Revisions-ID verwenden, um die neueste Version einer Projektrichtlinie zu aktualisieren oder zu löschen. Amazon Rekognition Custom Labels speichert nur die neueste Version einer Projektrichtlinie. Wenn Sie versuchen, eine vorherige Version einer Projektrichtlinie zu aktualisieren oder zu löschen, wird eine`InvalidPolicyRevisionIdException` Fehlermeldung angezeigt.

Um eine bestehende Projektrichtlinie zu aktualisieren, geben Sie die Revisions-ID der Projektrichtlinie im`PolicyRevisionId` Eingabeparameter an. Die Revisions-IDs für Projektrichtlinien in einem Projekt erhalten Sie telefonisch ListProjectPolicies.

Nachdem Sie eine Projektrichtlinie an ein Quellprojekt angehängt haben, können Sie das Modell aus dem Quellprojekt in das Zielprojekt kopieren. Weitere Informationen finden Sie unter Modell kopieren (SDK).

Um eine Projektrichtlinie aus einem Projekt zu entfernen, rufen Sie an DeleteProjectPolicy. Rufen Sie an, um eine Liste der einem Projekt beigefügten Projektrichtlinien zu erhalten ListProjectPolicies.

Um eine Projektrichtlinie an ein Projekt anzuhängen (SDK)

1. Falls noch nicht getan haben, installieren und konfigurieren Sie dieAWS CLIAWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Erstellen Sie ein Dokument mit den Projektrichtlinien.
3. Verwenden Sie den folgenden Code, um die Projektrichtlinie an das Projekt imAWS Vertrauenskonto anzuhängen, das die Modellversion enthält, die Sie kopieren möchten. Um das Projekt ARN zu erhalten, rufen Sie an DescribeProjects. Um die ARN der Modellversion zu erhalten, rufen Sie an DescribeProjectVersions.

   AWS CLI

   Ändern Sie die folgenden Werte:

   - `project-arn`zum ARN des Quellprojekts in demAWS Trusting-Konto, das die Modellversion enthält, die Sie kopieren möchten.
   - `policy-name`zu einem von Ihnen gewählten Richtliniennamen.
   - `principal`An den Principal, dem Sie den Zugriff auf die von Ihnen angegebenen Modellversionen gewähren oder verweigern möchten`Model version ARN`.
   - `project-version-arn`zum ARN der Modellversion, die Sie kopieren möchten.

   Wenn Sie eine bestehende Projektrichtlinie aktualisieren möchten, geben Sie den`policy-revision-id` Parameter an und geben Sie die Revisions-ID der gewünschten Projektrichtlinie an.

```
aws rekognition put-project-policy \
  --project-arn project-arn \
```

```
  --policy-name policy-name \
  --policy-document '{ "Version":"2012-10-17", "Statement":
[{ "Effect":"ALLOW or DENY", "Principal":{ "AWS":"principal" },
 "Action":"rekognition:CopyProjectVersion", "Resource":"project-version-
arn" }]}' \
  --profile custom-labels-access
```

Python

Verwenden Sie auch Sie folgenden Code Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- `project_arn`— Der ARN des Quellprojekts, an das Sie die Projektrichtlinie anfügen möchten.
- `policy_name`— Ein von Ihnen gewählter Richtlinienname.
- `project_policy`— Die Datei, die das Dokument mit den Projektrichtlinien enthält,.
- `policy_revision_id`— (Fakultativ). Wenn Sie eine vorhandene Version einer Projektrichtlinie aktualisieren möchten, geben Sie die Revisions-ID der Projektrichtlinie an.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to attach a project policy to an Amazon Rekognition Custom Labels
 project.
"""

import boto3
import argparse
import logging
import json
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
```

```
def put_project_policy(rek_client, project_arn, policy_name,
 policy_document_file, policy_revision_id=None):
    """
    Attaches a project policy to an Amazon Rekognition Custom Labels project.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param policy_name: A name for the project policy.
    :param project_arn: The Amazon Resource Name (ARN) of the source project
    that you want to attach the project policy to.
    :param policy_document_file: The JSON project policy document to
    attach to the source project.
    :param policy_revision_id: (Optional) The revision of an existing policy to
 update.
    Pass None to attach new policy.
    :return The revision ID for the project policy.
    """

    try:

        policy_document_json = ""
        response = None

        with open(policy_document_file, 'r') as policy_document:
            policy_document_json = json.dumps(json.load(policy_document))

        logger.info(
            "Attaching %s project_policy to project %s.",
            policy_name, project_arn)

        if policy_revision_id is None:
            response = rek_client.put_project_policy(ProjectArn=project_arn,
                                                     PolicyName=policy_name,

 PolicyDocument=policy_document_json)

        else:
            response = rek_client.put_project_policy(ProjectArn=project_arn,
                                                     PolicyName=policy_name,

 PolicyDocument=policy_document_json,

 PolicyRevisionId=policy_revision_id)

        new_revision_id = response['PolicyRevisionId']
```

```
        logger.info(
            "Finished creating project policy %s. Revision ID: %s",
            policy_name, new_revision_id)

        return new_revision_id

    except ClientError as err:
        logger.exception(
            "Couldn't attach %s project policy to project %s: %s }",
            policy_name, project_arn, err.response['Error']['Message'] )
        raise


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn",  help="The Amazon Resource Name (ARN) of the project "
        "that you want to attach the project policy to."
    )
    parser.add_argument(
        "policy_name",  help="A name for the project policy."

    )

    parser.add_argument(
        "project_policy",  help="The file containing the project policy JSON"
    )

    parser.add_argument(
        "--policy_revision_id",  help="The revision of an existing policy to
 update. "
        "If you don't supply a value, a new project policy is created.",
        required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
```

```
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)

        args = parser.parse_args()

        print(f"Attaching policy to {args.project_arn}")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")


        # Attach a new policy or update an existing policy.

        response = put_project_policy(rekognition_client,
                                      args.project_arn,
                                      args.policy_name,
                                      args.project_policy,
                                      args.policy_revision_id)

        print(
            f"project policy {args.policy_name} attached to project
 {args.project_arn}")
        print(f"Revision ID: {response}")

    except ClientError as err:
        print("Problem attaching project policy: %s", err)


if __name__ == "__main__":
    main()
```

Java V2

Verwenden Sie auch Sie folgenden Code Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- `project_arn`— Der ARN des Quellprojekts, an das Sie die Projektrichtlinie anfügen möchten.

- `project_policy_name`— Ein von Ihnen gewählter Richtlinienname.
- `project_policy_document`— Die Datei, die das Dokument mit den Projektrichtlinien enthält.
- `project_policy_revision_id`— (Fakultativ). Wenn Sie eine vorhandene Version einer Projektrichtlinie aktualisieren möchten, geben Sie die Revisions-ID der Projektrichtlinie an.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.PutProjectPolicyRequest;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class PutProjectPolicy {

    public static final Logger logger =
 Logger.getLogger(PutProjectPolicy.class.getName());

    public static void putMyProjectPolicy(RekognitionClient rekClient, String
 projectArn, String projectPolicyName,
             String projectPolicyFileName, String projectPolicyRevisionId)
 throws IOException {

        try {

            Path filePath = Path.of(projectPolicyFileName);
```

```
            String policyDocument = Files.readString(filePath);

            String[] logArguments = new String[] { projectPolicyFileName,
projectPolicyName };

            PutProjectPolicyRequest putProjectPolicyRequest = null;

            logger.log(Level.INFO, "Attaching Project policy: {0} to project:
{1}", logArguments);

            // Attach the project policy.

            if (projectPolicyRevisionId == null) {
                putProjectPolicyRequest =
PutProjectPolicyRequest.builder().projectArn(projectArn)

.policyName(projectPolicyName).policyDocument(policyDocument).build();
            } else {
                putProjectPolicyRequest =
PutProjectPolicyRequest.builder().projectArn(projectArn)

.policyName(projectPolicyName).policyRevisionId(projectPolicyRevisionId)
                        .policyDocument(policyDocument)

                        .build();
            }

            rekClient.putProjectPolicy(putProjectPolicyRequest);

            logger.log(Level.INFO, "Attached Project policy: {0} to project:
{1}", logArguments);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }
```

```
    public static void main(String args[]) {

        final String USAGE = "\n" + "Usage: "
                + "<project_arn> <project_policy_name> <policy_document>
 <project_policy_revision_id>\n\n" + "Where:\n"
                + "   project_arn - The ARN of the project that you want to
 attach the project policy to.\n\n"
                + "   project_policy_name - A name for the project policy.\n\n"
                + "   project_policy_document - The file name of the project
 policy.\n\n"
                + "   project_policy_revision_id - (Optional) The revision ID of
 the project policy that you want to update.\n\n";

        if (args.length < 3 || args.length > 4) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];
        String projectPolicyName = args[1];
        String projectPolicyDocument = args[2];
        String projectPolicyRevisionId = null;

        if (args.length == 4) {
            projectPolicyRevisionId = args[3];
        }

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();


            // Attach the project policy.
            putMyProjectPolicy(rekClient, projectArn, projectPolicyName,
 projectPolicyDocument,
                    projectPolicyRevisionId);

            System.out.println(
                    String.format("project policy %s: attached to project: %s",
 projectPolicyName, projectArn));
```

```
            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (IOException intError) {
            logger.log(Level.SEVERE, "Exception while reading policy document:
 {0}", intError.getMessage());
            System.exit(1);
        }

    }

}
```

4. Kopieren Sie die Modellversion, indem Sie den Anweisungen unter folgenModell kopieren (SDK).

Modell kopieren (SDK)

Sie können die`CopyProjectVersion` API verwenden, um eine Modellversion von einem Quellprojekt in ein Zielprojekt zu kopieren. Das Zielprojekt kann sich in einem anderenAWS Konto befinden, muss sich jedoch in derselbenAWS Region befinden. Wenn sich das Zielprojekt in einem anderenAWS Konto befindet (oder wenn Sie bestimmte Berechtigungen für eine in einAWS Konto kopierte Modellversion gewähren möchten), müssen Sie dem Quellprojekt eine Projektrichtlinie anhängen. Weitere Informationen finden Sie unter Erstellen eines Projektrichtliniendokuments. Die`CopyProjectVersion` API erfordert Sie Zugriff auf Ihren Amazon S3 Bucket.

Das kopierte Modell enthält die Trainingsergebnisse für das Quellmodell, nicht jedoch die Quelldatensätze.

DasAWS Quellkonto hat kein Eigentum an dem Modell, das in ein Zielkonto kopiert wurde, es sei denn, Sie richten die entsprechenden Berechtigungen ein.

Um ein Modell zu kopieren (SDK)

1. Falls noch nicht getan haben, installieren und konfigurieren Sie dieAWS CLIAWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.

2. Fügen Sie dem Quellprojekt eine Projektrichtlinie hinzu, indem Sie den Anweisungen unter folgenEine Projektrichtlinie anhängen (SDK).
3. Wenn Sie das Modell in ein anderesAWS Konto kopieren, stellen Sie sicher, dass Sie ein Projekt imAWS Zielkonto haben.
4. Verwenden Sie den folgenden Code, um die Modellversion in ein Zielprojekt zu kopieren.

   AWS CLI

   Ändern Sie die folgenden Werte:

   - `source-project-arn`zum ARN des Quellprojekts, das Sie kopieren möchten.
   - `source-project-version-arn`zum ARN der Modellversion, die Sie kopieren möchten.
   - `destination-project-arn`zum ARN des Zielprojekts, in das Sie das Modell kopieren möchten.
   - `version-name`zu einem Versionsnamen für das Modell im Zielprojekt.
   - `bucket`in den S3-Bucket, in den Sie die Trainingsergebnisse für das Quellmodell kopieren möchten.
   - `folder`in den Ordner`bucket`, in den Sie die Trainingsergebnisse für das Quellmodell kopieren möchten.
   - (Optional)`kms-key-id` zur Schlüssel-ID des AWS Key Management Service für das Modell.
   - (Optional)`key` zu einem Tag-Schlüssel Ihrer Wahl.
   - (Optional)`value` zu einem Tag-Wert Ihrer Wahl.

```
aws rekognition copy-project-version \
  --source-project-arn source-project-arn \
  --source-project-version-arn source-project-version-arn \
  --destination-project-arn destination-project-arn \
  --version-name version-name \
  --output-config '{"S3Bucket":"bucket","S3KeyPrefix":"folder"}' \
  --kms-key-id arn:myKey \
  --tags '{"key":"key"}' \
  --profile custom-labels-access
```

Python

Verwenden Sie auch Sie folgenden Code Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- `source_project_arn`— der ARN des Quellprojekts imAWS Quellkonto, das die Modellversion enthält, die Sie kopieren möchten.
- `source_project_version-arn`— der ARN der Modellversion imAWS Quellkonto, die Sie kopieren möchten.
- `destination_project_arn`— der ARN des Zielprojekts, in das Sie das Modell kopieren möchten.
- `destination_version_name`— ein Versionsname für das Modell im Zielprojekt.
- `training_results`— der S3-Speicherort, an den Sie die Trainingsergebnisse für die Quellmodellversion kopieren möchten.
- (Optional)`kms_key_id` zur Schlüssel-ID des AWS Key Management Service für das Modell.
- (Optional)`tag_name` zu einem Tag-Schlüssel Ihrer Wahl.
- (Optional)`tag_value` zu einem Tag-Wert Ihrer Wahl.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

import argparse
import logging
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def copy_model(
    rekognition_client, source_project_arn, source_project_version_arn,
        destination_project_arn, training_results, destination_version_name):
    """
    Copies a version of a Amazon Rekognition Custom Labels model.
```

```
    :param rekognition_client: A Boto3 Amazon Rekognition Custom Labels client.
    :param source_project_arn: The ARN of the source project that contains the
    model that you want to copy.
    :param source_project_version_arn: The ARN of the model version that you
want
    to copy.
    :param destination_project_Arn: The ARN of the project that you want to copy
the model
    to.
    :param training_results: The Amazon S3 location where training results for
the model
    should be stored.
    return: The model status and version.
    """
    try:
        logger.info("Copying model...%s from %s to %s ",
source_project_version_arn,
                    source_project_arn,
                    destination_project_arn)

        output_bucket, output_folder = training_results.replace(
            "s3://", "").split("/", 1)
        output_config = {"S3Bucket": output_bucket,
                         "S3KeyPrefix": output_folder}

        response = rekognition_client.copy_project_version(
            DestinationProjectArn=destination_project_arn,
            OutputConfig=output_config,
            SourceProjectArn=source_project_arn,
            SourceProjectVersionArn=source_project_version_arn,
            VersionName=destination_version_name
        )

        destination_model_arn = response["ProjectVersionArn"]

        logger.info("Destination model ARN: %s", destination_model_arn)

        # Wait until training completes.
        finished = False
        status = "UNKNOWN"
        while finished is False:
            model_description =
rekognition_client.describe_project_versions(ProjectArn=destination_project_arn,
                    VersionNames=[destination_version_name])
```

```
            status = model_description["ProjectVersionDescriptions"][0]
["Status"]

            if status == "COPYING_IN_PROGRESS":
                logger.info("Model copying in progress...")
                time.sleep(60)
                continue

            if status == "COPYING_COMPLETED":
                logger.info("Model was successfully copied.")

            if status == "COPYING_FAILED":
                logger.info(
                    "Model copy failed: %s ",
                    model_description["ProjectVersionDescriptions"][0]
["StatusMessage"])

            finished = True
    except ClientError:
        logger.exception("Couldn't copy model.")
        raise
    else:
        return destination_model_arn, status


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "source_project_arn",
        help="The ARN of the project that contains the model that you want to
 copy."
    )

    parser.add_argument(
        "source_project_version_arn",
        help="The ARN of the model version that you want to copy."
    )

    parser.add_argument(
        "destination_project_arn",
```

```
        help="The ARN of the project which receives the copied model."
    )

    parser.add_argument(
        "destination_version_name",
        help="The version name for the model in the destination project."
    )

    parser.add_argument(
        "training_results",
        help="The S3 location in the destination account that receives the
 training results for the copied model."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(
            f"Copying model version {args.source_project_version_arn} to project
 {args.destination_project_arn}")

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        # Copy the model.

        model_arn, status = copy_model(rekognition_client,
                                       args.source_project_arn,
                                       args.source_project_version_arn,
                                       args.destination_project_arn,
                                       args.training_results,
                                       args.destination_version_name,
                                       )
```

```
        print(f"Finished copying model: {model_arn}")
        print(f"Status: {status}")

    except ClientError as err:
        print(f"Problem copying model: {err}")


if __name__ == "__main__":
    main()
```

Java V2

Verwenden Sie auch Sie folgenden Code Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- `source_project_arn`— der ARN des Quellprojekts imAWS Quellkonto, das die Modellversion enthält, die Sie kopieren möchten.
- `source_project_version-arn`— der ARN der Modellversion imAWS Quellkonto, die Sie kopieren möchten.
- `destination_project_arn`— der ARN des Zielprojekts, in das Sie das Modell kopieren möchten.
- `destination_version_name`— ein Versionsname für das Modell im Zielprojekt.
- `output_bucket`— der S3-Bucket, in den Sie die Trainingsergebnisse für die Quellmodellversion kopieren möchten.
- `output_folder`— der Ordner im S3, in den Sie die Trainingsergebnisse für die Quellmodellversion kopieren möchten.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.CopyProjectVersionRequest;
```

```
import
 software.amazon.awssdk.services.rekognition.model.CopyProjectVersionResponse;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsRequest;
import
 software.amazon.awssdk.services.rekognition.model.DescribeProjectVersionsResponse;
import software.amazon.awssdk.services.rekognition.model.OutputConfig;
import
 software.amazon.awssdk.services.rekognition.model.ProjectVersionDescription;

import software.amazon.awssdk.services.rekognition.model.RekognitionException;

import java.util.logging.Level;
import java.util.logging.Logger;

public class CopyModel {

    public static final Logger logger =
 Logger.getLogger(CopyModel.class.getName());

    public static ProjectVersionDescription copyMyModel(RekognitionClient
 rekClient,
            String sourceProjectArn,
            String sourceProjectVersionArn,
            String destinationProjectArn,
            String versionName,
            String outputBucket,
            String outputFolder) throws InterruptedException {

        try {

            OutputConfig outputConfig =
 OutputConfig.builder().s3Bucket(outputBucket).s3KeyPrefix(outputFolder).build();

            String[] logArguments = new String[] { versionName,
 sourceProjectArn, destinationProjectArn };

            logger.log(Level.INFO, "Copying model {0} for from project {1} to
 project {2}", logArguments);

            CopyProjectVersionRequest copyProjectVersionRequest =
 CopyProjectVersionRequest.builder()
                    .sourceProjectArn(sourceProjectArn)
                    .sourceProjectVersionArn(sourceProjectVersionArn)
```

```
                    .versionName(versionName)
                    .destinationProjectArn(destinationProjectArn)
                    .outputConfig(outputConfig)
                    .build();

            CopyProjectVersionResponse response =
rekClient.copyProjectVersion(copyProjectVersionRequest);

            logger.log(Level.INFO, "Destination model ARN: {0}",
response.projectVersionArn());
            logger.log(Level.INFO, "Copying model...");

            // wait until copying completes.

            boolean finished = false;

            ProjectVersionDescription copiedModel = null;

            while (Boolean.FALSE.equals(finished)) {
                DescribeProjectVersionsRequest describeProjectVersionsRequest =
DescribeProjectVersionsRequest.builder()
                        .versionNames(versionName)
                        .projectArn(destinationProjectArn)
                        .build();

                DescribeProjectVersionsResponse describeProjectVersionsResponse
= rekClient

.describeProjectVersions(describeProjectVersionsRequest);

                for (ProjectVersionDescription projectVersionDescription :
describeProjectVersionsResponse
                        .projectVersionDescriptions()) {

                    copiedModel = projectVersionDescription;

                    switch (projectVersionDescription.status()) {

                        case COPYING_IN_PROGRESS:
                            logger.log(Level.INFO, "Copying model...");
                            Thread.sleep(5000);
                            continue;

                        case COPYING_COMPLETED:
```

```
                            finished = true;
                            logger.log(Level.INFO, "Copying completed");
                            break;

                        case COPYING_FAILED:
                            finished = true;
                            logger.log(Level.INFO, "Copying failed...");
                            break;

                        default:
                            finished = true;
                            logger.log(Level.INFO, "Unexpected copy status %s",
                                    projectVersionDescription.statusAsString());
                            break;

                    }

                }

            }

            logger.log(Level.INFO, "Finished copying model {0} for from project
{1} to project {2}", logArguments);

            return copiedModel;

        } catch (RekognitionException e) {
            logger.log(Level.SEVERE, "Could not train model: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {

        String sourceProjectArn = null;
        String sourceProjectVersionArn = null;
        String destinationProjectArn = null;
        String versionName = null;
        String bucket = null;
        String location = null;

        final String USAGE = "\n" + "Usage: "
```

```
                + "<source_project_arn> <source_project_version_arn>
 <destination_project_arn> <version_name> <output_bucket> <output_folder>\n\n"
                + "Where:\n"
                + "   source_project_arn - The ARN of the project that contains
 the model that you want to copy. \n\n"
                + "   source_project_version_arn - The ARN of the project that
 contains the model that you want to copy. \n\n"
                + "   destination_project_arn - The ARN of the destination
 project that you want to copy the model to. \n\n"
                + "   version_name - A version name for the copied model.\n\n"
                + "   output_bucket - The S3 bucket in which to place the
 training output. \n\n"
                + "   output_folder - The folder within the bucket that the
 training output is stored in. \n\n";

        if (args.length != 6) {
            System.out.println(USAGE);
            System.exit(1);
        }

        sourceProjectArn = args[0];
        sourceProjectVersionArn = args[1];
        destinationProjectArn = args[2];
        versionName = args[3];
        bucket = args[4];
        location = args[5];

        try {

            // Get the Rekognition client.
            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Copy the model.
            ProjectVersionDescription copiedModel = copyMyModel(rekClient,
                    sourceProjectArn,
                    sourceProjectVersionArn,
                    destinationProjectArn,
                    versionName,
                    bucket,
                    location);
```

```
            System.out.println(String.format("Model copied: %s Status: %s",
                    copiedModel.projectVersionArn(),
                    copiedModel.statusMessage()));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        } catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

Auflistung der Projektrichtlinien (SDK)

Sie können den ListProjectPoliciesVorgang verwenden, um die Projektrichtlinien aufzulisten, die an ein Amazon Rekognition Custom Labels-Projekt angehängt sind.

So listen Sie die einem Projekt angefügten Projektrichtlinien auf (SDK)

1. Falls noch nicht getan haben, installieren und konfigurieren Sie dieAWS CLIAWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Code, um die Projektrichtlinien aufzulisten.

   AWS CLI

   Wechseln Sie`project-arn` zum Amazon-Ressourcennamen des Projekts, für das Sie die angehängten Projektrichtlinien auflisten möchten.

   ```
   aws rekognition list-project-policies \
     --project-arn project-arn \
     --profile custom-labels-access
   ```

Python

Verwenden Sie auch Sie folgenden Code Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- project_arn — der Amazon-Ressourcenname des Projekts, für das Sie die angehängten Projektrichtlinien auflisten möchten.

Beispiel: `python list_project_policies.py` *`project_arn`*

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to list the project policies in an Amazon Rekogntion Custom Labels
 project.
"""


import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def display_project_policy(project_policy):
    """
    Displays information about a Custom Labels project policy.
    :param project_policy: The project policy (ProjectPolicy)
    that you want to display information about.
    """
    print(f"Policy name: {(project_policy['PolicyName'])}")
    print(f"Project Arn: {project_policy['ProjectArn']}")
    print(f"Document: {(project_policy['PolicyDocument'])}")
    print(f"Revision ID: {(project_policy['PolicyRevisionId'])}")
```

```
    print()



def list_project_policies(rek_client, project_arn):
    """
    Describes an Amazon Rekognition Custom Labels project, or all projects.
    :param rek_client: The Amazon Rekognition Custom Labels Boto3 client.
    :param project_arn: The Amazon Resource Name of the project you want to use.
    """

    try:

        max_results = 5
        pagination_token = ''
        finished = False

        logger.info("Listing project policies in: %s.", project_arn)
        print('Projects\n--------')
        while not finished:

            response = rek_client.list_project_policies(
                ProjectArn=project_arn, MaxResults=max_results,
 NextToken=pagination_token)

            for project in response['ProjectPolicies']:
                display_project_policy(project)

            if 'NextToken' in response:
                pagination_token = response['NextToken']
            else:
                finished = True

        logger.info("Finished listing project policies.")

    except ClientError as err:
        logger.exception(
            "Couldn't list policies for - %s: %s",
            project_arn,err.response['Error']['Message'])
        raise


def add_arguments(parser):
    """
```

```
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_arn",  help="The Amazon Resource Name of the project for which
 you want to list project policies."
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # get command line arguments
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        print(f"Listing project policies in: {args.project_arn}")

        # List the project policies.

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        list_project_policies(rekognition_client,
                              args.project_arn)

    except ClientError as err:
        print(f"Problem list project_policies: {err}")


if __name__ == "__main__":
    main()
```

Java V2

Verwenden Sie auch Sie folgenden Code Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- project_arn — Der ARN des Projekts, das die Projektrichtlinien enthält, die Sie auflisten möchten.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.ListProjectPoliciesRequest;
import
 software.amazon.awssdk.services.rekognition.model.ListProjectPoliciesResponse;
import software.amazon.awssdk.services.rekognition.model.ProjectPolicy;
import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class ListProjectPolicies {

    public static final Logger logger =
 Logger.getLogger(ListProjectPolicies.class.getName());

    public static void listMyProjectPolicies(RekognitionClient rekClient, String
 projectArn) {

        try {

            logger.log(Level.INFO, "Listing project policies for project: {0}",
 projectArn);

            // List the project policies.

            Boolean finished = false;
            String nextToken = null;

            while (Boolean.FALSE.equals(finished)) {
```

```
                ListProjectPoliciesRequest listProjectPoliciesRequest =
ListProjectPoliciesRequest.builder()
                        .maxResults(5)
                        .projectArn(projectArn)
                        .nextToken(nextToken)
                        .build();

                ListProjectPoliciesResponse response =
rekClient.listProjectPolicies(listProjectPoliciesRequest);

                for (ProjectPolicy projectPolicy : response.projectPolicies()) {

                    System.out.println(String.format("Name: %s",
projectPolicy.policyName()));
                    System.out.println(String.format("Revision ID: %s\n",
projectPolicy.policyRevisionId()));

                }

                nextToken = response.nextToken();

                if (nextToken == null) {
                    finished = true;

                }

            }

            logger.log(Level.INFO, "Finished listing project policies for
project: {0}", projectArn);

        } catch (

        RekognitionException e) {
            logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
            throw e;
        }

    }

    public static void main(String args[]) {
```

```
        final String USAGE = "\n" + "Usage: " + "<project_arn> \n\n" + "Where:
\n"
                + "   project_arn - The ARN of the project with the project
 policies that you want to list.\n\n";
        ;

        if (args.length != 1) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // List the project policies.
            listMyProjectPolicies(rekClient, projectArn);

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

    }

}
```

Löschen einer Projektrichtlinie (SDK)

Sie können den DeleteProjectPolicyVorgang verwenden, um eine Revision einer vorhandenen Projektrichtlinie aus einem Amazon Rekognition Custom Labels-Projekt zu löschen. Wenn Sie alle Revisionen einer Projektrichtlinie löschen möchten, die an ein Projekt angehängt sind, verwenden

Sie, ListProjectPoliciesum die Revisions-IDs jeder an das Projekt angehängten Projektrichtlinie abzurufen. Rufen Sie dann`DeletePolicy` nach jedem Richtliniennamen.

So löschen Sie eine Revision einer Projektrichtlinie (SDK)

1. Falls noch nicht getan haben, installieren und konfigurieren Sie dieAWS CLIAWS SDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
2. Verwenden Sie den folgenden Code, um eine Projektrichtlinie zu löschen.

   DeletePolicy dauert`ProjectARN`,`PolicyName` und`PolicyRevisionId`. `ProjectARN`und`PolicyName` sind für diese API erforderlich. `PolicyRevisionId`ist optional, kann aber für atomare Updates hinzugefügt werden.

   AWS CLI

   Ändern Sie die folgenden Werte:

   - `policy-name`zum Namen der Projektrichtlinie, die Sie löschen möchten.
   - `policy-revision-id`zur Revisions-ID der Projektrichtlinie, die Sie löschen möchten.
   - `project-arn`zum Amazon-Ressourcennamen des Projekts, das die Version der Projektrichtlinie enthält, die Sie löschen möchten.

   ```
   aws rekognition delete-project-policy \
       --policy-name policy-name \
       --policy-revision-id policy-revision-id \
       --project-arn project-arn \
     --profile custom-labels-access
   ```

   Python

   Verwenden Sie auch Sie folgenden Code Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

   - `policy-name`— Der Name der Projektrichtlinie, die Sie löschen möchten.
   - `project-arn`— Der Amazon-Ressourcenname des Projekts, das die Projektrichtlinie enthält, die Sie löschen möchten.
   - `policy-revision-id`— Die Revisions-ID der Projektrichtlinie, die Sie löschen möchten.

Zum Beispiel:`python delete_project_policy.py` *`policy_name project_arn policy_revision_id`*

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Amazon Rekognition Custom Labels model example used in the service
 documentation:
https://docs.aws.amazon.com/rekognition/latest/customlabels-dg/md-copy-model-
sdk.html
Shows how to delete a revision of a project policy.
"""

import argparse
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def delete_project_policy(rekognition_client, policy_name,  project_arn,
 policy_revision_id=None):
    """
    Deletes a project policy.

    :param rekognition_client: A Boto3 Amazon Rekognition client.
    :param policy_name: The name of the project policy that you want to delete.
    :param policy_revision_id: The revsion ID for the project policy that you
 want to delete.
    :param project_arn: The Amazon Resource Name of the project that contains
 the project policy
    that you want to delete.
    """
    try:
        logger.info("Deleting project policy: %s", policy_name)

        if policy_revision_id is None:
            rekognition_client.delete_project_policy(
```

```
                PolicyName=policy_name,
                ProjectArn=project_arn)

        else:
            rekognition_client.delete_project_policy(
                PolicyName=policy_name,
                PolicyRevisionId=policy_revision_id,
                ProjectArn=project_arn)

        logger.info("Deleted project policy: %s", policy_name)
    except ClientError:
        logger.exception("Couldn't delete project policy.")
        raise


def confirm_project_policy_deletion(policy_name):
    """
    Confirms deletion of the project policy. Returns True if delete entered.
    :param model_arn: The ARN of the model that you want to delete.
    """
    print(
        f"Are you sure you wany to delete project policy {policy_name} ?\n",
 policy_name)

    delete = input("Enter delete to delete your project policy: ")
    if delete == "delete":
        return True
    else:
        return False


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "policy_name", help="The ARN of the project that contains the project
 policy that you want to delete."
    )

    parser.add_argument(
```

```
        "project_arn", help="The ARN of the project project policy you want to
 delete."
    )

    parser.add_argument(
        "--policy_revision_id", help="(Optional) The revision ID of the project
 policy that you want to delete.",
        required=False
    )


def main():

    logging.basicConfig(level=logging.INFO,
                        format="%(levelname)s: %(message)s")

    try:

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        if confirm_project_policy_deletion(args.policy_name) is True:
            print(f"Deleting project_policy: {args.policy_name}")

            session = boto3.Session(profile_name='custom-labels-access')
            rekognition_client = session.client("rekognition")

            # Delete the project policy.

            delete_project_policy(rekognition_client,
                                  args.policy_name,
                                  args.project_arn,
                                  args.policy_revision_id)

            print(f"Finished deleting project policy: {args.policy_name}")
        else:
            print(f"Not deleting project policy {args.policy_name}")
    except ClientError as err:
        print(f"Couldn't delete project policy in {args.policy_name}: {err}")
```

```
if __name__ == "__main__":
    main()
```

Java V2

Verwenden Sie auch Sie folgenden Code Sie folgenden Code. Geben Sie die folgenden Befehlszeilenparameter an:

- `policy-name`— Der Name der Projektrichtlinie, die Sie löschen möchten.
- `project-arn`— Der Amazon-Ressourcenname des Projekts, das die Projektrichtlinie enthält, die Sie löschen möchten.
- `policy-revision-id`— Die Revisions-ID der Projektrichtlinie, die Sie löschen möchten.

```
/*
   Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
   SPDX-License-Identifier: Apache-2.0
*/

package com.example.rekognition;

import java.util.logging.Level;
import java.util.logging.Logger;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.rekognition.RekognitionClient;
import
 software.amazon.awssdk.services.rekognition.model.DeleteProjectPolicyRequest;

import software.amazon.awssdk.services.rekognition.model.RekognitionException;

public class DeleteProjectPolicy {

    public static final Logger logger =
 Logger.getLogger(DeleteProjectPolicy.class.getName());

    public static void deleteMyProjectPolicy(RekognitionClient rekClient, String
 projectArn,
            String projectPolicyName,
            String projectPolicyRevisionId)
            throws InterruptedException {
```

```
       try {
           String[] logArguments = new String[] { projectPolicyName,
projectPolicyRevisionId };

           logger.log(Level.INFO, "Deleting: Project policy: {0} revision:
{1}", logArguments);

           // Delete the project policy.

           DeleteProjectPolicyRequest deleteProjectPolicyRequest =
DeleteProjectPolicyRequest.builder()
                   .policyName(projectPolicyName)
                   .policyRevisionId(projectPolicyRevisionId)
                   .projectArn(projectArn).build();

           rekClient.deleteProjectPolicy(deleteProjectPolicyRequest);

           logger.log(Level.INFO, "Deleted: Project policy: {0} revision: {1}",
logArguments);

       } catch (

       RekognitionException e) {
           logger.log(Level.SEVERE, "Client error occurred: {0}",
e.getMessage());
           throw e;
       }

   }

   public static void main(String args[]) {

       final String USAGE = "\n" + "Usage: " + "<project_arn>
<project_policy_name> <project_policy_revision_id>\n\n"
               + "Where:\n"
               + "   project_arn - The ARN of the project that has the project
policy that you want to delete.\n\n"
               + "   project_policy_name - The name of the project policy that
you want to delete.\n\n"
               + "   project_policy_revision_id - The revision of the project
policy that you want to delete.\n\n";

       if (args.length != 3) {
```

```
            System.out.println(USAGE);
            System.exit(1);
        }

        String projectArn = args[0];
        String projectPolicyName = args[1];
        String projectPolicyRevisionId = args[2];

        try {

            RekognitionClient rekClient = RekognitionClient.builder()
            .credentialsProvider(ProfileCredentialsProvider.create("custom-
labels-access"))
            .region(Region.US_WEST_2)
            .build();

            // Delete the project policy.
            deleteMyProjectPolicy(rekClient, projectArn, projectPolicyName,
 projectPolicyRevisionId);

            System.out.println(String.format("project policy deleted: %s
 revision: %s", projectPolicyName,
                    projectPolicyRevisionId));

            rekClient.close();

        } catch (RekognitionException rekError) {
            logger.log(Level.SEVERE, "Rekognition client error: {0}",
 rekError.getMessage());
            System.exit(1);
        }

        catch (InterruptedException intError) {
            logger.log(Level.SEVERE, "Exception while sleeping: {0}",
 intError.getMessage());
            System.exit(1);
        }

    }

}
```

Beispiele

Dieser Abschnitt enthält Informationen zu Beispielen, die Sie mit Amazon Rekognition Custom Labels verwenden können.

Beispiel	Beschreibung
Feedback-Lösung modellieren	Zeigt, wie ein Modell mithilfe menschlicher Überprüfung verbessert werden kann, um einen neuen Trainingsdatensatz zu erstellen.
Vorführung von Amazon Rekognition Custom Labels	Demonstration einer Benutzeroberfläche , die die Ergebnisse eines Anrufs von anzeigt`DetectCustomLabels` .
Videoanalyse	Zeigt, wie Sie es verwenden können`DetectCustomLabels` mit Bildern, die aus einem Video extrahiert wurden.
Analysieren von Bildern mit einemAWS Lambdawirken	Zeigt, wie Sie es verwenden können`DetectCustomLabels` mit einer Lambda-Funktion.
Erstellen einer Manifestdatei aus einer CSV-Datei	Zeigt, wie eine CSV-Datei verwendet wird, um eine Manifestdatei zu erstellen, die zum Suchen geeignet istObjekte, Szenen und Konzeptemit einem ganzen Bild verknüpft (Klassifizierung).

Feedback-Lösung modellieren

Die Model Feedback-Lösung ermöglicht es Ihnen, Feedback zu den Vorhersagen Ihres Modells zu geben und Verbesserungen vorzunehmen, indem Sie die Überprüfung durch einen Menschen durchführen. Je nach Anwendungsfall können Sie mit einem Trainingsdatensatz, der nur wenige Bilder enthält, erfolgreich sein. Möglicherweise ist ein größerer Trainingssatz mit Anmerkungen

erforderlich, um ein genaueres Modell zu erstellen. Mit der Model Feedback-Lösung können Sie mithilfe der Modellunterstützung einen größeren Datensatz erstellen.

Informationen zur Installation und Konfiguration der Model Feedback-Lösung finden Sie unterFeedback-Lösung modellieren.

Der Arbeitsablauf für die kontinuierliche Modellverbesserung sieht wie folgt aus:

1. Trainieren Sie die erste Version Ihres Modells (möglicherweise mit einem kleinen Trainingsdatensatz).
2. Stellen Sie einen Datensatz ohne Anmerkungen für die Model Feedback-Lösung bereit.
3. Die Model Feedback-Lösung verwendet das aktuelle Modell. Es startet menschliche Überprüfungsaufträge, um einen neuen Datensatz mit Anmerkungen zu versehen.
4. Basierend auf menschlichem Feedback generiert die Model Feedback-Lösung eine Manifestdatei, mit der Sie ein neues Modell erstellen.

Vorführung von Amazon Rekognition Custom Labels

Die Amazon Rekognition Custom Labels-Demonstration zeigt eine Benutzeroberfläche, die Bilder von Ihrem lokalen Computer analysiert, indem sie dieDetectCustomLabelsAPI.

Die Anwendung zeigt Ihnen Informationen zu den Amazon Rekognition Custom Labels-Modellen in IhremAWSKonto. Nachdem Sie ein laufendes Modell ausgewählt haben, können Sie ein Bild von Ihrem lokalen Computer analysieren. Bei Bedarf können Sie ein Modell starten. Sie können ein laufendes Modell auch stoppen. Die Anwendung zeigt die Integration mit anderen AWS-Services wie Amazon Cognito, Amazon S3 und AmazonCloudFront.

Weitere Informationen finden Sie unterAmazon Rekognition — Demo für benutzerdefinierte Etiketten.

Videoanalyse

Das folgende Beispiel zeigt, wie Sie`DetectCustomLabels`mit Bildern, die aus einem Video extrahiert wurden. Der Code wurde mit Videodateien in getestetbewegenundmp4formatieren.

Verwenden**`DetectCustomLabels`**mit aufgenommenen Bildern

1. Falls Sie dies noch nicht getan haben, installieren und konfigurieren Sie denAWS CLIund derAWSSDKs. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.

2. Stellen Sie sicher, dass Sie`rekognition:DetectCustomLabels`und`AmazonS3ReadOnlyAccess`Berechtigungen. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.

3. Verwenden Sie den folgenden Beispielcode. Ändern Sie den Wert von`videoFile`auf den Namen einer Videodatei. Ändern Sie den Wert von`projectVersionArn`zum Amazon Resource Name (ARN) Ihres Amazon Rekognition Custom Labels-Modells.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Shows how to analyze a local video with an Amazon Rekognition Custom Labels model.
"""
import argparse
import logging
import json
import math
import cv2
import boto3

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def analyze_video(rek_client, project_version_arn, video_file):
    """
    Analyzes a local video file with an Amazon Rekognition Custom Labels model.
    Creates a results JSON file based on the name of the supplied video file.
    :param rek_client: A Boto3 Amazon Rekognition client.
    :param project_version_arn: The ARN of the Custom Labels model that you want to
 use.
    :param video_file: The video file that you want to analyze.
    """

    custom_labels = []
    cap = cv2.VideoCapture(video_file)
    frame_rate = cap.get(5)  # Frame rate.
    while cap.isOpened():
        frame_id = cap.get(1)  # Current frame number.
        print(f"Processing frame id: {frame_id}")
```

```
        ret, frame = cap.read()
        if ret is not True:
            break
        if frame_id % math.floor(frame_rate) == 0:
            has_frame, image_bytes = cv2.imencode(".jpg", frame)

            if has_frame:
                response = rek_client.detect_custom_labels(
                    Image={
                        'Bytes': image_bytes.tobytes(),
                    },
                    ProjectVersionArn=project_version_arn
                )

            for elabel in response["CustomLabels"]:
                elabel["Timestamp"] = (frame_id/frame_rate)*1000
                custom_labels.append(elabel)

    print(custom_labels)

    with open(video_file + ".json", "w", encoding="utf-8") as f:
        f.write(json.dumps(custom_labels))

    cap.release()


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "project_version_arn", help="The ARN of the model that you want to use."
    )

    parser.add_argument(
        "video_file", help="The local path to the video that you want to analyze."
    )


def main():

    logging.basicConfig(level=logging.INFO,
```

```
                        format="%(levelname)s: %(message)s")

    try:
        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        session = boto3.Session(profile_name='custom-labels-access')
        rekognition_client = session.client("rekognition")

        analyze_video(rekognition_client,
                      args.project_version_arn, args.video_file)

    except ClientError as err:
        print(f"Couldn't analyze video: {err}")


if __name__ == "__main__":
    main()
```

Analysieren von Bildern mit einemAWS Lambdawirken

AWS Lambda ist ein Datenverarbeitungsservice, mit dem Sie Code ausführen können, ohne Server bereitstellen oder verwalten zu müssen. Sie können beispielsweise Bilder analysieren, die über eine mobile Anwendung übermittelt wurden, ohne einen Server erstellen zu müssen, der den Anwendungscode hostet. Die folgende Anleitung zeigt, wie Sie in Python eine Lambda-Funktion erstellen, dieDetectCustomLabels. Die Funktion analysiert ein bereitgestelltes Bild und gibt eine Liste der im Bild gefundenen Labels zurück. Die Anweisungen enthalten einen Python-Beispielcode, der zeigt, wie die Lambda-Funktion mit einem Bild in einem Amazon S3-Bucket oder einem von einem lokalen Computer bereitgestellten Bild aufgerufen wird.

Themen

- Schritt 1: Erstellen Sie eineAWS LambdaFunktion (Konsole)
- Schritt 2: (Optional) Erstellen Sie eine Ebene (Konsole)
- Schritt 3: Python-Code hinzufügen (Konsole)
- Schritt 4: Testen Sie Ihre Lambda-Funktion

Schritt 1: Erstellen Sie eineAWS LambdaFunktion (Konsole)

In diesem Schritt erstellen Sie ein leeresAWSFunktion und eine IAM-Ausführungsrolle, die es Ihrer Funktion ermöglicht, die`DetectCustomLabels`Betrieb. Es gewährt auch Zugriff auf den Amazon S3-Bucket, in dem Bilder zur Analyse gespeichert werden. Sie geben auch Umgebungsvariablen für Folgendes an:

- Das Amazon Rekognition Custom Labels-Modell, das Ihre Lambda-Funktion verwenden soll.
- Die Konfidenzgrenze, die das Modell verwenden soll.

Später fügen Sie der Lambda-Funktion den Quellcode und optional eine Ebene hinzu.

Um eine zu erstellenAWS LambdaFunktion (Konsole)

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Lambda-Konsole an https://console.aws.amazon.com/lambda.
2. Wählen Sie Funktion erstellen aus. Weitere Informationen finden Sie unterErstellen Sie eine Lambda-Funktion mit der Konsole.
3. Wählen Sie die folgenden Optionen.
   - Wählen Sie Ohne Vorgabe erstellen aus.
   - Geben Sie einen Wert ein fürName der Funktion.
   - FürLaufzeitwählenPython 3.10.
4. Wählen Sie Create function, um die AWS Lambda-Funktion zu erstellen.
5. Wählen Sie auf der Funktionsseite dieKonfigurationTab.
6. Auf derUmgebungsvariablenFenster, wählenBearbeiten.
7. Fügen Sie die folgenden Umgebungsvariablen hinzu. Wählen Sie für jede VariableUmgebungsvariable hinzufügenund geben Sie dann den Variablenschlüssel und den Wert ein.

Schlüssel	Value (Wert)
MODELL_ARN	Der Amazon-Ressourcenname (ARN) des Modells, das Ihre Lambda-Funktion verwenden soll. Das Modell ARN erhalten Sie von derModell verwendenRegisterk

Schlüssel	Value (Wert)
	arte auf der Detailseite des Modells in der Amazon Rekognition Custom Labels-Konsole.
VERTRAUEN	Der Mindestwert (0—100) für das Vertrauen des Modells in die Vorhersage für ein Label. Die Lambda-Funktion gibt keine Labels mit Konfidenzwerten zurück, die unter diesem Wert liegen.

8. Wählen SieSpeichernum die Umgebungsvariablen zu speichern.
9. Auf derBerechtigungenFenster, UnterName der Rolle, wählen Sie die Ausführungsrolle aus, um die Rolle in der IAM-Konsole zu öffnen.
10. In derBerechtigungenTab, wähleBerechtigungen hinzufügenund dannInline-Richtlinie erstellen.
11. Wählen SieJSONund ersetzen Sie die bestehende Richtlinie durch die folgende Richtlinie.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "rekognition:DetectCustomLabels",
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DetectCustomLabels"
        }
    ]
}
```

12. Wählen Sie Weiter aus.
13. InEinzelheiten der Richtlinie, geben Sie einen Namen für die Richtlinie ein, z. B.DetectCustomLabels-Zugang.
14. Wählen Sie Create Policy (Richtlinie erstellen) aus.
15. Wenn Sie Bilder zur Analyse in einem Amazon S3-Bucket speichern, wiederholen Sie die Schritte 10—14.

    a. Verwenden Sie für Schritt 11 die folgende Richtlinie. Ersetzen*Bucket-/Ordnerpfad*mit dem Amazon S3-Bucket und dem Ordnerpfad zu den Bildern, die Sie analysieren möchten.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3Access",
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::bucket/folder path/*"
        }
    ]
}
```

b. Wählen Sie für Schritt 13 einen anderen Richtliniennamen, z. B.Zugriff auf den S3 Bucket.

Schritt 2: (Optional) Erstellen Sie eine Ebene (Konsole)

Um dieses Beispiel auszuführen, müssen Sie diesen Schritt nicht ausführen. Der`DetectCustomLabels`Die Operation ist in der Lambda-Python-Standardumgebung als Teil von enthaltenAWSSDK für Python (Boto3). Wenn andere Teile Ihrer Lambda-Funktion aktuell sein müssenAWSService-Updates, die nicht in der Lambda-Python-Standardumgebung enthalten sind, führen Sie diesen Schritt aus, um Ihrer Funktion die neueste Boto3-SDK-Version als Ebene hinzuzufügen.

Zunächst erstellen Sie ein ZIP-Dateiarchiv, das das Boto3-SDK enthält. Anschließend erstellen Sie eine Ebene und fügen der Ebene das ZIP-Dateiarchiv hinzu. Weitere Informationen finden Sie unterVerwenden von Ebenen mit Ihrer Lambda-Funktion.

Um eine Ebene zu erstellen und hinzuzufügen (Konsole)

1. Öffnen Sie eine Befehlszeile und geben Sie die folgenden Befehle ein.

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. Notieren Sie sich den Namen der Zip-Datei (boto3-layer.zip). Sie benötigen es in Schritt 6 dieses Verfahrens.
3. Öffnen Sie die AWS Lambda-Konsole unter https://console.aws.amazon.com/lambda/.
4. Wählen Sie im Navigationsbereich Layers aus.

5. Wählen Sie Create Layer (Ebene erstellen) aus.
6. Geben Sie einen Name (Namen) und eine Description (Beschreibung) ein.
7. Wählen SieLaden Sie eine ZIP-Datei hochund wähleUpload.
8. Wählen Sie im Dialogfeld das ZIP-Dateiarchiv (boto3-layer.zip) aus, das Sie in Schritt 1 dieses Verfahrens erstellt haben.
9. Wählen Sie für kompatible LaufzeitenPython 3.9.
10. Wählen SieErstellenum die Ebene zu erstellen.
11. Wählen Sie das Menüsymbol im Navigationsbereich.
12. Wählen Sie im Navigationsbereich Functions aus.
13. Wählen Sie in der Ressourcenliste die Funktion aus, die Sie in erstellt habenSchritt 1: Erstellen Sie eineAWS LambdaFunktion (Konsole).
14. Wählen Sie die Registerkarte Code (Code).
15. In derLagenAbschnitt, wählenEine Ebene hinzufügen.
16. Wählen SieBenutzerdefinierte Ebenen.
17. InBenutzerdefinierte Ebenen, wählen Sie den Layer-Namen, den Sie in Schritt 6 eingegeben haben.
18. InVersionwählen Sie die Layer-Version, die 1 sein sollte.
19. Wählen Sie Add (Hinzufügen) aus.

Schritt 3: Python-Code hinzufügen (Konsole)

In diesem Schritt fügen Sie Ihrer Lambda-Funktion Python-Code hinzu, indem Sie den Code-Editor der Lambda-Konsole verwenden. Der Code analysiert ein bereitgestelltes Bild mit`DetectCustomLabels`und gibt eine Liste der im Bild gefundenen Labels zurück. Das bereitgestellte Bild kann sich in einem Amazon S3-Bucket befinden oder als byte64-kodierte Bildbytes bereitgestellt werden.

Um Python-Code hinzuzufügen (Konsole)

1. Wenn Sie nicht in der Lambda-Konsole sind, gehen Sie wie folgt vor:
   a. Öffnen Sie die AWS Lambda-Konsole unter https://console.aws.amazon.com/lambda/.
   b. Öffnen Sie die Lambda-Funktion, die Sie in erstellt habenSchritt 1: Erstellen Sie eineAWS LambdaFunktion (Konsole).

2. Wählen Sie die Registerkarte Code (Code).
3. InQuellcode, ersetze den Code inlambda_function.pymit den folgenden:

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
An AWS lambda function that analyzes images with an the Amazon Rekognition
Custom Labels model.
"""
import json
import base64
from os import environ
import logging
import boto3

from botocore.exceptions import ClientError

# Set up logging.
logger = logging.getLogger(__name__)

# Get the model ARN and confidence.
model_arn = environ['MODEL_ARN']
min_confidence = int(environ.get('CONFIDENCE', 50))

# Get the boto3 client.
rek_client = boto3.client('rekognition')


def lambda_handler(event, context):
    """
    Lambda handler function
    param: event: The event object for the Lambda function.
    param: context: The context object for the lambda function.
    return: The labels found in the image passed in the event
    object.
    """

    try:

        # Determine image source.
        if 'image' in event:
```

```
            # Decode the image
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
            image = {'Bytes': img_b64decoded}


        elif 'S3Object' in event:
            image = {'S3Object':
                     {'Bucket':  event['S3Object']['Bucket'],
                      'Name': event['S3Object']['Name']}
                     }

        else:
            raise ValueError(
                'Invalid source. Only image base 64 encoded image bytes or S3Object
are supported.')


        # Analyze the image.
        response = rek_client.detect_custom_labels(Image=image,
            MinConfidence=min_confidence,
            ProjectVersionArn=model_arn)

        # Get the custom labels
        labels = response['CustomLabels']

        lambda_response = {
            "statusCode": 200,
            "body": json.dumps(labels)
        }

    except ClientError as err:
        error_message = f"Couldn't analyze image. " + \
            err.response['Error']['Message']

        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": err.response['Error']['Code'],
                "ErrorMessage": error_message
            }
        }
        logger.error("Error function %s: %s",
            context.invoked_function_arn, error_message)
```

```
    except ValueError as val_error:
        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": "ValueError",
                "ErrorMessage": format(val_error)
            }
        }
        logger.error("Error function %s: %s",
            context.invoked_function_arn, format(val_error))

    return lambda_response
```

4. Wählen SieBereitstellenum Ihre Lambda-Funktion bereitzustellen.

Schritt 4: Testen Sie Ihre Lambda-Funktion

In diesem Schritt verwenden Sie Python-Code auf Ihrem Computer, um ein lokales Bild oder ein Bild in einem Amazon S3-Bucket an Ihre Lambda-Funktion zu übergeben. Bilder, die von einem lokalen Computer übertragen werden, müssen kleiner als 6291456 Byte sein. Wenn Ihre Bilder größer sind, laden Sie die Bilder in einen Amazon S3-Bucket hoch und rufen Sie das Skript mit dem Amazon S3-Pfad zum Image auf. Informationen zum Hochladen von Bilddateien in einen Amazon S3-Bucket finden Sie unterObjekte hochladen.

Stellen Sie sicher, dass Sie den Code im selben ausführenAWSRegion, in der Sie die Lambda-Funktion erstellt haben. Sie können sich die ansehenAWSRegion für Ihre Lambda-Funktion in der Navigationsleiste der Seite mit den Funktionsdetails in derLambda-Konsole.

Wenn derAWS LambdaDie Funktion gibt einen Timeout-Fehler zurück, verlängern Sie den Timeout-Zeitraum für die Lambda-Funktionsfunktion. Weitere Informationen finden Sie unterFunktions-Timeout konfigurieren (Konsole).

Weitere Hinweise zum Aufrufen einer Lambda-Funktion aus Ihrem Code finden Sie unterAufrufenAWS LambdaFunktionen.

Um Ihre Lambda-Funktion auszuprobieren

1. Stellen Sie sicher, dass Sie`lambda:InvokeFunction`Erlaubnis. Sie können die folgende Richtlinie verwenden.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "InvokeLambda",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "ARN for lambda function"
        }
    ]
}
```

Den ARN für Ihre Lambda-Funktionsfunktion finden Sie in der Funktionsübersicht in derLambda-Konsole.

Um Zugriff zu gewähren, fügen Sie Ihren Benutzern, Gruppen oder Rollen Berechtigungen hinzu:

- Benutzer und Gruppen in AWS IAM Identity Center:

  Erstellen Sie einen Berechtigungssatz. Befolgen Sie die Anweisungen unter Erstellen eines Berechtigungssatzes im AWS IAM Identity Center-Benutzerhandbuch.

- Benutzer, die in IAM über einen Identitätsanbieter verwaltet werden:

  Erstellen Sie eine Rolle für den Identitätsverbund. Befolgen Sie die Anweisungen unter Erstellen einer Rolle für einen externen Identitätsanbieter (Verbund) im IAM-Benutzerhandbuch.

- IAM-Benutzer:
  - Erstellen Sie eine Rolle, die Ihr Benutzer annehmen kann. Folgen Sie den Anweisungen unter Erstellen einer Rolle für einen IAM-Benutzer im IAM-Benutzerhandbuch.
  - (Nicht empfohlen) Weisen Sie einem Benutzer eine Richtlinie direkt zu oder fügen Sie einen Benutzer zu einer Benutzergruppe hinzu. Befolgen Sie die Anweisungen unter Hinzufügen von Berechtigungen zu einem Benutzer (Konsole) im IAM-Benutzerhandbuch.

2. Installieren und konfigurierenAWSSDK für Python. Weitere Informationen finden Sie unter Schritt 4: Richten Sie die AWS CLI und SDKs ein AWS.
3. Starten Sie das Modelldie Sie in Schritt 7 von angegeben habenSchritt 1: Erstellen Sie eineAWS LambdaFunktion (Konsole).
4. Speichern Sie den folgenden Code in einer Datei mit dem Namen`client.py`.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0

"""
Purpose
Test code for running the Amazon Rekognition Custom Labels Lambda
function example code.
"""

import argparse
import logging
import base64
import json
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


def analyze_image(function_name, image):
    """Analyzes an image with an AWS Lambda function.
    :param image: The image that you want to analyze.
    :return The status and classification result for
    the image analysis.
    """

    lambda_client = boto3.client('lambda')

    lambda_payload = {}

    if image.startswith('s3://'):
        logger.info("Analyzing image from S3 bucket: %s", image)
        bucket, key = image.replace("s3://", "").split("/", 1)
        s3_object = {
            'Bucket': bucket,
            'Name': key
        }
        lambda_payload = {"S3Object": s3_object}

    # Call the lambda function with the image.
    else:
```

```
        with open(image, 'rb') as image_file:
            logger.info("Analyzing local image image: %s ", image)
            image_bytes = image_file.read()
            data = base64.b64encode(image_bytes).decode("utf8")

            lambda_payload = {"image": data}

    response = lambda_client.invoke(FunctionName=function_name,
                                    Payload=json.dumps(lambda_payload))

    return json.loads(response['Payload'].read().decode())


def add_arguments(parser):
    """
    Adds command line arguments to the parser.
    :param parser: The command line parser.
    """

    parser.add_argument(
        "function", help="The name of the AWS Lambda function that you want " \
        "to use to analyze the image.")
    parser.add_argument(
        "image", help="The local image that you want to analyze.")


def main():
    """
    Entrypoint for script.
    """
    try:
        logging.basicConfig(level=logging.INFO,
                            format="%(levelname)s: %(message)s")

        # Get command line arguments.
        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
        add_arguments(parser)
        args = parser.parse_args()

        # Get analysis results.
        result = analyze_image(args.function, args.image)
        status = result['statusCode']

        if status == 200:
```

```
            labels = result['body']
            labels = json.loads(labels)
            print(f"There are {len(labels)} labels in the image.")
            for custom_label in labels:
                confidence = int(round(custom_label['Confidence'], 0))
                print(
                    f"Label: {custom_label['Name']}: Confidence: {confidence}%")
        else:
            print(f"Error: {result['statusCode']}")
            print(f"Message: {result['body']}")

    except ClientError as error:
        logging.error(error)
        print(error)


if __name__ == "__main__":
    main()
```

5. Führen Sie den Code aus. Geben Sie für das Befehlszeilenargument den Namen der Lambda-Funktion und das Bild an, das Sie analysieren möchten. Sie können einen Pfad zu einem lokalen Image oder den S3-Pfad zu einem in einem Amazon S3-Bucket gespeicherten Image angeben. Beispiele:

   ```
   python client.py function_name s3://bucket/path/image.jpg
   ```

   Wenn sich das Image in einem Amazon S3-Bucket befindet, stellen Sie sicher, dass es sich um denselben Bucket handelt, den Sie in Schritt 15 von angegeben habenSchritt 1: Erstellen Sie eineAWS LambdaFunktion (Konsole).

   Wenn dies erfolgreich ist, ist die Ausgabe eine Liste der im Bild gefundenen Labels. Wenn keine Labels zurückgegeben werden, sollten Sie erwägen, den Konfidenzwert zu senken, den Sie in Schritt 7 von festgelegt habenSchritt 1: Erstellen Sie eineAWS LambdaFunktion (Konsole).
6. Wenn Sie mit der Lambda-Funktion fertig sind und das Modell nicht von anderen Anwendungen verwendet wird,Stoppe das Modell. Denken Sie daranstarte das Modellwenn Sie das nächste Mal die Lambda-Funktion verwenden möchten.

Sicherheit

Sie können die Verwaltung Ihrer Projekte, Modelle und der`DetectCustomLabels`Vorgang, den Ihre Kunden verwenden, um benutzerdefinierte Etiketten zu erkennen.

Weitere Informationen zur Absicherung von Amazon Rekognition finden Sie unterSicherheit bei Amazon Rekognition.

Sicherung von Amazon Rekognition Custom Labels-Projekten

Sie können Ihre Amazon Rekognition Custom Labels-Projekte schützen, indem Sie die Berechtigungen auf Ressourcenebene angeben, die in identitätsbasierten Richtlinien festgelegt sind. Weitere Informationen finden Sie unter Identitätsbasierte Richtlinien und ressourcenbasierte Richtlinien.

Die Amazon Rekognition Custom Labels-Ressourcen, die Sie sichern können, sind:

Ressource	Format des Amazon-Ressourcennamens
Projekt	arn:aws:rekognition: *:*:project/*projektname* /datum/uhrzeit
Model	arn:aws:rekognition: *:*:project/*projektname* /version/*Name*/datum/uhrzeit

Die folgende Beispielrichtlinie zeigt, wie eine Identitätsberechtigung erteilt wird:

- Beschreiben Sie alle Projekte.
- Erstellen, starten, beenden und verwenden Sie ein bestimmtes Modell für Inferenz.
- Erstellen eines Projekts Erstellen und beschreiben Sie ein bestimmtes Modell.
- Verweigern Sie die Erstellung eines bestimmten Projekts.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AllResources",
            "Effect": "Allow",
            "Action": "rekognition:DescribeProjects",
            "Resource": "*"
        },
        {
            "Sid": "SpecificProjectVersion",
            "Effect": "Allow",
            "Action": [
                "rekognition:StopProjectVersion",
                "rekognition:StartProjectVersion",
                "rekognition:DetectCustomLabels",
                "rekognition:CreateProjectVersion"
            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/version/MyVersion/*"
       },
        {
            "Sid": "SpecificProject",
            "Effect": "Allow",
            "Action": [
                "rekognition:CreateProject",
                "rekognition:DescribeProjectVersions",
                "rekognition:CreateProjectVersion"
            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/*"
        },
        {
            "Sid": "ExplicitDenyCreateProject",
            "Effect": "Deny",
            "Action": [
                "rekognition:CreateProject"
            ],
            "Resource": ["arn:aws:rekognition:*:*:project/SampleProject/*"]
        }
    ]
}
```

SicherungDetectCustomLabels

Die Identität, die zur Erkennung benutzerdefinierter Labels verwendet wird, kann sich von der Identität unterscheiden, die Amazon Rekognition Custom Labels-Modelle verwaltet.

Sie können den Zugriff einer Identität auf sichern`DetectCustomLabels`indem Sie eine Richtlinie auf die Identität anwenden. Das folgende Beispiel beschränkt den Zugriff auf`DetectCustomLabels`nur und für ein bestimmtes Modell. Die Identität hat keinen Zugriff auf die anderen Amazon Rekognition-Operationen.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "rekognition:DetectCustomLabels"


            ],
            "Resource": "arn:aws:rekognition:*:*:project/MyProject/version/MyVersion/*"
        }
    ]
}
```

Von AWS verwaltete Richtlinien

Wir bieten die`AmazonRekognitionCustomLabelsFullAccess` AWSverwaltete Richtlinie, mit der Sie den Zugriff auf Amazon Rekognition Custom Labels steuern können. Weitere Informationen finden Sie unterVon AWS verwaltete Richtlinie: AmazonRekognitionCustomLabelsFullAccess.

Richtlinien und Kontingente in Amazon Rekognition Custom Labels

Die folgenden Abschnitte enthalten Richtlinien und Kontingente für die Verwendung von Amazon Rekognition Custom Labels.

Unterstützte Regionen

Eine Liste der AWS Regionen, in denen Amazon Rekognition Custom Labels verfügbar ist, finden Sie unter -Regionen und Endpunkte in der Allgemeinen Amazon-Web-Services-Referenz.

Kontingente

Im Folgenden finden Sie eine Liste der Einschränkungen in Amazon Rekognition Custom Labels. Informationen dazu, welche Limits geändert werden können, finden Sie unter AWS Service Limits. Wenn Sie ein Limit ändern möchten, lesen Sie unter CreateCase weiter.

Schulung

- Unterstützte Dateiformate sind PNG- und JPEG-Bildformate.
- Die maximale Anzahl von Trainingsdatensätzen in einer Version eines Modells ist 1.
- Die maximale Größe der Datensatz-Manifestdatei beträgt 1 GB.
- Die Mindestanzahl eindeutiger Labels pro Datensatz mit Objekten, Szenen und Konzepten (Klassifikation) beträgt 2.
- Die Mindestanzahl eindeutiger Labels pro Datensatz zur Objektposition (Erkennung) beträgt 1.
- Die maximale Anzahl eindeutiger Labels pro Manifest beträgt 250.
- Die MindestAnzahl der der
- Die maximale Anzahl von Bildern pro Datensatz zur Objektposition (Erkennung) beträgt 250.000.

  Das Limit für die AWS Regionen Asien-Pazifik (Mumbai) und Europa (London) liegt bei 28.000 Bildern.
- Die maximale Anzahl von Bildern pro Datensatz für Objekte, Szenen und Konzepte (Klassifizierung) beträgt 500.000. Die Standardeinstellung ist 250.000. Um eine Erhöhung zu beantragen, siehe Fall erstellen.

Das Limit für die AWS Regionen Asien-Pazifik (Mumbai) und Europa (London) liegt bei 28.000 Bildern. Sie können eine Erhöhung des Limits beantragen nicht beantragen.

- Die maximale Anzahl der der Bezeichnungen zeichnungen zeichnungen
- Die Mindestanzahl von Begrenzungsfeldern in einem Bild ist 0.
- Die maximale Anzahl von Begrenzungsrahmen in einem Bild beträgt 50.
- Die Mindestbildgröße der Bilddatei in einem Amazon S3 S3-Bucket beträgt 64 Pixel x 64 Pixel.
- Die maximale Bildgröße der Bilddatei beträgt 4096 Pixel
- Die maximale Dateigröße Amazon S3 MB.
- Das maximale Bildseitenverhältnis beträgt 20:1.

Testen

- Die maximale Anzahl von Testdatensätzen in einer Version eines Modells ist 1.
- Die maximale Größe der Datensatz-Manifestdatei beträgt 1 GB.
- Die Mindestanzahl eindeutiger Labels pro Datensatz mit Objekten, Szenen und Konzepten (Klassifikation) beträgt 2.
- Die Mindestanzahl eindeutiger Labels pro Datensatz zur Objektposition (Erkennung) beträgt 1.
- Die maximale Anzahl eindeutiger Labels pro Datensatz beträgt 250.
- Die MindestAnzahl der der
- Die maximale Anzahl der
- Die maximale Anzahl von Bildern pro Datensatz zur Objektposition (Erkennung) beträgt 250.000.

  Das Limit für die AWS Regionen Asien-Pazifik (Mumbai) und Europa (London) liegt bei 7.000 Bildern.
- Die maximale Anzahl von Bildern pro Datensatz für Objekte, Szenen und Konzepte (Klassifizierung) beträgt 500.000. Die Standardeinstellung ist 250.000. Um eine Erhöhung zu beantragen, siehe Fall erstellen.

  Das Limit für die AWS Regionen Asien-Pazifik (Mumbai) und Europa (London) liegt bei 7.000 Bildern. Sie können eine Erhöhung des Limits beantragen nicht beantragen.
- Die Mindestanzahl von Labels pro Bild und Manifest ist 0.
- Die maximale Anzahl der der Bezeichnungen zeichnungen zeichnungen
- Die Mindestanzahl von Begrenzungsfeldern in einem Bild pro Manifest ist 0.

- Die maximale Anzahl von Begrenzungsfeldern in einem Bild pro Manifest beträgt 50.
- Die Mindestbildgröße einer Bilddatei in einem Amazon S3 S3-Bucket beträgt 64 Pixel x 64 Pixel.
- Die maximale Bildgröße Amazon S3 Bilddatei beträgt 4096 Pixel
- Die maximale Dateigröße Amazon S3 MB.
- Unterstützte Dateiformate sind PNG- und JPEG-Bildformate.
- Das maximale Bildseitenverhältnis beträgt 20:1.

Erkennung

- Die maximale Größe von Bildern, die als Rohbytes übergeben werden, beträgt 4 MB.
- Die maximale Dateigröße Amazon S3 MB.
- Die Mindestbildgröße einer Eingabebilddatei (gespeichert in einem Amazon S3 S3-Bucket oder als Bildbyte bereitgestellt) beträgt 64 Pixel x 64 Pixel.
- Die maximale Bildgröße einer Eingabebilddatei (in einem Amazon S3 gespeichert oder als Bildbyte bereitgestellt) beträgt 4096 Pixel x 4096 Pixel.
- Unterstützte Dateiformate sind PNG- und JPEG-Bildformate.
- Das maximale Bildseitenverhältnis beträgt 20:1.

Modell kopieren

- Die maximale Anzahl von Projektrichtlinien, die Sie einem Projekt zuordnen können, beträgt 5.
- Die maximale Anzahl der gleichzeitigen Kopieraufträge

Amazon Rekognition Custom Labels (-API)

Die Amazon Rekognition Custom Labels API ist als Teil des Amazon Rekognition API-Referenzinhalts dokumentiert. Dies ist eine Liste der Amazon Rekognition Custom Labels API-Operationen mit Links zum entsprechenden Amazon Rekognition Rekognition-API-Referenzthema. Die API-Referenzlinks in diesem Dokument führen außerdem zum entsprechenden API-Referenzthema des Amazon Rekognition Developer Guide. Informationen über die Verwendung der -API finden Sie unter

Dieser Abschnitt gibt Ihnen einen Überblick über den Arbeitsablauf zum Trainieren und Verwenden eines Amazon Rekognition Custom Labels-Modells mit der Konsole und dem AWS SDK.

> **Note**
>
> Amazon Rekognition Custom Labels verwaltet jetzt Datensätze innerhalb eines Projekts. Sie können Datensätze für Ihre Projekte mit der Konsole und dem SDK erstellen. AWS

Wenn Sie zuvor Amazon Rekognition Custom Labels verwendet haben, müssen Ihre älteren

Datensätze möglicherweise einem neuen Projekt zugeordnet werden. Weitere Informationen finden Sie unter Schritt 6: (Optional) Zuordnen früherer Datensätze zu mit neuen Projekten.

Themen

Ihren Modelltyp festlegen

Sie entscheiden zunächst, welche Art von Modell Sie trainieren möchten, was von Ihren Geschäftszielen abhängt. Sie könnten einem Modell beispielsweise beibringen, Ihr Logo in Social-Media-Posts zu finden, Ihre Produkte in den Verkaufsregalen zu identifizieren oder Maschinenteile auf einem Fließband zu klassifizieren.

Amazon Rekognition Custom Labels kann die folgenden Modelltypen trainieren:

- Objekte, Szenen und Konzepte finden

- Nach Objektpositionen suchen
- Position von Marken finden

Um Ihnen bei der Entscheidung zu helfen, welche Art von Modell Sie trainieren möchten, bietet Amazon Rekognition Custom Labels Beispielprojekte, die Sie verwenden können. Weitere Informationen finden Sie unter Erste Schritte mit Amazon Rekognition Custom Labels.

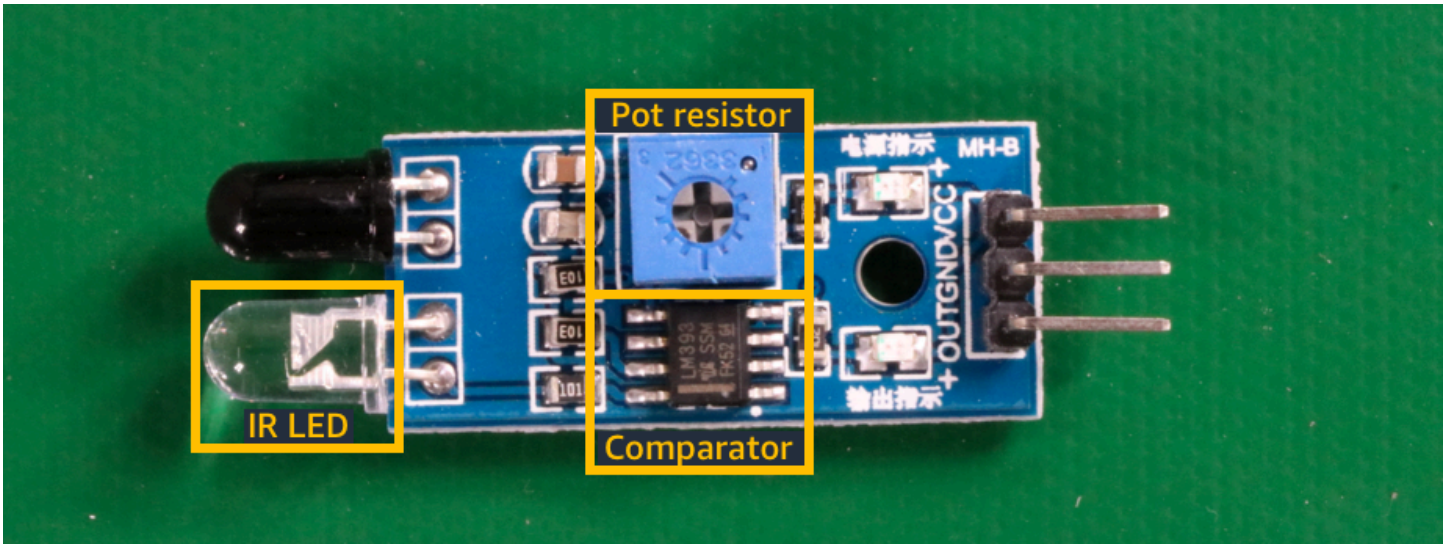
Objekte, Szenen und Konzepte finden

Das Modell sagt Klassifizierungen für die Objekte, Szenen und Konzepte voraus, die einem gesamten Bild zugeordnet sind. Sie können beispielsweise ein Modell trainieren, das bestimmt, ob ein Bild eine Touristenattraktion enthält oder nicht. Ein Beispielobjekt finden Sie unter Bildklassifizierung. Das folgende Bild eines Sees ist ein Beispiel für die Art von Bild, in dem Sie Objekte, Szenen und Konzepte erkennen können.

Alternativ können Sie ein Modell trainieren, das Bilder in mehrere Kategorien einteilt. Das vorherige Bild könnte beispielsweise Kategorien wie Himmelsfarbe, Spiegelung oder See enthalten. Ein Beispielobjekt finden Sie unter Bildklassifizierung (mehrere Label).

Nach Objektpositionen suchen

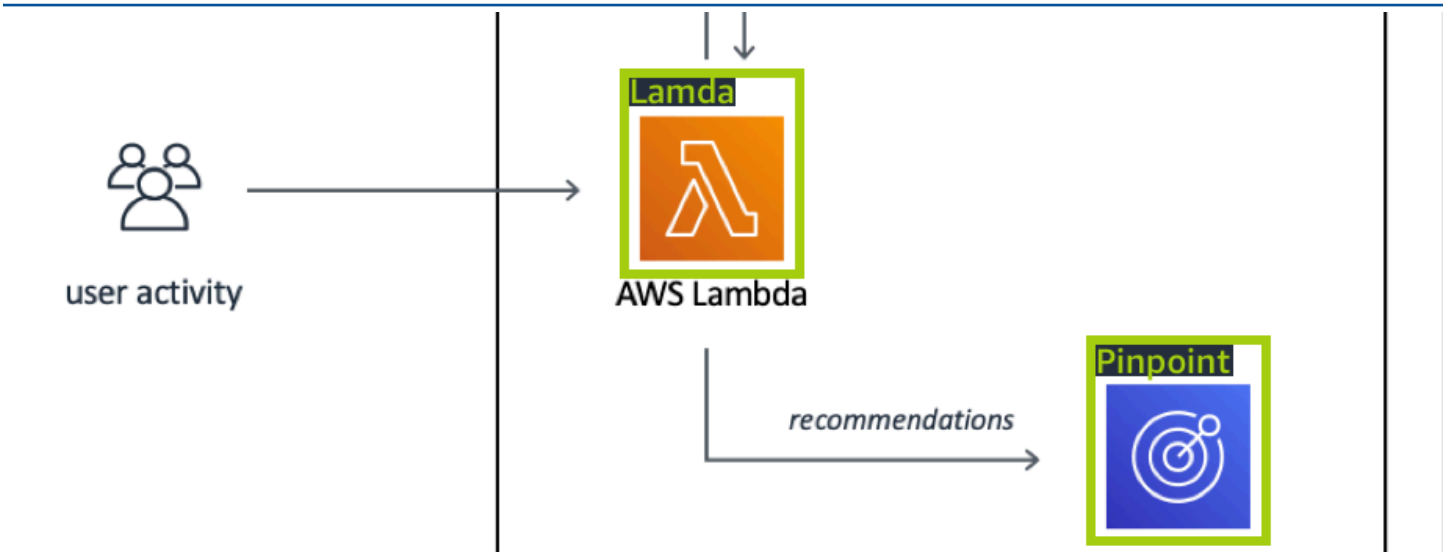
Das Modell sagt die Position eines Objekts auf einem Bild voraus. Die Vorhersage umfasst Begrenzungsrahmen-Informationen für die Objektposition und ein Label, das das Objekt innerhalb des Begrenzungsrahmens identifiziert. Die folgende Abbildung zeigt beispielsweise Begrenzungsrahmen, die verschiedene Teile einer Leiterplatte umgeben, z. B. einen Komparator oder einen Pot-Widerstand.



Das Lokalisierung von Objekten-Beispielprojekt zeigt, wie Amazon Rekognition Custom Labels mit Labeln versehene Begrenzungsrahmen verwendet, um ein Modell zu trainieren, das Objektpositionen findet.

Position von Marken finden

Mit Amazon Rekognition Custom Labels kann ein Modell trainiert werden, das die Position von Marken, z. B. Logos, auf einem Bild ermittelt. Die Vorhersage umfasst Informationen zum Begrenzungsrahmen für die Position der Marke und ein Label, das das Objekt innerhalb des Begrenzungsrahmens identifiziert. Ein Beispielobjekt finden Sie unter Erkennung von Marken. Die folgende Abbildung zeigt ein Beispiel für einige Marken, die das Modell erkennen kann.



Erstellen eines Modells

Die Schritte zum Erstellen eines Modells umfassen das Erstellen eines Projekts, das Erstellen von Trainings- und Testdatensätzen und das Trainieren des Modells.

Erstellen eines Projekts

Ein Amazon Rekognition Custom Labels-Projekt ist eine Gruppe von Ressourcen, die zum Erstellen und Verwalten eines Modells benötigt werden. Ein Projekt verwaltet Folgendes:

- Datensätze – Die Bilder und Bildlabels, die zum Trainieren eines Modells verwendet werden. Ein Projekt hat einen Trainingsdatensatz und einen Testdatensatz.

- Modelle – Die Software, die Sie trainieren, um Konzepte, Szenen und Objekte zu finden, die für Ihr Unternehmen einzigartig sind. Sie können mehrere Versionen eines Modells in einem Projekt haben.

Es wird empfohlen, ein Projekt für einen einzigen Anwendungsfall zu verwenden, z. B. für die Suche nach Leiterplattenteilen auf einer Leiterplatte.

Sie können ein Projekt mit der Amazon Rekognition Custom Labels-Konsole und mit der CreateProjectAPI erstellen. Weitere Informationen finden Sie unter Erstellen eines eines Projekts.

Erstellen von Trainings- und Testdatensätzen

Ein Datensatz besteht aus einer Reihe von Bildern und Labels, die diese Bilder beschreiben. In Ihrem Projekt erstellen Sie einen Trainingsdatensatz und einen Testdatensatz, den Amazon Rekognition Custom Labels zum Trainieren und Testen Ihres Modells verwendet.

Ein Label identifiziert ein Objekt, eine Szene, ein Konzept oder einen Begrenzungsrahmen, der ein Objekt in einem Bild umgibt. Labels werden entweder einem ganzen Bild (Bildebene) oder einem Begrenzungsrahmen zugewiesen, der ein Objekt auf einem Bild umgibt.

> **Important**
>
> Wie Sie die Bilder in Ihren Datensätzen beschriften, bestimmt den Modelltyp, den Amazon Rekognition Custom Labels erstellt. Um beispielsweise ein Modell zu trainieren, das Objekte, Szenen und Konzepte findet, weisen Sie den Bildern in Ihren Trainings- und Testdatensätzen

Labels auf Bildebene zu. Weitere Informationen finden Sie unter Datensätzen einen Zweck geben.

Bilder müssen im PNG- und JPEG-Format vorliegen, und Sie sollten die Empfehlungen für Eingabebilder befolgen. Weitere Informationen finden Sie unter Vorbereiten der Bilder.

Erstellen von Trainings- und Testdatensätzen (Konsole)

Sie können ein Projekt mit einem einzelnen Datensatz oder mit separaten Trainings- und Testdatensätzen beginnen. Wenn Sie mit einem einzelnen Datensatz beginnen, teilt Amazon Rekognition Custom Labels Ihren Datensatz während des Trainings auf, um einen Trainingsdatensatz (80 %) und einen Testdatensatz (20 %) für Ihr Projekt zu erstellen. Beginnen Sie mit einem einzigen Datensatz, wenn Amazon Rekognition Custom Labels entscheiden soll, welche Bilder zum Trainieren und Testen verwendet werden. Um die vollständige Kontrolle über Trainings, Tests und Leistungsoptimierungen zu haben, empfehlen wir, dass Sie Ihr Projekt mit separaten Trainings- und Testdatensätzen beginnen.

Um die Datensätze für ein Projekt zu erstellen, importieren Sie die Bilder auf eine der folgenden Arten:

- Importieren Sie Bilder von Ihrem lokalen Computer.
- Importieren Sie Bilder aus einem S3-Bucket. Amazon Rekognition Custom Labels kann die Bilder anhand der Ordnernamen beschriften, die die Bilder enthalten.
- Importieren Sie eine Amazon SageMaker Ground Truth Manifestdatei.
- Kopieren Sie einen bestehenden Amazon Rekognition Custom Labels-Datensatz.

Weitere Informationen finden Sie unter Erstellen von Trainings- und Testdatensätzen mit Bildern.

Je nachdem, von wo Sie Ihre Bilder importieren, haben Ihre Bilder möglicherweise keine Labels. Beispielsweise haben Bilder, die von einem lokalen Computer importiert wurden, keine Label. Bilder, die aus einer Amazon SageMaker Ground Truth Manifest-Datei importiert wurden, sind beschriftet. Sie können die Amazon Rekognition Custom Labels-Konsole verwenden, um Labels hinzuzufügen, zu ändern und zuzuweisen. Weitere Informationen finden Sie unter Labeling von Bildern.

Informationen zum Erstellen Ihrer Trainings- und Testdatensätze mit der Konsole finden Sie unter Erstellen von Trainings- und Testdatensätzen mit Bildern. Ein Tutorial, das das Erstellen von Trainings- und Testdatensätzen beinhaltet, finden Sie unter Tutorial: Bilder klassifizieren.

Erstellen von Trainings- und Testdatensätzen (SDK)

Um Ihre Trainings- und Testdatensätze zu erstellen, verwenden Sie die `CreateDataset`-API. Sie können einen Datensatz erstellen, indem Sie eine Manifestdatei im Amazon Sagemaker-Format verwenden oder einen vorhandenen Amazon Rekognition Custom Labels-Datensatz kopieren. Weitere Informationen finden Sie unter Erstellen von Trainings- und Testdatensätzen (SDK) . Bei Bedarf können Sie Ihre eigene Manifestdatei erstellen. Weitere Informationen finden Sie unter the section called “Erstellen einer Manifestdatei”.

Trainieren Ihres Modells

Trainieren Sie Ihr Modell mit dem Trainingsdatensatz. Bei jedem Training wird eine neue Version eines Modells erstellt. Während des Trainings testet Amazon Rekognition Custom Labels die Leistung Ihres trainierten Modells. Sie können die Ergebnisse verwenden, um Ihr Modell zu bewerten und zu verbessern. Es dauert eine Weile, bis das Training abgeschlossen ist. Ihnen wird nur ein erfolgreiches Modelltraining in Rechnung gestellt. Weitere Informationen finden Sie unter Schulung eines Amazon-Rekognition-Custom-Labels-Modells. Wenn das Modelltraining fehlschlägt, stellt Amazon Rekognition Custom Labels Debugging-Informationen bereit, die Sie verwenden können. Weitere Informationen finden Sie unter Debuggen eines fehlgeschlagenen Modelltrainings.

Ihr Modell trainieren (Konsole)

Informationen zum Trainieren Ihres Modells mit der Konsole finden Sie unter Ein Model trainieren (Konsole).

Ein Modell trainieren (SDK)

Sie trainieren ein Amazon Rekognition Custom Labels-Modell, indem Sie Version aufrufenCreateProject. Weitere Informationen finden Sie unter Ein Modell trainieren (SDK).

Verbessern Ihres Modells

Während des Tests erstellt Amazon Rekognition Custom Labels Bewertungsmetriken, mit denen Sie Ihr trainiertes Modell verbessern können.

Bewerten Ihres Modells

Bewerten Sie die Leistung Ihres Modells anhand der beim Testen erstellten Leistungsmetriken. Leistungsmetriken wie F1, Präzision und Rückruf ermöglichen es Ihnen, die Leistung Ihres trainierten

Modells zu verstehen und zu entscheiden, ob Sie es in der Produktion einsetzen möchten. Weitere Informationen finden Sie unter Metriken für die Bewertung Ihres Modells.

Bewerten eines Modells (Konsole)

Die Leistungsmetriken finden Sie unter Zugreifen auf Bewertungsmetriken (Konsole).

Bewerten eines Modells (SDK)

Um Leistungskennzahlen zu erhalten, rufen Sie DescribeProjectVersions auf, um die Testergebnisse abzurufen. Weitere Informationen finden Sie unter Zugreifen auf Amazon Rekognition Custom Labels-Bewertungsmetriken (SDK). Die Testergebnisse enthalten Metriken, die in der Konsole nicht verfügbar sind, z. B. eine Konfusionsmatrix für Klassifizierungsergebnisse. Die Testergebnisse werden in den folgenden Formaten zurückgegeben:

- F1-Score – Ein einzelner Wert, der die Gesamtleistung des Modells in Bezug auf Präzision und Rückrufvermögen darstellt. Weitere Informationen finden Sie unter F1.

- Speicherort der Datei mit der Zusammenfassung – Die Testzusammenfassung umfasst aggregierte Bewertungsmetriken für den gesamten Testdatensatz und Messwerte für jedes einzelne Label. `DescribeProjectVersions` gibt den S3-Bucket und den Speicherort des Ordners der Datei mit der Zusammenfassung zurück. Weitere Informationen finden Sie unter Übersichtsdatei.
- Speicherort des Snapshots des Bewertungsmanifests – Der Snapshot enthält Details zu den Testergebnissen, einschließlich der Konfidenzwerte und der Ergebnisse binärer Klassifizierungstests, z. B. falsch positiver Ergebnisse. `DescribeProjectVersions` gibt den S3-Bucket und den Speicherort der Snapshot-Dateien zurück. Weitere Informationen finden Sie unter Bewertungsmanifest-Snapshot.

Verbessern Ihres Modells

Wenn Verbesserungen erforderlich sind, können Sie weitere Trainingsbilder hinzufügen oder die Datensatz-Labels verbessern. Weitere Informationen finden Sie unter Verbessern eines Amazon Rekognition Custom Labels-Modells. Sie können auch Feedback zu den Vorhersagen geben, die Ihr Modell macht, und es verwenden, um Ihr Modell zu verbessern. Weitere Informationen finden Sie unter Feedback-Lösung modellieren.

Verbessern Sie Ihr Modell (Konsole)

Informationen zum Hinzufügen von Bildern zu einem Datensatz finden Sie unter Hinzufügen weiterer Bilder zu einem Datensatz. Informationen zum Hinzufügen oder Ändern von Labels finden Sie unter the section called “Labeling von Bildern”.

Informationen zum erneuten Trainieren Ihres Modells finden Sie unter Ein Model trainieren (Konsole).

Verbessern Sie Ihr Modell (SDK)

Verwenden Sie die `UpdateDatasetEntries`-API, um Bilder zu einem Datensatz hinzuzufügen oder die Labels für ein Bild zu ändern. `UpdateDatasetEntries` aktualisiert oder fügt JSON-Zeilen zu einer Manifestdatei hinzu. Jede JSON-Zeile enthält Informationen für ein einzelnes Bild, z. B. zugewiesene Labels oder Begrenzungsrahmen-Informationen. Weitere Informationen finden Sie unter Weitere Bilder hinzufügen (SDK). Verwenden Sie die `ListDatasetEntries`-API, um die Einträge in einem Datensatz anzuzeigen.

Informationen zum erneuten Trainieren Ihres Modells finden Sie unter Ein Modell trainieren (SDK).

Starten Ihres Modells

Bevor Sie Ihr Modell verwenden können, starten Sie das Modell mithilfe der Amazon Rekognition Custom Labels-Konsole oder der `StartProjectVersion`-API. Ihnen wird die Zeit in Rechnung gestellt, während der Ihr Modell ausgeführt wird. Weitere Informationen finden Sie unter Ein trainiertes Modell ausführen.

Starten Ihres Modells (Konsole)

Informationen zum Starten Ihres Modells mit der Konsole finden Sie unter Starten eines Amazon Rekognition Custom Labels-Modells (Konsole).

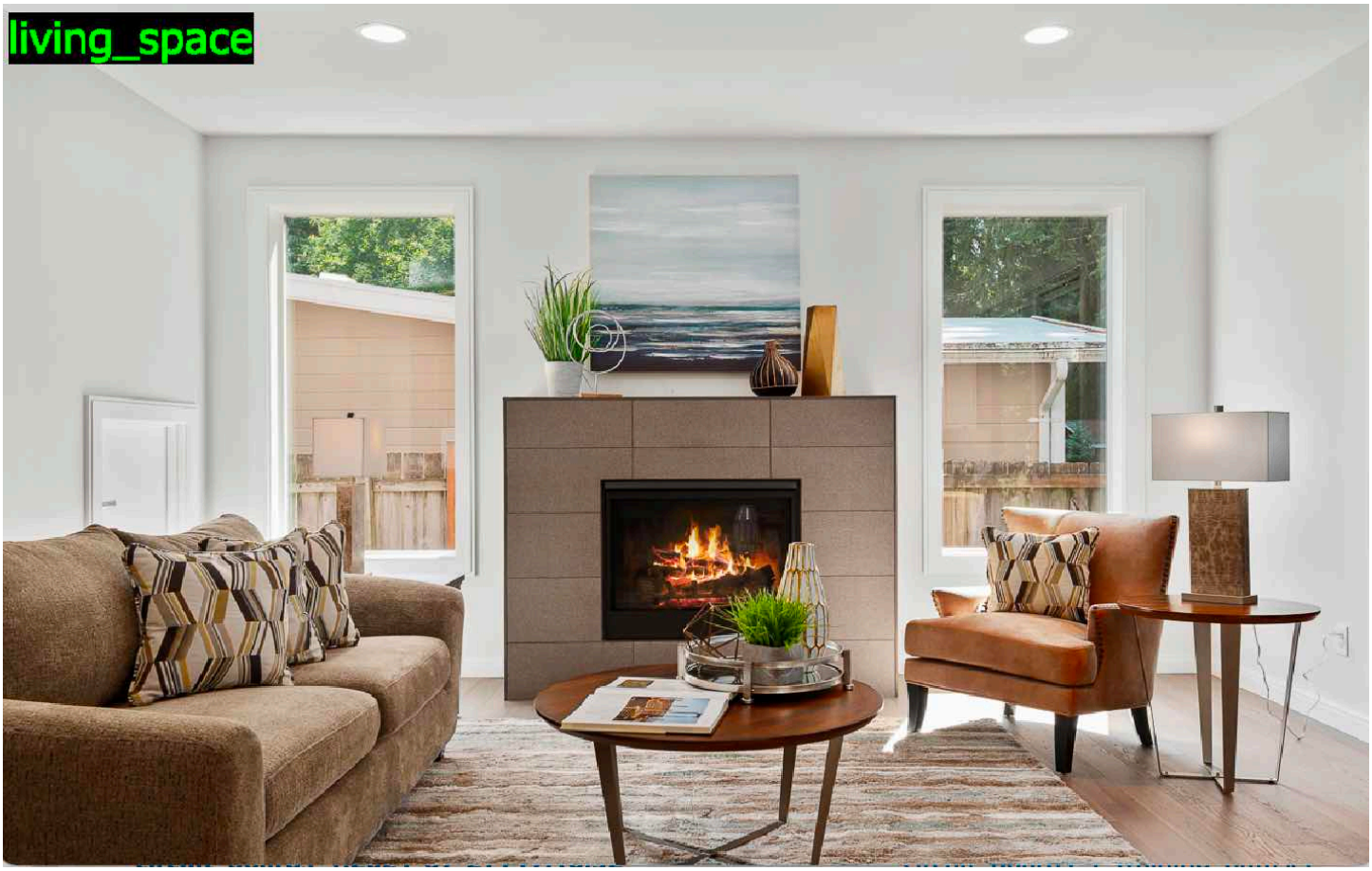
Starten Ihres Modells

Sie starten Ihr Modell mit dem Aufruf StartProjectvon Version. Weitere Informationen finden Sie unter Starten eines Amazon Rekognition Custom Labels-Modells (SDK).

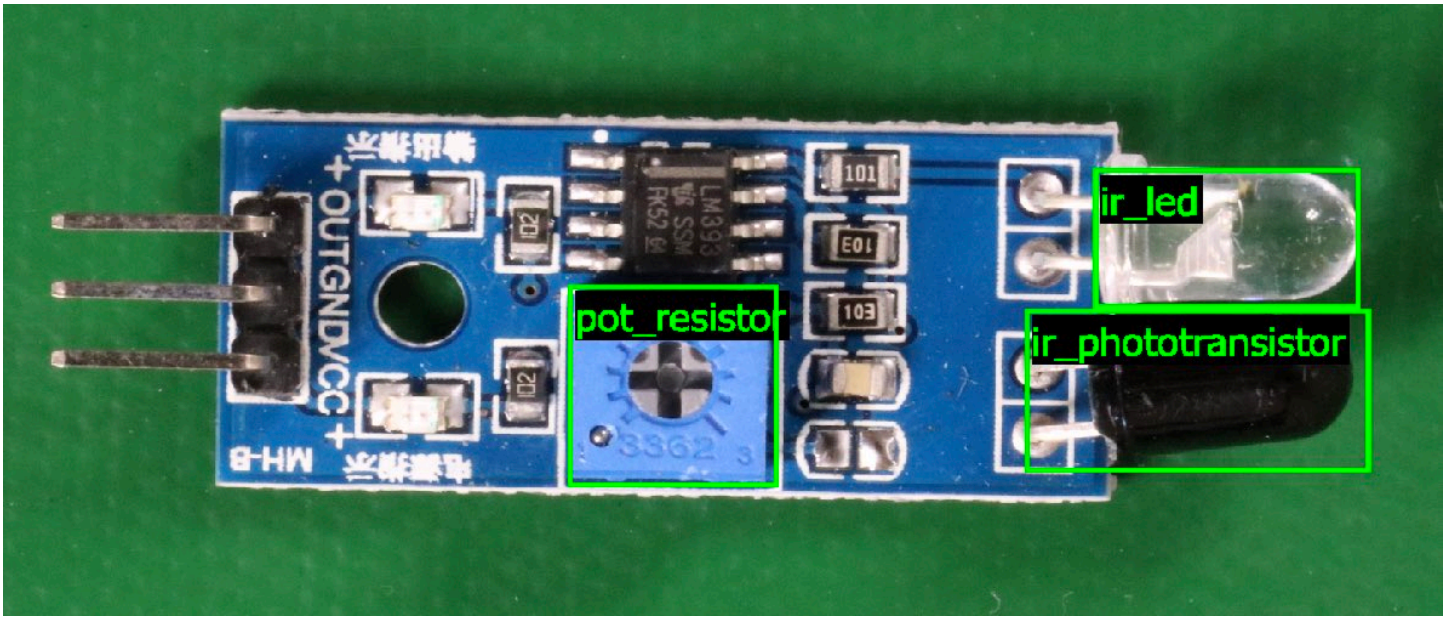
Analysieren eines Bilds

Um ein Bild mit Ihrem Modell zu analysieren, verwenden Sie die `DetectCustomLabels`-API. Sie können ein lokales Bild oder ein in einem S3-Bucket gespeichertes Bild angeben. Für den Vorgang ist auch der Amazon-Ressourcenname (ARN) des Modells, das Sie verwenden möchten, erforderlich.

Wenn Ihr Modell Objekte, Szenen und Konzepte findet, enthält die Antwort eine Liste der Labels auf Bildebene, die im Bild gefunden wurden. In der folgenden Abbildung werden beispielsweise die Labels auf Bildebene angezeigt, die mit dem Beispielprojekt Zimmer gefunden wurden.



Wenn das Modell Objektpositionen findet, enthält die Antwort eine Liste der mit Labels versehenen Begrenzungsrahmen, die im Bild zu finden sind. Ein Begrenzungsrahmen stellt die Position eines Objekts auf einem Bild dar. Sie können die Informationen zum Begrenzungsrahmen verwenden, um einen Begrenzungsrahmen um ein Objekt zu zeichnen. Die folgende Abbildung zeigt beispielsweise Begrenzungsrahmen rund um Leiterplattenteile, die im Beispielprojekt Leiterplatten gefunden wurden.



Weitere Informationen finden Sie unter Analysieren eines Bildes mit einem trainierten Modell.

Stoppen Ihres Modells

Ihnen wird die Zeit in Rechnung gestellt, während der Ihr Modell ausgeführt wird. Wenn Sie Ihr Modell nicht mehr verwenden, stoppen Sie es mithilfe der Amazon Rekognition Custom Labels-Konsole oder mithilfe der `StopProjectVersion`-API. Weitere Informationen finden Sie unter Stoppen eines Amazon Rekognition Custom Labels-Modells.

Stoppen Ihres Modells (Konsole)

Informationen zum Stoppen eines Modells, das mit der Konsole ausgeführt wird, finden Sie unter Stoppen eines Amazon Rekognition Custom Labels-Modells (Konsole).

Stoppen Ihres Modells (SDK)

Um ein laufendes Modell zu beenden, rufen Sie StopProject Version auf. Weitere Informationen finden Sie unter Stoppen eines Amazon Rekognition Custom Labels-Modells (SDK).

.

Trainiere dein Model

Projekte

- CreateProject— Erstellt Ihr Amazon Rekognition Custom Labels-Projekt, bei dem es sich um eine logische Gruppierung von Ressourcen (Bilder, Labels, Modelle) und Vorgängen (Schulung, Bewertung und Erkennung) handelt.
- DeleteProject— Löscht ein Amazon Rekognition Custom Labels Projekt
- DescribeProjects— Gibt eine Liste all Ihrer Amazon Rekognition Custom Labels-Projekte zurück.

Projektrichtlinien

- PutProjectPolicy— Fügt einem Amazon Rekognition Custom Labels-Projekt in einem vertrauenswürdigenAWS Konto eine Projektrichtlinie hinzu.
- ListProjectPolicies— Gibt eine Liste der Projektrichtlinien zurück, die einem Projekt zugeordnet sind.
- DeleteProjectPolicy— Löscht eine bestehende Projektrichtlinie.

Datensätze

- CreateDataset— Erstellt einen Amazon Rekognition Custom Labels Datensatz
- DeleteDataset— Löscht einen Amazon Rekognition Custom Labels Datensatz
- DescribeDataset— Beschreibt einen Amazon Rekognition Custom Labels Datensatz
- DistributeDatasetEntries— Verteilt die Einträge (Bilder) in einem Trainings-Datensatz über den Trainings-Datensatz und den Test-Datensatz für ein Projekt.
- ListDatasetEntries— Gibt eine Liste von Einträgen (Bildern) in einem Amazon Rekognition Custom Labels-Datensatz zurück.
- ListDatasetLabels— Gibt eine Liste von Labels zurück, die einem Amazon Rekognition Custom Labels-Datensatz zugewiesen wurden.
- UpdateDatasetEntries— Fügt Einträge (Bilder) in einem Amazon Rekognition Custom Labels-Datensatz hinzu oder aktualisiert sie.

Modelle

- CreateProjectVersion— Trainiert Ihr Amazon Rekognition Custom Labels Modell
- CopyProjectVersion— Kopiert Ihr Amazon Rekognition Custom Labels Modell
- DeleteProjectVersion— Löscht ein Amazon Rekognition Custom Labels Modell
- DescribeProjectVersions— Gibt eine Liste aller Amazon Rekognition Custom Labels-Modelle innerhalb eines bestimmten Projekts zurück.

Tags (Markierungen)

- TagResource— Fügt einem Amazon Rekognition Custom Labels-Modell ein oder mehrere Schlüsselwert-Tags hinzu.
- UntagResource— Entfernt ein oder mehrere Tags aus einem Amazon Rekognition Custom Labels-Modell.

Verwenden Sie Ihr Modell

- DetectCustomLabels— Analysiert ein Bild mit Ihrem benutzerdefinierten Labelmodell.
- StartProjectVersion— Startet Ihr benutzerdefiniertes Etikettenmodell.
- StopProjectVersion— Stoppt Ihr Modell für benutzerdefinierte Labels.

Dokumentverlauf für Amazon Rekognition Custom Labels

In der folgenden Tabelle sind wichtige Änderungen der einzelnen Versionen des Amazon Rekognition Custom Labels Benutzerhandbuchs beschrieben. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

- Letzte Aktualisierung der Dokumentation: 19. April 2023

Änderung	Beschreibung	Datum
Thema Modelldauer hinzugefügt	Zeigt, wie die Anzahl der Betriebsstunden und die von einem Modell verwendeten Inferenzeinheiten ermittelt werden. Weitere Informationen finden Sie unter Berichterstattung über die Laufdauer und die verwendeten Inferenzeinheiten.	19. April 2023
Reorganisierter Datensatzinhalt	Der Inhalt der Erstellung von Manifestdateien wurde in die Manifest-Datei verschoben. Die Themen zur Konvertierung von Datensätzen wurden in die Konvertierung anderer Datensatzformate in eine Manifestdatei verschoben.	20. Februar 2023
Die IAM-Anleitung für wurde aktualisiertAWS WAF	Aktualisierter Leitfaden, angepasst an die bewährten IAM-Methoden. Weitere Informationen finden Sie unter Bewährte IAM-Methoden.	15. Februar 2023

Sehen Sie sich die Konfusionsmatrix für ein Klassifizierungsmodell an	Die Amazon Rekognition Custom Labels-Konsole zeigt die Konfusionsmatrix für ein Klassifizierungsmodell nicht an. Stattdessen können Sie dasAWS SDK verwenden, um eine Konfusionsmatrix abzurufen und anzuzeigen. Weitere Informationen finden Sie unter Die Konfusionsmatrix für ein Modell anzeigen.	4. Januar 2023
Aktualisiertes Beispiel für eine Lambda-Funktion	Das Lambda-Funktionsbeispiel zeigt nun, wie Bilder analysiert werden, die aus einer lokalen Datei oder einem Amazon S3-Bucket übergeben wurden. Weitere Informationen finden Sie unter Analysieren von Bildern mit einer AWS Lambda Lambda-Funktion.	02. Dezember 2022
Amazon Rekognition Custom Labels können jetzt trainierte Modelle kopieren	Sie können jetzt ein trainiertes Modell von einemAWS Konto auf ein anderesAWS Konto innerhalb derselbenAWS Region kopieren. Weitere Informationen finden Sie unter Kopieren eines Amazon Rekognition Custom Labels-Modells (SDK).	16. August 2022

Amazon Rekognition Custom Labels können jetzt Inferenzeinheiten automatisch skalieren.	Um Nachfragespitzen entgegenzuwirken, kann Amazon Rekognition Custom Labels jetzt die Anzahl der Inferenzeinheiten, die Ihr Modell verwendet, skalieren. Weitere Informationen finden Sie unter Ausführen eines trainierten Amazon Rekognition Custom Labels-Modells.	16. August 2022
Erstellen Sie eine Manifestdatei aus einer CSV-Datei	Sie können jetzt die Erstellung einer Manifestdatei vereinfachen, indem Sie ein Skript verwenden, das Bildklassifizierungsinformationen aus einer CSV-Datei liest. Weitere Informationen finden Sie unter Erstellen einer Manifestdatei aus einer CSV-Datei.	2. Februar 2022
Amazon Rekognition Custom Labels verwaltet jetzt Datensätze mit Projekten	Sie können Projekte verwenden, um die Trainings- und Testdatensätze zu verwalten, die Sie zum Erstellen eines Modells verwenden. Weitere Informationen finden Sie unter Grundlegendes zu Amazon Rekognition Custom Labels.	1. November 2021

Amazon Rekognition Custom Labels ist integriert in AWS CloudFormation	Sie können AWS CloudFormation damit Amazon Rekognition Custom Labels Projekte bereitstellen und konfigurieren. Weitere Informationen finden Sie unter Erstellen eines Projekts mit AWS CloudFormation.	21. Oktober 2021
Aktualisiertes Einstiegserlebnis	Die Amazon Rekognition Custom Labels-Konsole enthält jetzt Tutorial-Videos und Beispielprojekte. Weitere Informationen finden Sie unter Erste Schritte mit Amazon Rekognition Custom Labels.	22. Juli 2021
Aktualisierte Informationen zu Schwellenwerten und zur Verwendung von Metriken	Informationen zur Einstellung eines gewünschten Schwellenwerts mithilfe des `MinConfidence` Eingabeparameters bis `DetectCustomLabels` . Weitere Informationen finden Sie unter Analysieren eines Bilds mit einem trainierten Modell.	8. Juni 2021
AWS KMS key Unterstützung hinzugefügt	Sie können jetzt Ihren eigenen KMS-Schlüssel verwenden , um Ihre Trainings- und Testbilder zu verschlüsseln. Weitere Informationen finden Sie unter Trainieren eines Modells.	19. Mai 2021

Tagging hinzugefügt	Amazon Rekognition Custom Labels unterstützt jetzt Tagging. Sie können Tags verwenden, um Ihre Amazon Rekognition Custom Labels-Modelle zu identifizieren, zu organisieren, nach ihnen zu suchen und zu filtern. Weitere Informationen finden Sie unter Taggen eines Modell.	25. März 2021
Aktualisiertes Setup-Thema	Die Setup-Informationen zum Verschlüsseln von Trainingsdateien wurden aktualisiert. Weitere Informationen finden Sie unter Schritt 5: (Optional) Verschlüsseln von Trainingsdateien.	18. März 2021
Thema zum Kopieren des Datensatzes hinzugefügt	Informationen zum Kopieren eines Datensatzes in eine andereAWS Region. Weitere Informationen finden Sie unter Einen Datensatz in eine andereAWS Region kopieren.	5. März 2021
Thema zur Transformation von Amazon-Manifesten SageMaker GroundTruth mit mehreren Labeln hinzugefügt	Informationen zur Umwandlung eines Manifests im SageMaker GroundTruth Amazon-Multi-Label-Format in eine Manifestdatei im Amazon Rekognition Custom Labels-Format. Weitere Informationen finden Sie unter Transformieren von SageMaker Ground Truth-Manifestdateien mit mehreren Labels.	22. Februar 2021

Debugging-Informationen für das Modelltraining hinzugefügt	Sie können jetzt Manifeste mit Validierungsergebnissen verwenden, um detaillierte Debugging-Informationen zu Modelltrainingsfehlern zu erhalten. Weitere Informationen finden Sie unter Debuggen eines fehlgeschlagenen Modelltraining.	8. Oktober 2020
Informationen und ein Beispiel für die COCO-Transformation wurden hinzugefügt	Informationen zur Umwandlung eines Datensatzes im COCO-Objekterkennungsformat in eine Amazon Rekognition Custom Labels-Manifestdatei. Weitere Informationen finden Sie unter Transformieren von COCO-Dasets.	2. September 2020
Amazon Rekognition Custom Labels unterstützt jetzt das Training mit einem einzelnen Objekt	Um ein Amazon Rekognition Custom Labels-Modell zu erstellen, das die Position eines einzelnen Objekts ermittelt, können Sie jetzt einen Datensatz erstellen, für den nur eine Bezeichnung erforderlich ist. Weitere Informationen finden Sie unter Zeichnen von Begrenzungsrahmen.	25. Juni 2020

Projekt- und Modelllöschvorgänge hinzugefügt	Sie können jetzt Amazon Rekognition Custom Labels-Projekte und -Modelle mit der Konsole und der API löschen. Weitere Informationen finden Sie unter Löschen eines Amazon Rekognition Custom Labels-Modells und Löschen eines Amazon Rekognition Custom Labels-Projekts	01. April 2020
Java-Beispiele hinzugefügt	Es wurden Java-Beispiele für Projekterstellung, Modelltraining, Modellausführung und Bildanalyse hinzugefügt.	13. Dezember 2019
Neue Funktion und Anleitung	Dies ist die erste Version der Amazon Rekognition Custom Labels-Funktion und des Amazon Rekognition Custom Labels Developer Guide.	3. Dezember 2019

AWS-Glossar

Die neueste AWS-Terminologie finden Sie im AWS-Glossar in der AWS-Glossar-Referenz.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.