aws

Panduan Developer

Amazon Lookout for Vision

Amazon Lookout for Vision: Panduan Developer

Table of Contents

Apa Amazon Lookout for Vision?

Anda dapat menggunakan Amazon Lookout for Vision untuk menemukan cacat visual pada produk industri, secara akurat dan dalam skala besar. Hal ini menggunakan computer vision untuk mengidentifikasi komponen yang hilang dalam produk industri, kerusakan pada kendaraan atau struktur, penyimpangan dalam jalur produksi, dan bahkan cacat kecil dalam wafer silikon—atau item fisik lainnya yang mengutamakan kualitas seperti kapasitor yang hilang pada papan sirkuit tercetak.

Manfaat utama

Amazon Lookout for Vision memberikan manfaat sebagai berikut:

- Meningkatkan proses dengan cepat dan efisien — Anda dapat menggunakan Amazon Lookout for Vision untuk mengimplementasikan inspeksi berbasis visi komputer dalam proses industri dengan cepat dan efisien, dalam skala besar. Anda dapat menyediakan sedikitnya 30 gambar dasar yang bagus dan Lookout for Vision dapat secara otomatis membuat model MS khusus untuk deteksi cacat. Anda kemudian dapat memproses gambar dari kamera IP, secara batch atau secara real time, untuk mengidentifikasi anomali seperti penyok, retakan, dan goresan dengan cepat dan akurat.
- Meningkatkan kualitas produksi, cepat - Dengan Amazon Lookout for Vision Anda dapat mengurangi cacat dalam proses produksi, secara real time. Ini mengidentifikasi dan melaporkan anomali visual di dasbor sehingga Anda dapat mengambil tindakan dengan cepat untuk menghentikan lebih banyak cacat dari terjadi—meningkatkan kualitas produksi dan mengurangi biaya.
- Kurangi biaya operasional — Amazon Lookout for Vision melaporkan tren dalam data inspeksi visual Anda, seperti mengidentifikasi proses dengan tingkat cacat tertinggi atau menandai variasi cacat terbaru. Dengan menggunakan informasi ini, Anda dapat menentukan apakah akan menjadwalkan pemeliharaan pada lini proses atau mengubah rute produksi ke mesin lain sebelum terjadi downtime yang mahal dan tidak direncanakan.

Apakah Anda baru pertama kali menggunakan Amazon Lookout for Vision?

Jika Anda baru pertama kali menggunakan Amazon Lookout for Vision, kami menyarankan agar Anda baru pertama kali menggunakan Amazon Lookout for Vision, kami menyarankan agar Anda membaca bagian-bagian berikut secara berurutan:

1. Menyiapkan Amazon Lookout for Vision— Di bagian ini, Anda mengatur detail akun Anda.
2. Memulai dengan Amazon Lookout for Vision- Di bagian ini, Anda belajar tentang membuat model Amazon Lookout for Vision pertama Anda.

Menyiapkan Amazon Lookout for Vision

Di bagian ini, Anda mendaftar untuk akun AWS dan menyiapkan Amazon Lookout for Vision.

Untuk informasi tentang AWS Wilayah yang mendukung Amazon Lookout for Vision, lihat Amazon Lookout for Vision Endpoint dan Quota.

Topik

Langkah 1: Buat AWS akun

Pada langkah ini, Anda mendaftar untuk AWS akun dan membuat pengguna administratif.

Topik

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

   Saat Anda mendaftar untuk sebuah Akun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai

praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan mengunjungi https://aws.amazon.com/ dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan Pengguna root akun AWS, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan Anda Pengguna root akun AWS

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.

2. Mengaktifkan autentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan perangkat MFA virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan Pengguna IAM.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat Identitas IAM.

   Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .

2. Di Pusat Identitas IAM, berikan akses administratif ke pengguna.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat Identitas IAM, gunakan URL masuk yang dikirim ke alamat email saat Anda membuat pengguna Pusat Identitas IAM.

  Untuk bantuan masuk menggunakan pengguna Pusat Identitas IAM, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat Identitas IAM, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.
2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Langkah 2: Siapkan izin

Untuk menggunakan Amazon Lookout for Vision, Anda memerlukan izin akses ke konsol Lookout for Vision, operasi AWS SDK, dan bucket Amazon S3 yang Anda gunakan untuk pelatihan model.

> **Note**
>
> Jika Anda hanya menggunakan operasi AWS SDK, Anda dapat menggunakan kebijakan yang dicakup untuk AWS operasi SDK. Untuk informasi selengkapnya, lihat Siapkan izin SDK.

Topik

- Menyetel akses konsol dengan kebijakan AWS terkelola
- Menyetel izin bucket Amazon S3
- Menetapkan izin

Menyetel akses konsol dengan kebijakan AWS terkelola

Gunakan kebijakan AWS terkelola berikut untuk menerapkan izin akses yang sesuai untuk konsol Amazon Lookout for Vision dan operasi SDK.

- AmazonLookoutVisionConsoleFullAccess— memungkinkan akses penuh ke konsol Amazon Lookout for Vision dan operasi SDK. Anda memerlukan `AmazonLookoutVisionConsoleFullAccess` izin untuk membuat bucket konsol. Untuk informasi selengkapnya, lihat Langkah 3: Buat bucket konsol.
- AmazonLookoutVisionConsoleReadOnlyAccess— memungkinkan akses hanya-baca ke konsol Amazon Lookout for Vision dan operasi SDK.

Untuk menetapkan izin, lihat. Menetapkan izin

Untuk informasi tentang kebijakan AWS terkelola, lihat kebijakan yang dikelola AWS.

Menyetel izin bucket Amazon S3

Amazon Lookout for Vision menggunakan bucket Amazon S3 untuk menyimpan file-file berikut:

- Gambar dataset — Gambar yang digunakan untuk melatih model. Untuk informasi selengkapnya, lihat Membuat dataset Anda.
- File manifes format Amazon SageMaker Ground Truth. Misalnya, output file manifes dari SageMaker GroundTruth pekerjaan. Untuk informasi selengkapnya, lihat Membuat kumpulan data menggunakan file manifes Amazon SageMaker Ground Truth.
- Output dari pelatihan model.

Jika Anda menggunakan konsol, Lookout for Vision akan membuat bucket Amazon S3 (bucket konsol) untuk mengelola proyek Anda. Kebijakan `LookoutVisionConsoleReadOnlyAccess` dan `LookoutVisionConsoleFullAccess` terkelola mencakup izin akses Amazon S3 untuk bucket konsol.

Anda dapat menggunakan bucket konsol untuk menyimpan gambar kumpulan data dan file manifes format SageMaker Ground Truth. Atau, Anda dapat menggunakan bucket Amazon S3 yang berbeda. Bucket harus dimiliki oleh akun AWS Anda dan harus berada di AWS Wilayah tempat Anda menggunakan Lookout for Vision.

Untuk menggunakan bucket yang berbeda, tambahkan kebijakan berikut ke pengguna atau grup yang diinginkan. Ganti `my-bucket` dengan nama ember yang diinginkan. Untuk informasi tentang menambahkan kebijakan IAM, lihat Membuat Kebijakan IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionS3BucketAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket"
            ]
        },
        {
            "Sid": "LookoutVisionS3ObjectAccessPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::my-bucket/*"
            ]
        }
    ]
}
```

Untuk menetapkan izin, lihat. Menetapkan izin

Menetapkan izin

Untuk memberikan akses, menambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

Buat rangkaian izin. Ikuti instruksi di Buat rangkaian izin di Panduan Pengguna AWS IAM Identity Center .

- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti instruksi dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.

- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam Membuat peran untuk pengguna IAM dalam Panduan Pengguna IAM.
  - (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti instruksi dalam Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

Langkah 3: Buat bucket konsol

Untuk menggunakan konsol Amazon Lookout for Vision, Anda memerlukan bucket Amazon S3 yang dikenal sebagai bucket konsol. Bucket konsol menyimpan yang berikut ini:

- Gambar yang Anda unggah ke kumpulan data dengan konsol.
- Hasil pelatihan untuk pelatihan model yang Anda mulai dengan konsol.
- Hasil deteksi percobaan.
- File manifes sementara yang dibuat konsol saat Anda menggunakan konsol untuk membuat kumpulan data dengan memberi label gambar secara otomatis di bucket S3. Konsol tidak menghapus file manifes.

Saat pertama kali membuka konsol Amazon Lookout for Vision di Wilayah AWS baru, Lookout for Vision membuat bucket konsol atas nama Anda. Perhatikan nama bucket konsol karena Anda mungkin perlu menggunakan nama bucket dalam operasi AWS SDK atau tugas konsol, seperti membuat kumpulan data.

Atau, Anda dapat membuat bucket konsol dengan menggunakan Amazon S3. Gunakan pendekatan ini jika kebijakan bucket Amazon S3 tidak membiarkan konsol Amazon Lookout for Vision berhasil membuat bucket konsol. Misalnya, kebijakan yang melarang pembuatan otomatis bucket Amazon S3.

> **Note**
>
> Jika Anda hanya menggunakan AWS SDK dan bukan konsol Lookout for Vision, Anda tidak perlu membuat bucket konsol. Anda dapat menggunakan bucket S3 yang berbeda dengan nama pilihan Anda.

Format nama bucket konsol adalah `lookoutvision` - <region>-<random value>. Nilai acak memastikan bahwa tidak ada tabrakan antara nama bucket.

Topik

Membuat bucket konsol dengan konsol Amazon Lookout for Vision

Gunakan prosedur berikut untuk membuat bucket konsol untuk AWS Wilayah dengan konsol Amazon Lookout for Vision. Untuk informasi tentang pengaturan bucket S3 yang kami aktifkan, lihatPengaturan bucket konsol.

Untuk membuat bucket konsol dengan menggunakan konsol Amazon Lookout for Vision

1. Pastikan pengguna atau grup yang Anda gunakan memiliki `AmazonLookoutVisionConsoleFullAccess` izin. Untuk informasi selengkapnya, lihat Langkah 2: Siapkan izin.
2. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
3. Pada bilah navigasi, pilih Pilih wilayah. Kemudian pilih AWS Wilayah yang ingin Anda buat bucket konsol.
4. Pilih Mulai.
5. Jika ini adalah pertama kalinya Anda membuka konsol di Wilayah AWS saat ini, lakukan hal berikut di kotak dialog Penyiapan pertama kali:
    a. Salin nama bucket Amazon S3 yang ditampilkan. Anda akan memerlukan informasi ini nanti.
    b. Pilih Buat bucket S3 agar Amazon Lookout for Vision membuat bucket konsol atas nama Anda.

Kotak dialog Penyetelan pertama kali tidak ditampilkan jika bucket konsol untuk AWS Wilayah saat ini sudah ada.

6. Tutup jendela browser.

Membuat bucket konsol dengan Amazon S3

Anda dapat menggunakan Amazon S3 untuk membuat bucket konsol. Anda harus membuat bucket dengan mengaktifkan versi Amazon S3. Sebaiknya gunakan konfigurasi siklus hidup Amazon S3 untuk menghapus versi objek yang tidak aktif (sebelumnya) dan menghapus unggahan multibagian yang tidak lengkap. Kami tidak merekomendasikan konfigurasi siklus hidup yang menghapus versi objek saat ini. Untuk informasi tentang pengaturan bucket S3 yang kami aktifkan untuk bucket konsol yang Anda buat dengan konsol Amazon Lookout for Vision, lihat. Pengaturan bucket konsol

1. Tentukan AWS Wilayah tempat Anda ingin membuat bucket konsol. Untuk informasi tentang wilayah yang didukung, lihat titik akhir dan kuota Amazon Lookout for Vision.
2. Buat bucket menggunakan instruksi konsol S3 di Membuat ember. Lakukan hal-hal berikut:

   a. Untuk langkah 3 tentukan nama bucket yang ditambahkan. `lookoutvision-`*`region`*`-`*`your-identifier`* Ubah `region` ke kode wilayah yang Anda pilih pada langkah sebelumnya. Ubah `your-identifier` ke pengenal unik pilihan Anda. Misalnya, `lookoutvision-us-east-1-my-console-bucket-1`

   b. Untuk langkah 4 pilih AWS Wilayah yang ingin Anda gunakan.
3. Aktifkan pembuatan versi untuk bucket dengan mengikuti instruksi konsol S3 di Mengaktifkan versi pada bucket.
4. (Opsional) Tentukan konfigurasi siklus hidup untuk bucket dengan mengikuti instruksi konsol S3 di Menyetel konfigurasi siklus hidup pada bucket. Lakukan hal berikut untuk menghapus versi noncurrent (sebelumnya) dari objek dan menghapus unggahan multipart yang tidak lengkap. Anda tidak perlu melakukan langkah 6, 8, 9, 10.

   a. Untuk langkah 5 pilih Terapkan ke semua objek di ember.

   b. Untuk langkah 7 pilih Hapus secara permanen versi objek yang tidak terkini dan Hapus penanda hapus objek kedaluwarsa atau unggahan multibagian yang tidak lengkap.

   c. Untuk langkah 11 masukkan jumlah hari untuk menunggu sebelum menghapus versi objek yang tidak terkini.

d. Untuk langkah 12 masukkan jumlah hari untuk menunggu sebelum menghapus unggahan multibagian yang tidak lengkap.

Pengaturan bucket konsol

Jika Anda membuat bucket konsol dengan konsol Amazon Lookout for Vision, kami mengaktifkan pengaturan berikut di bucket konsol.

- Pembuatan versi objek di bucket konsol.
- Enkripsi objek sisi server di bucket konsol.
- Konfigurasi siklus hidup untuk penghapusan objek noncurrent (30 hari) dan unggahan multipart yang tidak lengkap (3 hari).
- Blokir akses publik ke bucket konsol.

Langkah 4: Siapkan AWS CLI dan AWS SDK

Langkah-langkah berikut menunjukkan cara menginstal AWS Command Line Interface (AWS CLI) dan AWS SDK. Contoh dalam dokumentasi ini menggunakan SDK AWS CLI, Python, dan Java AWS .

Topik

- Instal AWS SDKS
- Memberikan akses programatis
- Siapkan izin SDK
- Hubungi operasi Amazon Lookout for Vision

Instal AWS SDKS

Ikuti langkah-langkah untuk mengunduh dan mengonfigurasi AWS SDK.

Untuk mengatur AWS CLI dan AWS SDK

- Unduh dan instal AWS CLIdan AWS SDK yang ingin Anda gunakan. Panduan ini memberikan contoh untuk Java AWS CLI, dan Python. Untuk informasi tentang menginstal AWS SDK, lihat Alat untuk Amazon Web Services.

Memberikan akses programatis

Anda dapat menjalankan contoh AWS CLI dan kode dalam panduan ini di komputer lokal atau AWS lingkungan lain, seperti instans Amazon Elastic Compute Cloud. Untuk menjalankan contoh, Anda perlu memberikan akses ke operasi AWS SDK yang digunakan contoh.

Topik

Menjalankan kode di komputer lokal Anda

Untuk menjalankan kode di komputer lokal, sebaiknya gunakan kredensyal jangka pendek untuk memberikan akses pengguna ke operasi AWS SDK. Untuk informasi spesifik tentang menjalankan contoh kode AWS CLI dan pada komputer lokal, lihatMenggunakan profil di komputer lokal Anda.

Pengguna membutuhkan akses terprogram jika mereka ingin berinteraksi dengan AWS luar. AWS Management Console Cara untuk memberikan akses terprogram tergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses programatis, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
Identitas tenaga kerja (Pengguna yang dikelola di Pusat Identitas IAM)	Gunakan kredensyal sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk AWS CLI, lihat Mengkonfigurasi yang akan AWS CLI digunakan AWS IAM Identity Center dalam Panduan AWS Command Line Interface Pengguna. • Untuk AWS SDK, alat, dan AWS API, lihat autentika

Pengguna mana yang membutuhkan akses programatis?	Untuk	Oleh
		si Pusat Identitas IAM di Panduan Referensi AWS SDK dan Alat.
IAM	Gunakan kredensyal sementara untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk dalam Menggunakan kredensyal sementara dengan AWS sumber daya di Panduan Pengguna IAM.
IAM	(Tidak direkomendasikan) Gunakan kredensyal jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, AWS SDK, atau API. AWS	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. • Untuk mengetahui AWS CLI, lihat Mengautentikasi menggunakan kredensyal pengguna IAM di Panduan Pengguna.AWS Command Line Interface • Untuk AWS SDK dan alat bantu, lihat Mengautentikasi menggunakan kredensyal jangka panjang di Panduan Referensi AWS SDK dan Alat. • Untuk AWS API, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Menggunakan profil di komputer lokal Anda

Anda dapat menjalankan contoh AWS CLI dan kode dalam panduan ini dengan kredensyal jangka pendek yang Anda buat. Menjalankan kode di komputer lokal Anda Untuk mendapatkan kredensyal dan informasi pengaturan lainnya, contoh menggunakan profil bernama `lookoutvision-access` Misalnya:

```
session = boto3.Session(profile_name='lookoutvision-access')
lookoutvision_client = session.client("lookoutvision")
```

Pengguna yang diwakili profil harus memiliki izin untuk memanggil operasi Lookout for Vision SDK dan operasi SDK lainnya AWS yang diperlukan oleh contoh. Untuk informasi selengkapnya, lihat Siapkan izin SDK. Untuk menetapkan izin, lihat. Menetapkan izin

Untuk membuat profil yang sesuai dengan contoh kode AWS CLI dan, pilih salah satu dari berikut ini. Pastikan nama profil yang Anda buat adalah`lookoutvision-access`.

- Pengguna yang dikelola oleh IAM — Ikuti petunjuk di Beralih ke peran IAM (AWS CLI).
- Identitas tenaga kerja (Pengguna dikelola oleh AWS IAM Identity Center) — Ikuti petunjuk di Mengonfigurasi AWS CLI untuk digunakan. AWS IAM Identity Center Untuk contoh kode, sebaiknya gunakan Integrated Development Environment (IDE), yang mendukung AWS Toolkit yang mengaktifkan otentikasi melalui IAM Identity Center. Untuk contoh Java, lihat Mulai membangun dengan Java. Untuk contoh Python, lihat Mulai membangun dengan Python. Untuk informasi selengkapnya, lihat kredensil Pusat Identitas IAM.

> **Note**
>
> Anda dapat menggunakan kode untuk mendapatkan kredensi jangka pendek. Untuk informasi selengkapnya, lihat Beralih ke peran IAM (AWS API). Untuk Pusat Identitas IAM, dapatkan kredensi jangka pendek untuk suatu peran dengan mengikuti instruksi di Mendapatkan kredensil peran IAM untuk akses CLI.

Menjalankan kode di AWS lingkungan

Anda tidak boleh menggunakan kredensyal pengguna untuk menandatangani panggilan AWS SDK di AWS lingkungan, seperti kode produksi yang berjalan dalam suatu fungsi. AWS Lambda Sebagai

gantinya, Anda mengonfigurasi peran yang menentukan izin yang dibutuhkan kode Anda. Anda kemudian melampirkan peran ke lingkungan tempat kode Anda berjalan. Cara Anda melampirkan peran dan membuat kredensyal sementara tersedia bervariasi tergantung pada lingkungan tempat kode Anda berjalan:

- AWS Lambda fungsi — Gunakan kredenal sementara yang secara otomatis disediakan Lambda ke fungsi Anda saat mengasumsikan peran eksekusi fungsi Lambda. Kredensialnya tersedia di variabel lingkungan Lambda. Anda tidak perlu menentukan profil. Untuk informasi selengkapnya, silakan lihat Peran eksekusi Lambda.
- Amazon EC2 — Gunakan penyedia kredensil titik akhir metadata instans Amazon EC2. Penyedia secara otomatis membuat dan menyegarkan kredensil untuk Anda menggunakan profil instans Amazon EC2 yang Anda lampirkan ke instans Amazon EC2. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2
- Amazon Elastic Container Service — Gunakan penyedia kredensi Container. Amazon ECS mengirim dan menyegarkan kredensional ke titik akhir metadata. Peran IAM tugas yang Anda tentukan menyediakan strategi untuk mengelola kredensyal yang digunakan aplikasi Anda. Untuk informasi selengkapnya, lihat Berinteraksi dengan layanan AWS.
- Perangkat inti Greengrass - Gunakan sertifikat X.509 untuk terhubung AWS ke IoT Core menggunakan protokol otentikasi timbal balik TLS. Sertifikat ini memungkinkan perangkat berinteraksi dengan AWS IoT tanpa kredensi AWS. Penyedia kredensi AWS IoT mengautentikasi perangkat menggunakan sertifikat X.509 dan mengeluarkan kredensi AWS dalam bentuk token keamanan hak istimewa terbatas sementara. Untuk informasi selengkapnya, lihat Berinteraksi dengan layanan AWS.

Untuk informasi selengkapnya tentang penyedia kredensi, lihat Penyedia kredensi terstandarisasi.

Siapkan izin SDK

Untuk menggunakan operasi Amazon Lookout for Vision SDK, Anda memerlukan izin akses ke Lookout for Vision API dan bucket Amazon S3 yang digunakan untuk pelatihan model.

Topik

- Memberikan izin operasi SDK
- Memberikan izin Bucket Amazon S3
- Menetapkan izin

Memberikan izin operasi SDK

Sebaiknya Anda hanya memberikan izin yang diperlukan untuk melakukan tugas (izin hak istimewa paling sedikit). Misalnya, untuk menelepon DetectAnomalies, Anda memerlukan izin untuk melakukan`lookoutvision:DetectAnomalies`. Untuk menemukan izin operasi, periksa referensi API.

Ketika Anda baru memulai dengan aplikasi, Anda mungkin tidak tahu izin spesifik yang Anda butuhkan, sehingga Anda dapat mulai dengan izin yang lebih luas. AWS kebijakan terkelola memberikan izin untuk membantu Anda memulai.

- AmazonLookoutVisionFullAccess— memungkinkan akses penuh ke operasi Amazon Lookout for Vision SDK.
- AmazonLookoutVisionReadOnlyAccess— memungkinkan akses ke operasi SDK read-only.

Kebijakan terkelola untuk konsol juga menyediakan izin akses untuk operasi SDK. Untuk informasi selengkapnya, lihat Langkah 2: Siapkan izin.

Untuk informasi tentang kebijakan AWS terkelola, lihat kebijakan yang dikelola AWS.

Bila Anda mengetahui izin yang dibutuhkan aplikasi Anda, kurangi izin lebih lanjut dengan menentukan kebijakan terkelola pelanggan khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan yang dikelola pelanggan.

> **Note**
>
> Instruksi memulai memerlukan `s3:PutObject` izin. Untuk informasi selengkapnya, lihat Langkah 1: Buat file manifes dan unggah gambar.

Untuk menetapkan izin, lihat. Menetapkan izin

Memberikan izin Bucket Amazon S3

Untuk melatih model, Anda memerlukan bucket Amazon S3 dengan izin yang sesuai untuk menyimpan gambar, file manifes, dan output pelatihan. Bucket harus dimiliki oleh akun AWS Anda dan harus berada di Wilayah AWS tempat Anda menggunakan Amazon Lookout for Vision.

Kebijakan terkelola khusus SDK (`AmazonLookoutVisionFullAccess`dan`AmazonLookoutVisionReadOnlyAccess`) tidak

menyertakan izin bucket Amazon S3 dan Anda perlu menerapkan kebijakan izin berikut untuk mengakses bucket yang Anda gunakan, termasuk bucket konsol yang ada.

Kebijakan terkelola konsol (`AmazonLookoutVisionConsoleFullAccess`dan`AmazonLookoutVisionConsoleReadOnlyAccess`) menyertakan izin akses ke bucket konsol. Jika Anda mengakses bucket konsol dengan operasi SDK dan memiliki izin kebijakan terkelola konsol, Anda tidak perlu menggunakan kebijakan berikut. Untuk informasi selengkapnya, lihat Langkah 2: Siapkan izin.

Memutuskan izin tugas

Gunakan informasi berikut untuk memutuskan izin mana yang diperlukan untuk tugas yang ingin Anda lakukan.

Membuat kumpulan data

Untuk membuat kumpulan data dengan CreateDataset, Anda memerlukan izin berikut.

- `s3:GetBucketLocation`— memungkinkan Lookout for Vision untuk memvalidasi bahwa bucket Anda berada di wilayah yang sama dengan tempat Anda menggunakan Lookout for Vision.
- `s3:GetObject`— Memungkinkan akses ke file manifes yang ditentukan dalam parameter `DatasetSource` input. Jika Anda ingin menentukan versi objek S3 yang tepat dari file manifes, Anda juga perlu `s3:GetObjectVersion` pada file manifes. Untuk informasi selengkapnya, lihat Menggunakan pembuatan versi di bucket S3.

Membuat model

Untuk membuat model dengan CreateModel, Anda memerlukan izin berikut.

- `s3:GetBucketLocation`— memungkinkan Lookout for Vision untuk memvalidasi bahwa bucket Anda berada di wilayah yang sama dengan tempat Anda menggunakan Lookout for Vision.
- `s3:GetObject`— memungkinkan akses ke gambar yang ditentukan dalam kumpulan data pelatihan dan pengujian proyek.
- `s3:PutObject`— memungkinkan izin untuk menyimpan output pelatihan dalam ember yang ditentukan. Anda menentukan lokasi bucket keluaran dalam `OutputConfig` parameter. Secara opsional, Anda dapat mencakupkan izin hingga hanya kunci objek yang ditentukan di `Prefix` bidang bidang `S3Location` input. Untuk informasi lebih lanjut, lihat OutputConfig.

Mengakses gambar, file manifes, dan output pelatihan

Izin bucket Amazon S3 tidak diperlukan untuk melihat respons operasi Amazon Lookout for Vision. Anda memerlukan `s3:GetObject` izin jika Anda ingin mengakses gambar, file manifes, dan output pelatihan yang direferensikan dalam respons operasi. Jika Anda mengakses objek Amazon S3 berversi, Anda memerlukan izin. `s3:GetObjectVersion`

Menyetel kebijakan bucket Amazon S3

Anda dapat menggunakan kebijakan berikut untuk menentukan izin bucket Amazon S3 yang diperlukan untuk membuat kumpulan data (`CreateDataset`), membuat model (`CreateModel`), dan mengakses gambar, file manifes, dan output pelatihan. Ubah nilai *`my-bucket`* menjadi nama bucket yang ingin Anda gunakan.

Anda dapat menyesuaikan kebijakan dengan kebutuhan Anda. Untuk informasi selengkapnya, lihat Memutuskan izin tugas. Tambahkan kebijakan ke pengguna yang diinginkan. Untuk informasi selengkapnya, lihat Membuat Kebijakan IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionS3BucketAccess",
            "Effect": "Allow",
            "Action": "s3:GetBucketLocation",
            "Resource": [
                "arn:aws:s3:::my-bucket"
            ],
            "Condition": {
                "Bool": {
                    "aws:ViaAWSService": "true"
                }
            }
        },
        {
            "Sid": "LookoutVisionS3ObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::my-bucket/*"
            ],
            "Condition": {
                "Bool": {
                    "aws:ViaAWSService": "true"
                }
            }
        }
    ]
}
```

Untuk menetapkan izin, lihat. Menetapkan izin

Menetapkan izin

Untuk memberikan akses, menambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup di AWS IAM Identity Center:

  Buat rangkaian izin. Ikuti instruksi di Buat rangkaian izin di Panduan Pengguna AWS IAM Identity Center .
- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti instruksi dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.
- Pengguna IAM:
  - Buat peran yang dapat diambil pengguna Anda. Ikuti instruksi dalam Membuat peran untuk pengguna IAM dalam Panduan Pengguna IAM.
  - (Tidak disarankan) Pasang kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti instruksi dalam Menambahkan izin ke pengguna (konsol) dalam Panduan Pengguna IAM.

Hubungi operasi Amazon Lookout for Vision

Jalankan kode berikut untuk mengonfirmasi bahwa Anda dapat melakukan panggilan ke Amazon Lookout for Vision API. Kode mencantumkan proyek di AWS akun Anda, di AWS Wilayah saat ini. Jika sebelumnya Anda belum membuat proyek, responsnya kosong, tetapi mengonfirmasi bahwa Anda dapat memanggil `ListProjects` operasi.

Secara umum, memanggil fungsi contoh memerlukan klien AWS SDK Lookout for Vision dan parameter lain yang diperlukan. Klien AWS SDK Lookout for Vision dideklarasikan dalam fungsi utama.

Jika kode gagal, periksa apakah pengguna yang Anda gunakan memiliki izin yang benar. Periksa juga AWS Wilayah yang Anda gunakan sebagai Amazon Lookout for Vision tidak tersedia di semua Wilayah. AWS

Untuk memanggil operasi Amazon Lookout for Vision

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode contoh berikut untuk melihat proyek Anda.

   CLI

   Gunakan `list-projects` perintah untuk membuat daftar proyek di akun Anda.

   ```
   aws lookoutvision list-projects \
   --profile lookoutvision-access
   ```

   Python

   Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

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


   from botocore.exceptions import ClientError
   import boto3

   class GettingStarted:

       @staticmethod
       def list_projects(lookoutvision_client):
           """
           Lists information about the projects that are in in your AWS account
   ```

```
        and in the current AWS Region.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            for project in response["Projects"]:
                print("Project: " + project["ProjectName"])
                print("ARN: " + project["ProjectArn"])
                print()
            print("Done!")
        except ClientError:
            raise
def main():
    session = boto3.Session(profile_name='lookoutvision-access')
    lookoutvision_client = session.client("lookoutvision")

    GettingStarted.list_projects(lookoutvision_client)

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

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

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

package com.example.lookoutvision;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.services.lookoutvision.LookoutVisionClient;
import software.amazon.awssdk.services.lookoutvision.model.ProjectMetadata;
import
 software.amazon.awssdk.services.lookoutvision.paginators.ListProjectsIterable;
import software.amazon.awssdk.services.lookoutvision.model.ListProjectsRequest;
import
 software.amazon.awssdk.services.lookoutvision.model.LookoutVisionException;
```

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

public class GettingStarted {

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

    /**
     * Lists the Amazon Lookoutfor Vision projects in the current AWS account
 and
     * AWS Region.
     *
     * @param lfvClient   An Amazon Lookout for Vision client.
     * @return List<ProjectMetadata> Metadata for each project.
     */
    public static List<ProjectMetadata> listProjects(LookoutVisionClient
 lfvClient)
            throws LookoutVisionException {

        logger.log(Level.INFO, "Getting projects:");
        ListProjectsRequest listProjectsRequest = ListProjectsRequest.builder()
                .maxResults(100)
                .build();

        List<ProjectMetadata> projectMetadata = new ArrayList<>();

        ListProjectsIterable projects =
 lfvClient.listProjectsPaginator(listProjectsRequest);

        projects.stream().flatMap(r -> r.projects().stream())
                .forEach(project -> {
                    projectMetadata.add(project);
                    logger.log(Level.INFO, project.projectName());
                });

        logger.log(Level.INFO, "Finished getting projects.");

        return projectMetadata;

    }
```

```
    public static void main(String[] args) throws Exception {

        try {

            // Get the Lookout for Vision client.
            LookoutVisionClient lfvClient = LookoutVisionClient.builder()

 .credentialsProvider(ProfileCredentialsProvider.create("lookoutvision-access"))
                    .build();

            List<ProjectMetadata> projects = Projects.listProjects(lfvClient);

            System.out.printf("Projects%n--------%n");

            for (ProjectMetadata project : projects) {
                System.out.printf("Name: %s%n", project.projectName());
                System.out.printf("ARN: %s%n", project.projectArn());
                System.out.printf("Date: %s%n%n",
 project.creationTimestamp().toString());
            }

        } catch (LookoutVisionException lfvError) {
            logger.log(Level.SEVERE, "Could not list projects: {0}: {1}",
                    new Object[] { lfvError.awsErrorDetails().errorCode(),
                            lfvError.awsErrorDetails().errorMessage() });
            System.out.println(String.format("Could not list projects: %s",
 lfvError.getMessage()));
            System.exit(1);
        }

    }

}
```

Langkah 5: (Opsional) Menggunakan kunci AWS Key Management Service Anda sendiri

Anda dapat menggunakan AWS Key Management Service (KMS) untuk mengelola enkripsi gambar masukan yang Anda simpan di bucket Amazon S3.

Secara default gambar Anda dienkripsi dengan kunci yang dimiliki dan dikelola AWS. Anda juga dapat memilih untuk menggunakan kunci AWS Key Management Service (KMS) milik Anda sendiri. Untuk informasi selengkapnya, lihat Konsep AWS Key Management Service.

Jika Anda ingin menggunakan kunci KMS Anda sendiri, gunakan kebijakan berikut untuk menentukan kunci KMS. Ubah *kms_key_arn* ke ARN dari kunci KMS (atau alias KMS ARN) yang ingin Anda gunakan. Atau, tentukan * untuk menggunakan kunci KMS apa pun. Untuk informasi tentang menambahkan kebijakan ke pengguna atau peran, lihat Membuat Kebijakan IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionKmsDescribeAccess",
            "Effect": "Allow",
            "Action": "kms:DescribeKey",
            "Resource": "kms_key_arn"
        },
        {
            "Sid": "LookoutVisionKmsCreateGrantAccess",
            "Effect": "Allow",
            "Action": "kms:CreateGrant",
            "Resource": "kms_key_arn",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": "lookoutvision.*.amazonaws.com"
                },
                "Bool": {
                    "kms:GrantIsForAWSResource": "true"
                }
            }
        }
    ]
}
```

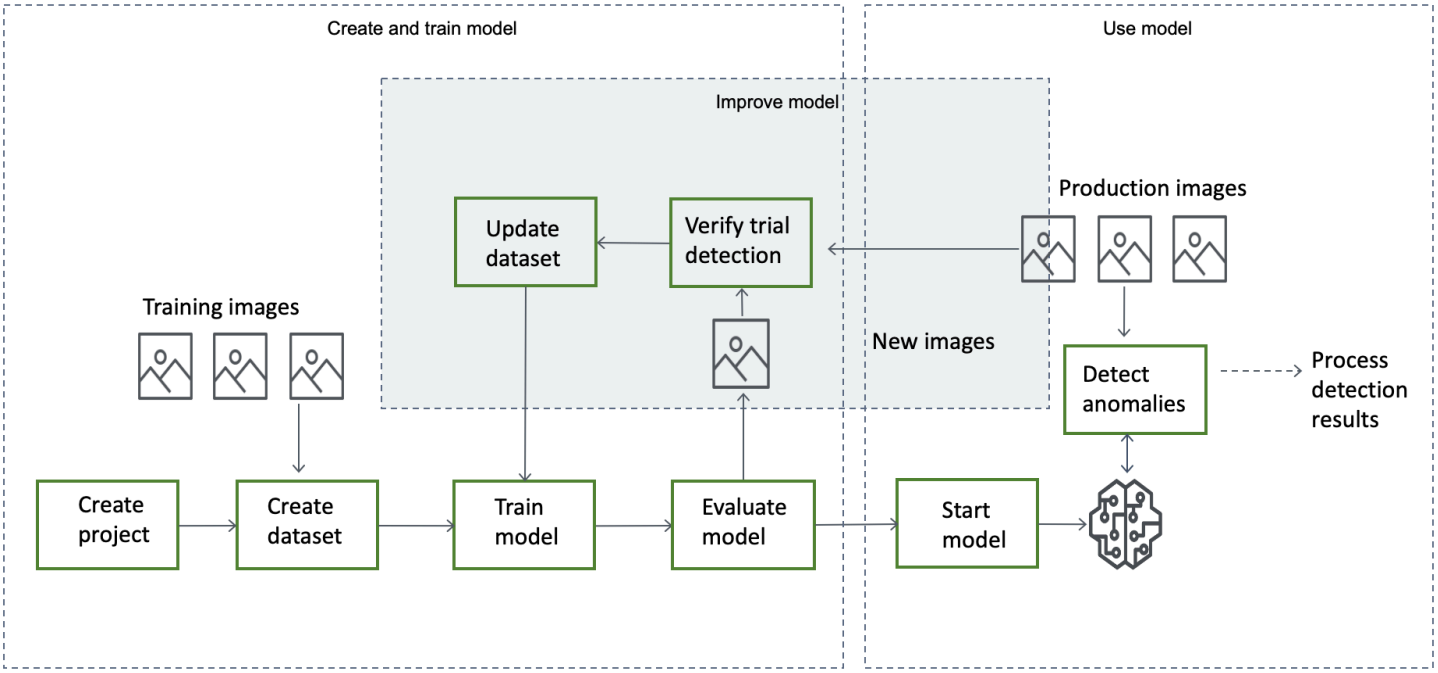
Memahami Amazon Lookout for Vision

Anda dapat menggunakan Amazon Lookout for Vision untuk menemukan cacat visual pada produk industri, akurat dan dalam skala besar, untuk tugas-tugas seperti:

- Mendeteksi bagian yang rusak - Spot kerusakan pada kualitas, warna, dan bentuk permukaan produk selama proses fabrikasi dan perakitan.
- Mengidentifikasi komponen yang hilang - Tentukan komponen yang hilang berdasarkan ketidakhadiran, keberadaan, atau penempatan objek. Misalnya, kapasitor yang hilang pada papan sirkuit tercetak.
- Mengungkap masalah proses - Mendeteksi cacat dengan pola berulang, seperti goresan berulang di tempat yang sama pada wafer silikon.

Dengan Lookout for Vision Anda membuat model visi komputer yang memprediksi adanya anomali dalam gambar. Anda memberikan gambar yang digunakan Amazon Lookout for Vision untuk melatih dan menguji model Anda. Amazon Lookout for Vision menyediakan metrik yang dapat digunakan untuk mengevaluasi dan meningkatkan model terlatih Anda. Anda dapat meng-host model terlatih diAWS cloud atau Anda dapat menerapkan model ke perangkat edge. Operasi API sederhana mengembalikan prediksi yang dibuat model Anda.

Alur kerja umum untuk membuat, mengevaluasi, dan menggunakan model adalah sebagai berikut:



Topik

Pilih tipe model Anda

Sebelum Anda dapat membuat model, Anda harus memutuskan jenis model yang Anda inginkan. Anda dapat membuat dua jenis model, klasifikasi gambar dan segmentasi gambar. Anda menentukan jenis model mana yang akan dibuat berdasarkan kasus penggunaan Anda.

Model klasifikasi gambar

Jika Anda hanya perlu mengetahui apakah gambar mengandung anomali, tetapi tidak perlu mengetahui lokasinya, buat model klasifikasi gambar. Model klasifikasi gambar membuat prediksi apakah gambar mengandung anomali. Prediksi tersebut mencakup kepercayaan model terhadap keakuratan prediksi. Model tidak memberikan informasi apa pun tentang lokasi anomali apa pun yang ditemukan pada gambar.

Model segmentasi gambar

Jika Anda perlu mengetahui lokasi anomali, seperti lokasi goresan, buat model segmentasi gambar. Model Amazon Lookout for Vision menggunakan segmentasi semantik untuk mengidentifikasi piksel pada gambar tempat terdapat jenis anomali (seperti goresan atau bagian yang hilang).

> **Note**
>
> Model segmentasi semantik menempatkan berbagai jenis anomali. Ini tidak memberikan informasi contoh untuk anomali individu. Misalnya, jika gambar berisi dua penyok, Lookout for Vision mengembalikan informasi tentang kedua penyok dalam satu entitas yang mewakili jenis anomali penyok.

Model segmentasi Amazon Lookout for Vision memprediksi hal berikut:

Klasifikasi

Model mengembalikan klasifikasi untuk gambar yang dianalisis (normal/anomali), yang mencakup kepercayaan model dalam prediksi. Informasi klasifikasi dihitung secara terpisah dari informasi segmentasi dan Anda tidak boleh mengasumsikan hubungan di antara mereka.

Segmentasi

Model mengembalikan topeng gambar yang menandai piksel di mana anomali terjadi pada gambar. Berbagai jenis anomali dikodekan warna sesuai dengan warna yang ditetapkan ke label anomali dalam dataset. Label anomali mewakili jenis anomali. Misalnya, topeng biru pada gambar berikut menandai lokasi jenis anomali goresan yang ditemukan pada mobil.

Model mengembalikan kode warna untuk setiap label anomali di mask. Model ini juga mengembalikan persentase penutup gambar yang dimiliki label anomali.

Dengan model segmentasi Lookout for Vision, Anda dapat menggunakan berbagai kriteria untuk menganalisis hasil analisis dari model. Misalnya:

- Lokasi anomali - Jika Anda perlu mengetahui lokasi anomali, gunakan informasi segmentasi untuk melihat topeng yang mencakup anomali.
- Jenis anomali - Gunakan informasi segmentasi untuk memutuskan apakah gambar berisi lebih dari jumlah jenis anomali yang dapat diterima.
- Area cakupan - Gunakan informasi segmentasi untuk memutuskan apakah jenis anomali mencakup lebih dari area gambar yang dapat diterima.
- Klasifikasi gambar - Jika Anda tidak perlu mengetahui lokasi anomali, gunakan informasi klasifikasi untuk menentukan apakah gambar mengandung anomali.

Untuk kode sampel, lihat Mendeteksi anomali dalam gambar.

Setelah Anda memutuskan jenis model yang Anda inginkan, Anda membuat proyek dan dataset untuk mengelola model Anda. Menggunakan Label, Anda dapat mengklasifikasikan gambar sebagai normal atau anomali. Label juga mengidentifikasi informasi segmentasi seperti masker dan jenis anomali. Bagaimana Anda memberi label pada gambar dalam kumpulan data Anda menentukan jenis model yang dibuat oleh Lookout for Vision untuk Anda.

Pelabelan model segmentasi gambar lebih kompleks daripada memberi label pada model klasifikasi gambar. Untuk melatih model segmentasi, Anda harus mengklasifikasikan gambar pelatihan sebagai normal atau anomali. Anda juga harus mendefinisikan topeng anomali dan jenis anomali untuk setiap gambar anomali. Model klasifikasi hanya mengharuskan Anda mengidentifikasi gambar pelatihan sebagai normal atau anomali.

Buat model Anda

Langkah-langkah untuk membuat model adalah membuat proyek, membuat dataset, dan melatih model adalah sebagai berikut:

Membuat proyek

Buat proyek untuk mengelola kumpulan data dan model yang Anda buat. Sebuah proyek harus digunakan untuk kasus penggunaan tunggal, seperti mendeteksi anomali dalam satu jenis bagian mesin.

Anda dapat menggunakan dasbor untuk mendapatkan gambaran umum proyek Anda. Untuk informasi selengkapnya, lihat Menggunakan dasbor Amazon Lookout for Vision.

Informasi lebih lanjut: Buat proyek Anda.

Membuat set data

Untuk melatih model Amazon Lookout for Vision membutuhkan gambar objek normal dan anomali untuk kasus penggunaan Anda. Anda menyediakan gambar-gambar ini dalam dataset.

Dataset adalah sekumpulan gambar dan label yang menggambarkan gambar-gambar tersebut. Gambar harus mewakili satu jenis objek di mana anomali dapat terjadi. Untuk informasi selengkapnya, lihat Mempersiapkan gambar untuk dataset.

Dengan Amazon Lookout for Vision, Anda dapat memiliki proyek yang menggunakan satu set data, atau proyek yang memiliki kumpulan data pelatihan dan pengujian terpisah. Sebaiknya gunakan

proyek kumpulan data tunggal kecuali Anda menginginkan kontrol yang lebih baik atas pelatihan, pengujian, dan penyetelan kinerja.

Anda membuat dataset dengan mengimpor gambar. Tergantung pada bagaimana Anda mengimpor gambar, gambar mungkin juga diberi label. Jika tidak, Anda menggunakan konsol untuk memberi label pada gambar.

Mengimpor gambar

Jika membuat set data dengan Lookout for Vision console, Anda dapat mengimpor gambar dengan salah satu cara berikut:

- Impor gambar dari komputer lokal Anda. Gambar tidak diberi label.
- Impor gambar dari bucket S3. Amazon Lookout for Vision dapat mengklasifikasikan gambar menggunakan nama folder yang berisi gambar. Gunakan`normal` untuk gambar normal. Gunakan`anomaly` untuk gambar anomali. Anda tidak dapat menetapkan label segmentasi secara otomatis.
- Impor file manifes Amazon SageMaker Ground Truth. Gambar dalam file manifes diberi label. Anda dapat membuat dan mengimpor file manifes Anda sendiri. Jika Anda memiliki banyak gambar, pertimbangkan untuk menggunakan layanan pelabelan SageMaker Ground Truth. Anda kemudian mengimpor file manifes keluaran dari pekerjaan Amazon SageMaker Ground Truth.

Pelabelan gambar

Label menggambarkan gambar dalam dataset. Label menentukan apakah gambar normal atau anomali (klasifikasi). Label juga menggambarkan lokasi anomali pada gambar (segmentasi).

Jika gambar Anda tidak diberi label, Anda dapat menggunakan konsol untuk memberi label.

Label yang Anda tetapkan ke gambar dalam kumpulan data Anda menentukan jenis model yang dibuat oleh Lookout for Vision:

Klasifikasi gambar

Untuk membuat model klasifikasi gambar, gunakan konsol Lookout for Vision untuk mengklasifikasikan gambar dalam kumpulan data seperti biasa atau anomali.

Anda juga dapat menggunakan`CreateDataset` operasi untuk membuat kumpulan data dari file manifes yang menyertakan informasi klasifikasi.

Segmentasi gambar

Untuk membuat model segmentasi gambar, gunakan konsol Lookout for Vision untuk mengklasifikasikan gambar dalam kumpulan data seperti biasa atau anomali. Anda juga menentukan topeng piksel untuk area anomali pada gambar (jika ada) serta label anomali untuk masker anomali individu.

Anda juga dapat menggunakan`CreateDataset` operasi untuk membuat kumpulan data dari file manifes yang menyertakan informasi segmentasi dan klasifikasi.

Jika proyek Anda memiliki set data pelatihan dan pengujian terpisah, Lookout for Vision menggunakan set data pelatihan untuk mempelajari dan menentukan jenis model. Anda harus memberi label pada gambar dalam kumpulan data pengujian Anda dengan cara yang sama.

Informasi lebih lanjut: Membuat dataset Anda.

Latih model Anda

Pelatihan menciptakan model dan melatihnya untuk memprediksi adanya anomali dalam gambar. Versi baru model Anda dibuat setiap kali Anda berlatih.

Pada awal pelatihan, Amazon Lookout for Vision memilih algoritme yang paling cocok untuk melatih model Anda. Model ini dilatih dan kemudian diuji. DiMemulai dengan Amazon Lookout for Vision, Anda melatih proyek kumpulan data tunggal, kumpulan data dibagi secara internal untuk membuat set data pelatihan dan kumpulan data pengujian. Anda juga dapat membuat proyek yang memiliki kumpulan data pelatihan dan pengujian terpisah. Dalam konfigurasi ini, Amazon Lookout for Vision melatih model Anda dengan set data pelatihan dan menguji model dengan set data pengujian.

> **Important**
>
> Anda dikenai biaya untuk jumlah waktu yang diperlukan untuk melatih model Anda dengan sukses.

Informasi lebih lanjut: Latih model Anda.

Evaluasi model Anda

Evaluasi kinerja model Anda dengan menggunakan metrik kinerja yang dibuat selama pengujian.

Dengan menggunakan metrik kinerja, Anda dapat lebih memahami kinerja model terlatih Anda, dan memutuskan apakah Anda siap menggunakannya dalam produksi.

Informasi lebih lanjut: Meningkatkan model Anda.

Jika metrik kinerja menunjukkan bahwa perbaikan diperlukan, Anda dapat menambahkan lebih banyak data latihan dengan menjalankan tugas deteksi uji coba dengan gambar baru. Setelah tugas selesai, Anda dapat memverifikasi hasil dan menambahkan gambar terverifikasi ke set data latihan Anda. Atau, Anda dapat menambahkan gambar pelatihan baru secara langsung ke set data. Selanjutnya, Anda melatih ulang model Anda dan memeriksa kembali metrik kinerja.

Informasi lebih lanjut: Memverifikasi model Anda dengan tugas deteksi uji coba.

Gunakan model Anda

Sebelum Anda dapat menggunakan model Anda diAWS cloud, Anda memulai model dengan StartModeloperasi. Anda bisa mendapatkan perintah`StartModel` CLI untuk model Anda dari konsol.

Informasi lebih lanjut: Mulai model Anda.

Model Amazon Lookout for Vision yang terlatih memprediksi apakah gambar input berisi konten normal atau anomali. Jika model Anda adalah model segmentasi, prediksi mencakup topeng anomali yang menandai piksel tempat anomali ditemukan.

Untuk membuat prediksi dengan model Anda, panggil DetectAnomaliesoperasi dan berikan gambar input dari komputer lokal Anda. Anda bisa mendapatkan perintah CLI yang memanggil`DetectAnomalies` dari konsol.

Informasi lebih lanjut: Mendeteksi anomali dalam gambar.

> **Important**
>
> Anda dikenai biaya untuk menjalankan model Anda.

Jika Anda tidak lagi menggunakan model Anda, gunakan StopModeloperasi untuk menghentikan model. Anda bisa mendapatkan perintah CLI dari konsol.

Informasi lebih lanjut: Hentikan model Anda.

Gunakan model Anda pada perangkat edge

Anda dapat menggunakan model Lookout for Vision pada perangkatAWS IoT Greengrass Version 2 inti.

Informasi lebih lanjut: Menggunakan model Amazon Lookout for Vision pada perangkat edge.

Gunakan dasbor Anda

Anda dapat menggunakan dasbor untuk mendapatkan ikhtisar semua proyek dan informasi ikhtisar Anda untuk masing-masing proyek.

Informasi lebih lanjut: Gunakan dasbor Anda.

Memulai dengan Amazon Lookout for Vision

Sebelum memulai instruksi Memulai, kami sarankan Anda membacaMemahami Amazon Lookout for Vision.

Petunjuk Memulai menunjukkan kepada Anda cara menggunakan membuat model segmentasi gambar contoh. Jika Anda ingin membuat contoh model klasifikasi gambar, lihatSet data klasifikasi gambar.

Jika Anda ingin mencoba model contoh dengan cepat, kami menyediakan contoh gambar pelatihan dan gambar topeng. Kami juga menyediakan skrip Python yang membuat file manifes segmentasi gambar. Anda menggunakan file manifes untuk membuat kumpulan data untuk proyek Anda dan Anda tidak perlu memberi label pada gambar dalam kumpulan data. Saat Anda membuat model dengan gambar Anda sendiri, Anda harus memberi label pada gambar dalam kumpulan data. Untuk informasi selengkapnya, lihat Membuat dataset Anda.

Gambar yang kami sediakan adalah cookie normal dan anomali. Cookie anomali memiliki celah di seluruh bentuk cookie. Model yang Anda latih dengan gambar memprediksi klasifikasi (normal atau anomali) dan menemukan area (topeng) retakan dalam cookie anomali, seperti yang ditunjukkan pada contoh berikut.

Topik

Langkah 1: Buat file manifes dan unggah gambar

Dalam prosedur ini, Anda mengkloning repositori dokumentasi Amazon Lookout for Vision ke komputer Anda. Anda kemudian menggunakan skrip Python (versi 3.7 atau lebih tinggi) untuk membuat file manifes dan mengunggah gambar pelatihan dan gambar topeng ke lokasi Amazon S3 yang Anda tentukan. Anda menggunakan file manifes untuk membuat model Anda. Kemudian, Anda menggunakan gambar uji di repositori lokal untuk mencoba model Anda.

Untuk membuat file manifes dan mengunggah gambar

1. Siapkan Amazon Lookout for Vision dengan mengikuti petunjuk di Pengaturan Amazon Lookout for Vision. Pastikan untuk memasang AWSSDK for Python.
2. Di AWS Wilayah tempat Anda ingin menggunakan Lookout for Vision, buat bucket S3.
3. Di bucket Amazon S3, buat folder bernama`getting-started`.
4. Perhatikan Amazon S3 URI dan Amazon Resource name (ARN) untuk folder tersebut. Anda menggunakannya untuk mengatur izin dan menjalankan skrip.
5. Pastikan bahwa pengguna yang memanggil skrip memiliki izin untuk memanggil `s3:PutObject` operasi. Anda dapat menggunakan kebijakan berikut. Untuk menetapkan izin, lihatMenetapkan izin.

   ```
   {
    "Version": "2012-10-17",
    "Statement": [{
     "Sid": "Statement1",
     "Effect": "Allow",
     "Action": [
      "s3:PutObject"
     ],
     "Resource": [
      "arn:aws:s3::: ARN for S3 folder in step 4/*"
     ]
    }]
   }
   ```

6. Pastikan bahwa Anda memiliki profil lokal bernama `lookoutvision-access` dan bahwa pengguna profil memiliki izin dari langkah sebelumnya. Untuk informasi selengkapnya, lihat Menggunakan profil di komputer lokal Anda.
7. Unduh file zip, getting-started.zip. File zip berisi dataset memulai dan mengatur skrip.

8. Buka. `getting-started.zip`
9. Di prompt perintah, lakukan hal berikut:

   a. Arahkan ke `getting-started` folder.

   b. Jalankan perintah berikut untuk membuat file manifes dan mengunggah gambar pelatihan dan masker gambar ke jalur Amazon S3 yang Anda catat di langkah 4.

   ```
   python getting_started.py S3-URI-from-step-4
   ```

   c. Saat skrip selesai, perhatikan jalur ke `train.manifest` file yang ditampilkan skrip setelahnya`Create dataset using manifest file:`. Jalannya harus serupa dengan`s3://path to getting started folder/manifests/train.manifest`.

Langkah 2: Buat model

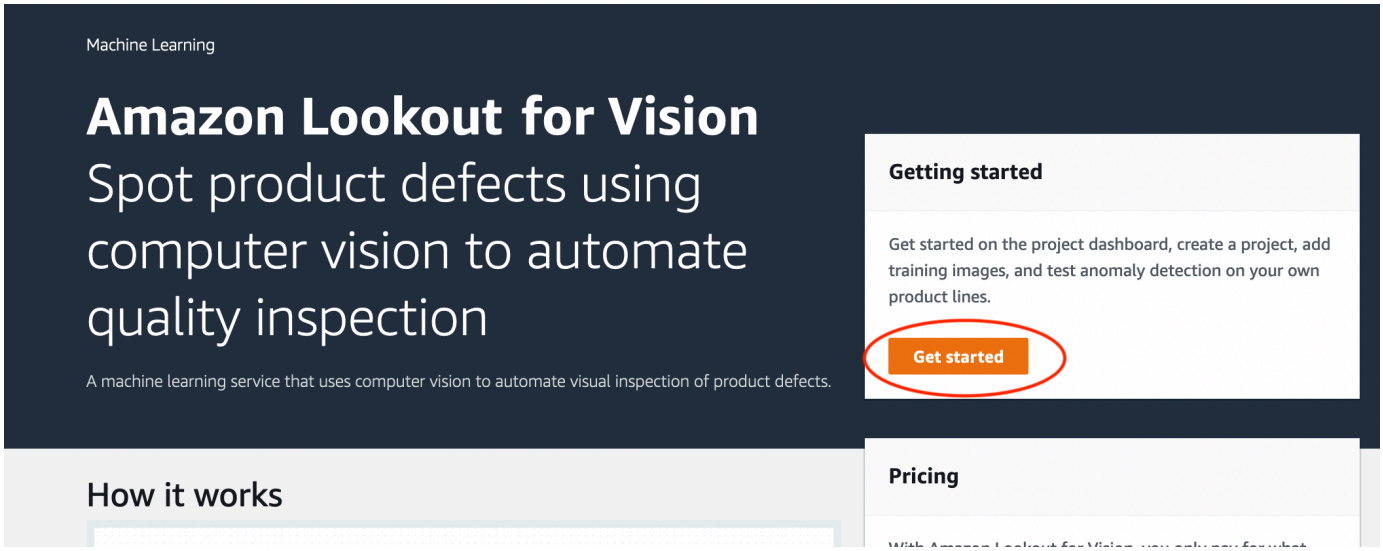
Dalam prosedur ini, Anda membuat proyek dan kumpulan data menggunakan image dan file manifes yang sebelumnya Anda unggah ke bucket Amazon S3 Anda. Anda kemudian membuat model dan melihat hasil evaluasi dari pelatihan model.

Karena Anda membuat kumpulan data dari file manifes memulai, Anda tidak perlu memberi label pada gambar kumpulan data. Saat Anda membuat kumpulan data dengan gambar Anda sendiri, Anda perlu memberi label pada gambar. Untuk informasi selengkapnya, lihat Pelabelan gambar.

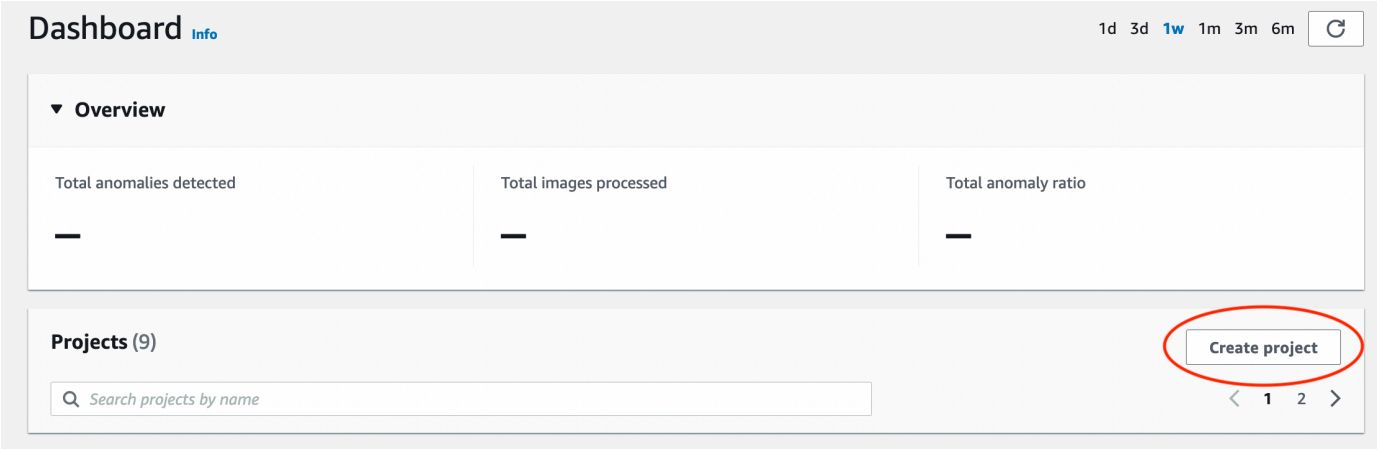
> **Important**
>
> Anda dikenai biaya untuk pelatihan model yang berhasil.

Untuk membuat model

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pastikan Anda berada di AWS Wilayah yang sama dengan Anda membuat bucket Amazon S3. Langkah 1: Buat file manifes dan unggah gambar Untuk mengubah Wilayah, pilih nama Wilayah yang ditampilkan saat ini ditampilkan di bilah navigasi. Kemudian pilih Wilayah yang ingin Anda alihkan.
3. Pilih Mulai.



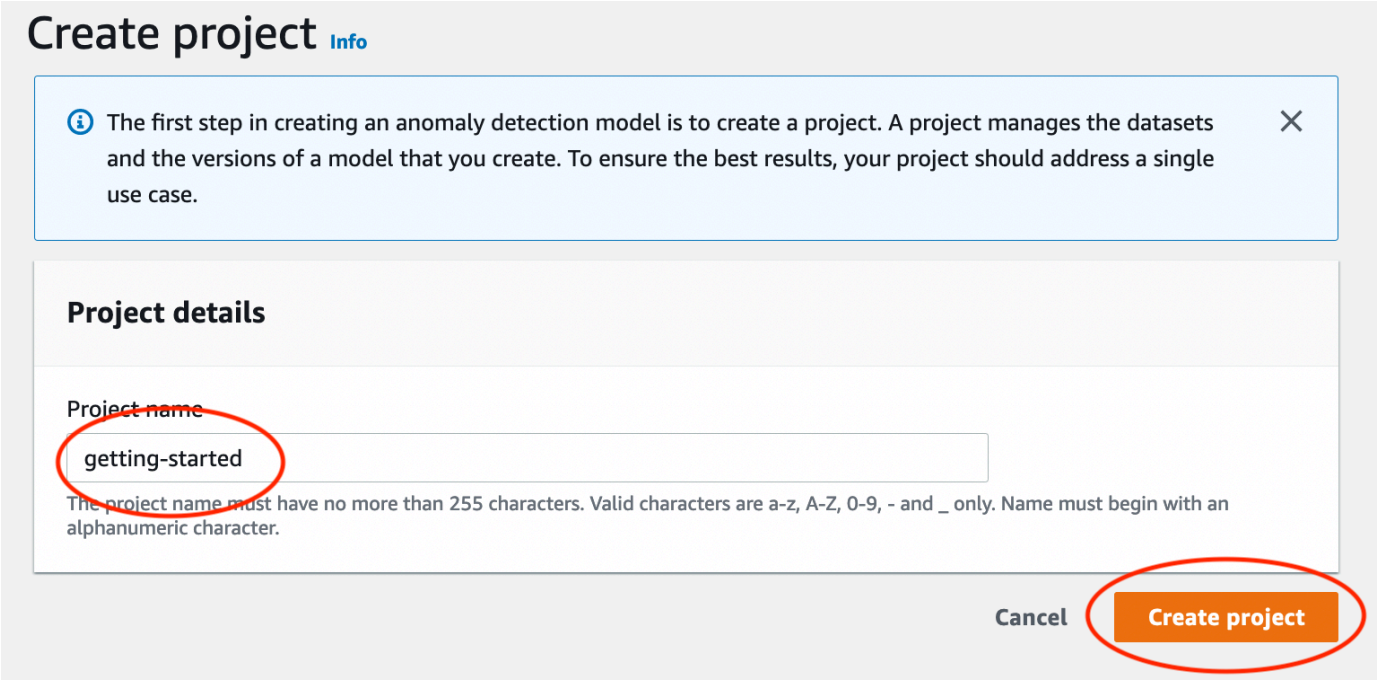
4. Di bagian Projects, pilih Create project.



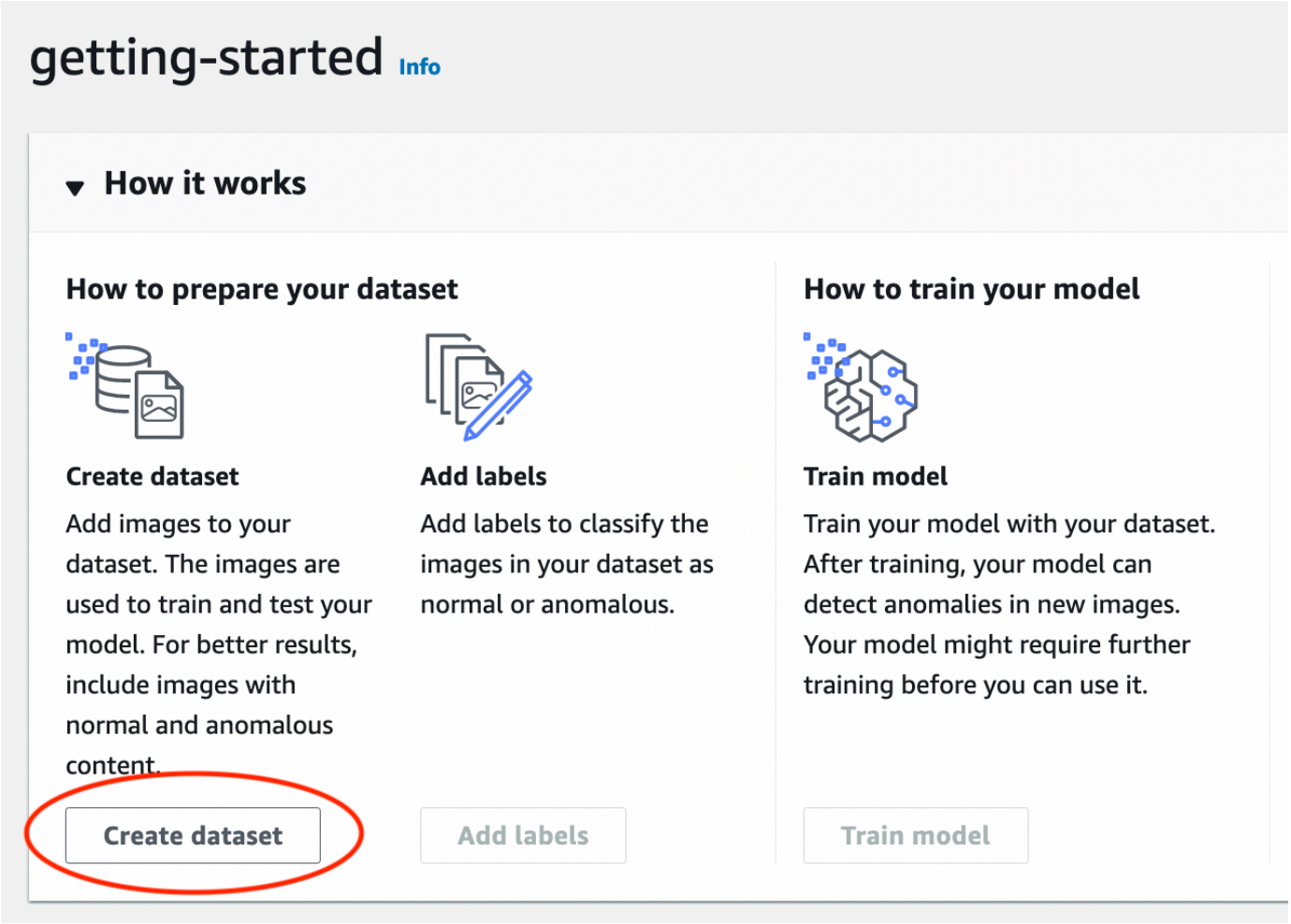
5. Di halaman Buat proyek, lakukan hal berikut:

   a. Di Nama proyek, masukkan`getting-started`.

   b. Pilih Buat proyek.



6. Pada halaman proyek, di bagian Cara kerjanya, pilih Create dataset.



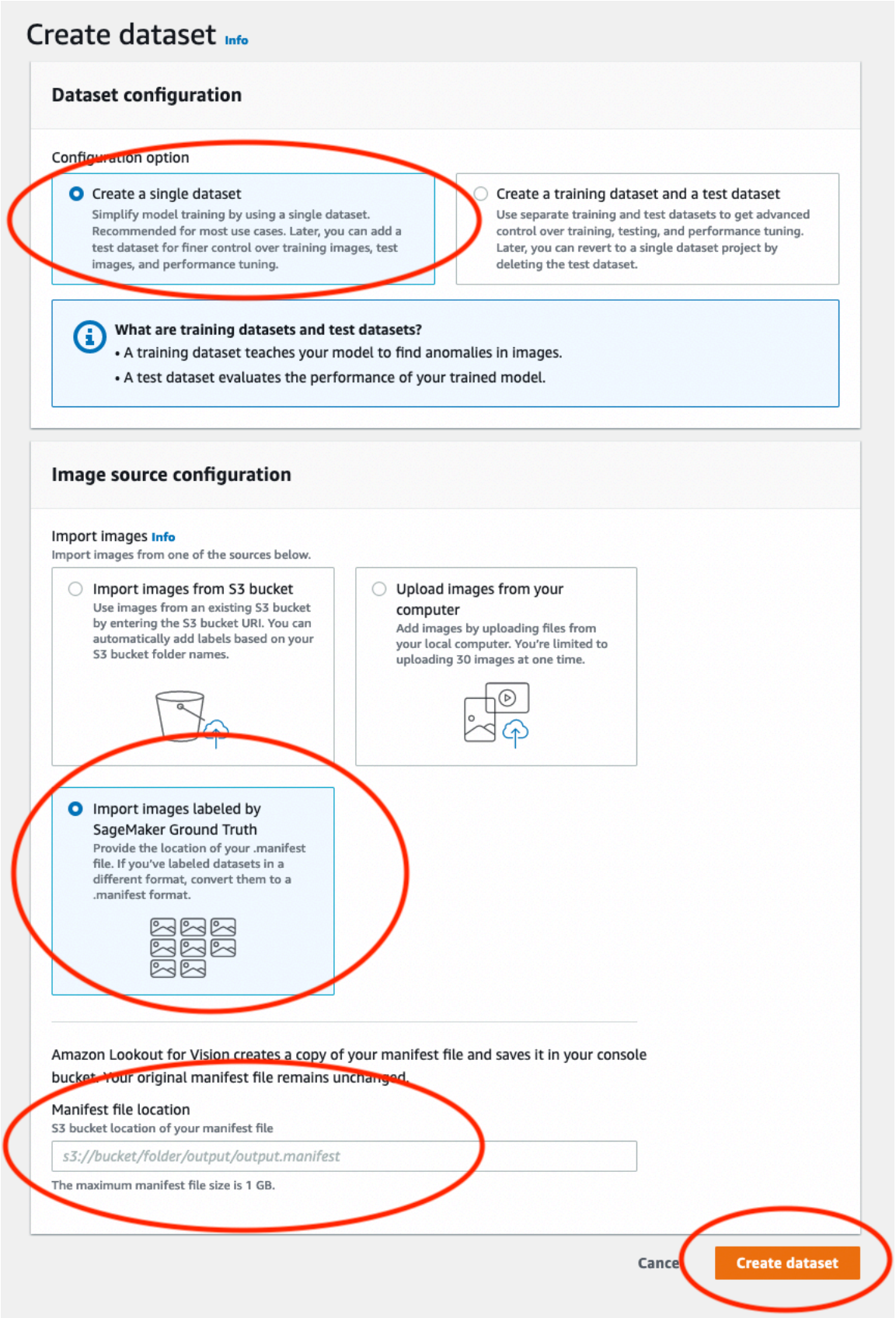
7. Di halaman Buat dataset, lakukan hal berikut:

   a. Pilih Buat set data tunggal.

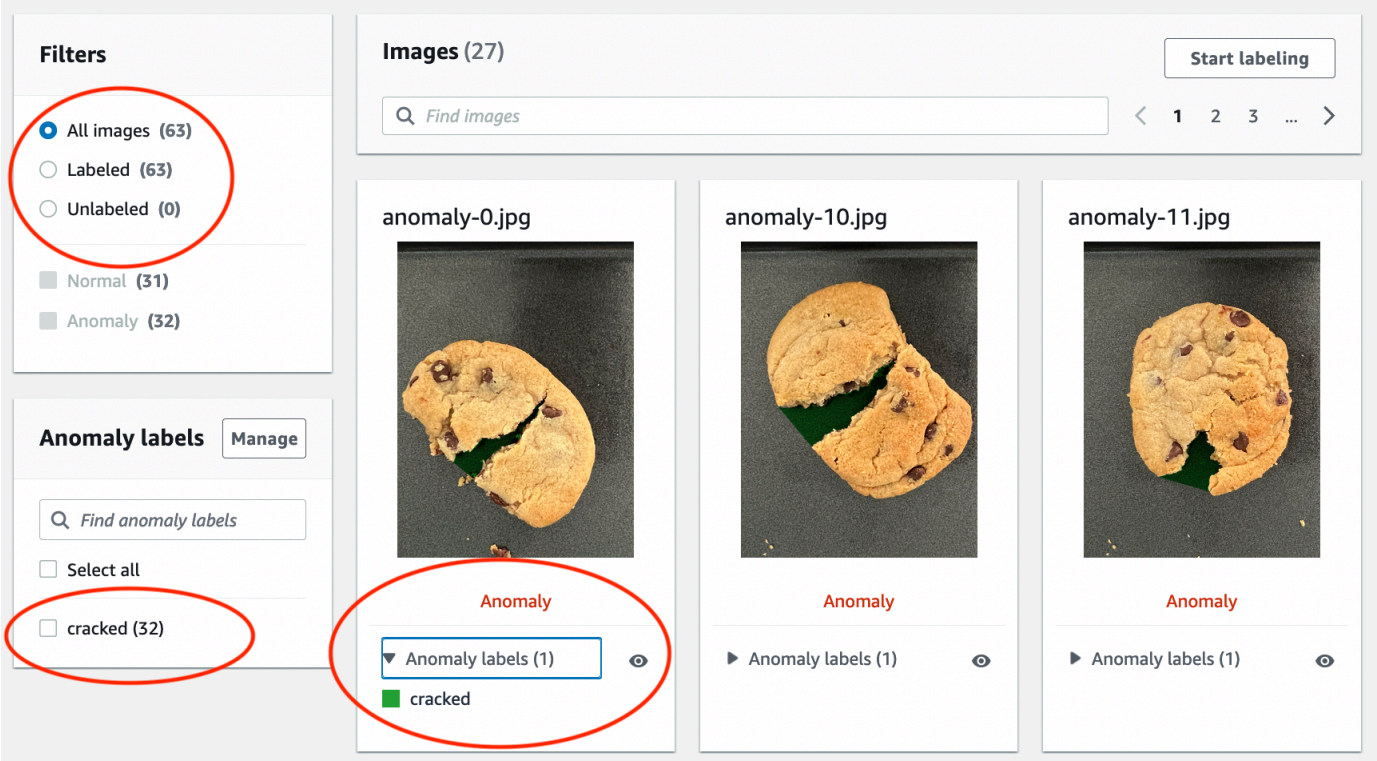
   b. Di bagian Konfigurasi sumber gambar, pilih Impor gambar yang diberi label oleh SageMaker Ground Truth.

   c. Untuk lokasi file.manifest, masukkan lokasi Amazon S3 dari file manifes yang Anda catat pada langkah 6.c. dari. Langkah 1: Buat file manifes dan unggah gambar Lokasi Amazon S3 harus serupa dengan `s3://path to getting started folder/manifests/train.manifest`

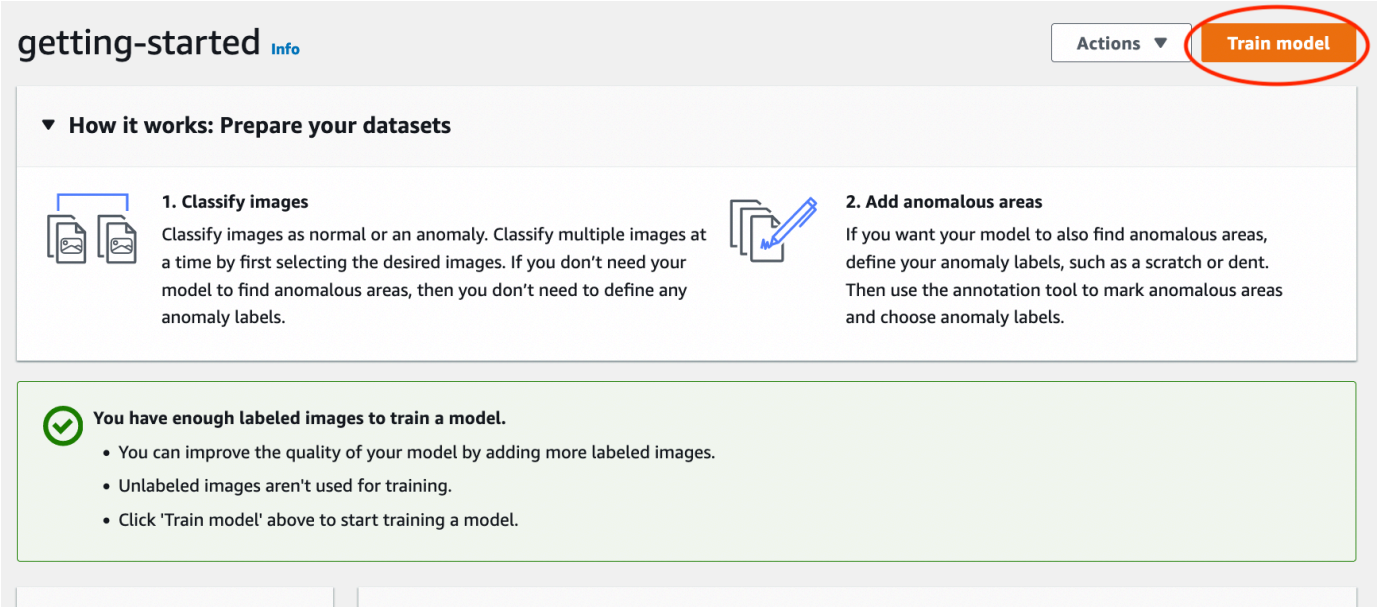
   d. Pilih Buat dataset.


Create dataset Info
Dataset configuration
Configuration option
Create a single dataset
Simplify model training by using a single dataset. Recommended for most use cases. Later, you can add a test dataset for finer control over training images, test images, and performance tuning.
Create a training dataset and a test dataset
Use separate training and test datasets to get advanced control over training, testing, and performance tuning. Later, you can revert to a single dataset project by deleting the test dataset.
What are training datasets and test datasets?
• A training dataset teaches your model to find anomalies in images.
• A test dataset evaluates the performance of your trained model.
Image source configuration
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 URI. 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.
Import images labeled by SageMaker Ground Truth
Provide the location of your .manifest file. If you've labeled datasets in a different format, convert them to a .manifest format.
Amazon Lookout for Vision creates a copy of your manifest file and saves it in your console bucket. Your original manifest file remains unchanged.
Manifest file location
S3 bucket location of your manifest file
s3://bucket/folder/output/output.manifest
The maximum manifest file size is 1 GB.
Cancel
Create dataset

8. Pada halaman detail proyek, di bagian Gambar, lihat gambar dataset. Anda dapat melihat informasi klasifikasi dan segmentasi gambar (label topeng dan anomali) untuk setiap gambar set data. Anda juga dapat mencari gambar, memfilter gambar dengan memberi label status (berlabel/tidak berlabel), atau memfilter gambar berdasarkan label anomali yang ditetapkan padanya.

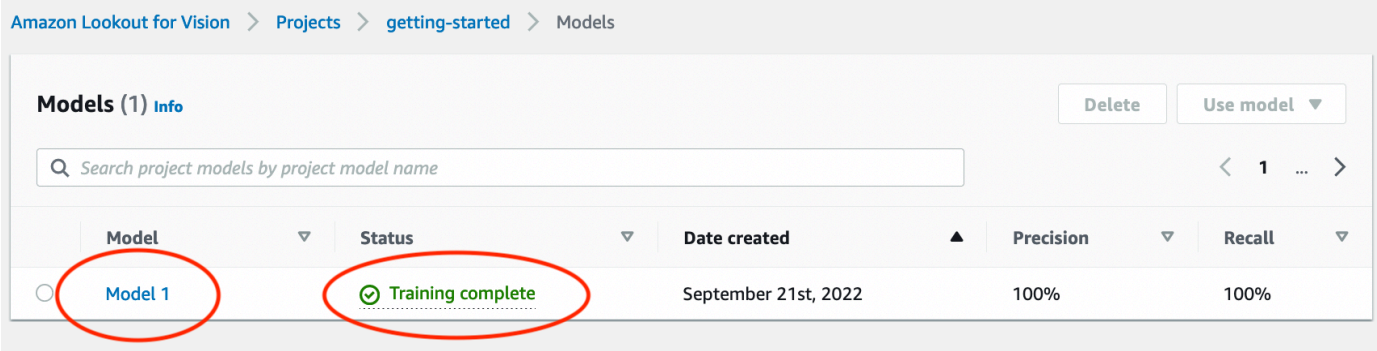


9. Pada halaman detail proyek, pilih model Kereta.



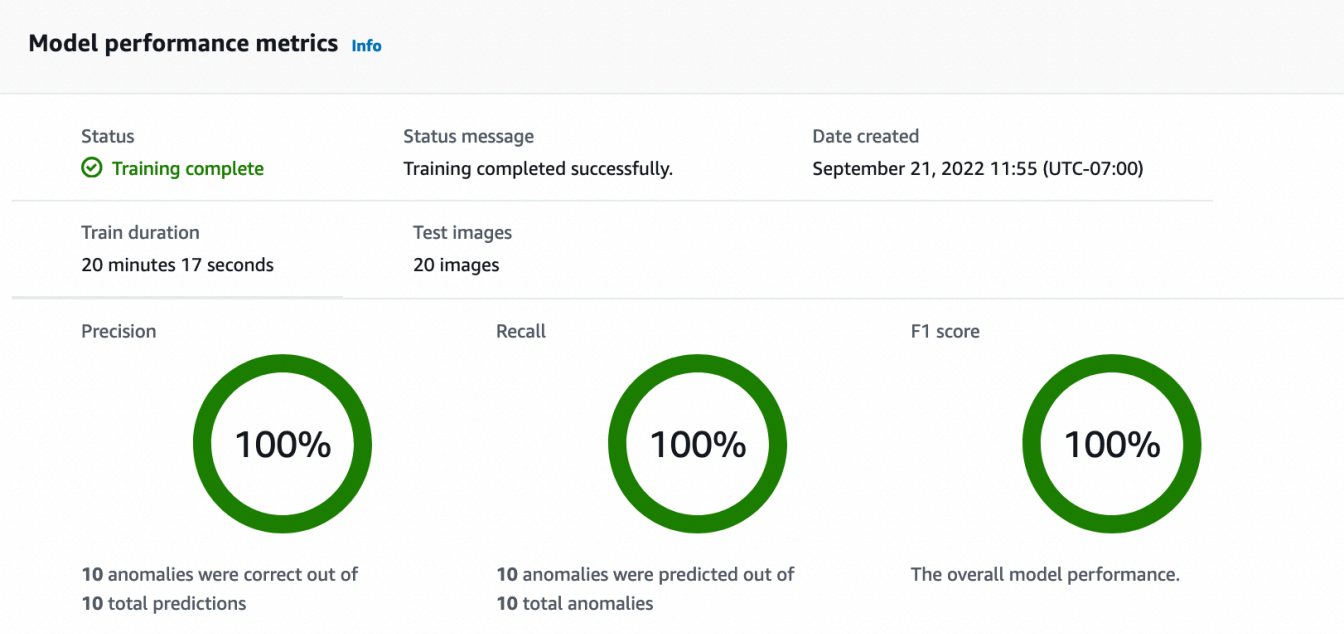
10. Pada halaman Detail model Kereta, pilih Model kereta.

11. Di Apakah Anda ingin melatih model Anda? kotak dialog, pilih model Kereta.
12. Di halaman Model proyek, Anda dapat melihat bahwa pelatihan telah dimulai. Periksa status saat ini dengan melihat kolom Status untuk versi model. Melatih model membutuhkan waktu setidaknya 30 menit untuk menyelesaikannya. Pelatihan telah berhasil diselesaikan ketika status berubah menjadi Pelatihan selesai.
13. Saat latihan selesai, pilih model Model 1 di halaman Model.

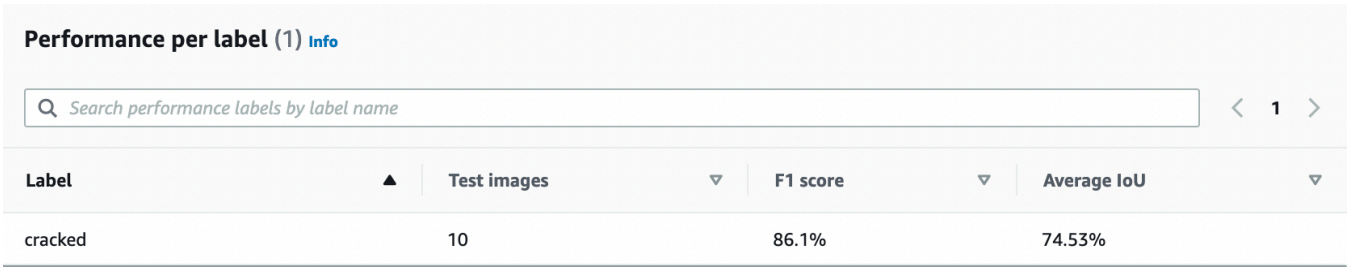


14. Di halaman detail model, lihat hasil evaluasi di tab Metrik kinerja. Ada metrik untuk berikut:

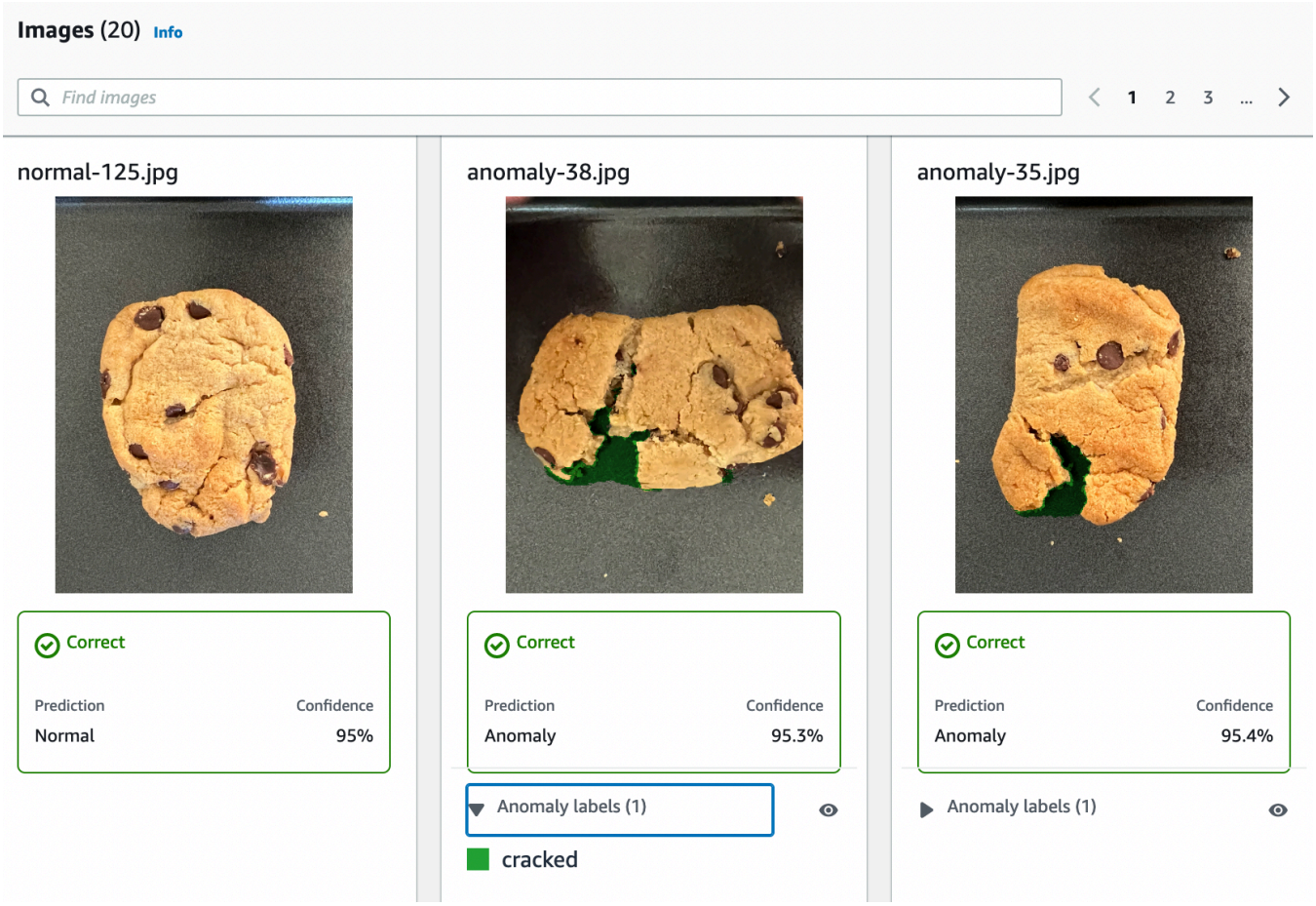
    - Metrik kinerja model keseluruhan (presisi, penarikan, dan skor F1) untuk prediksi klasifikasi yang dibuat oleh model.



    - Metrik kinerja untuk label anomali yang ditemukan dalam gambar uji (IoU rata-rata, skor F1)



- Prediksi untuk gambar uji (klasifikasi, masker segmentasi, dan label anomali)



Karena pelatihan model tidak deterministik, hasil evaluasi Anda mungkin berbeda dari hasil yang ditampilkan di halaman ini. Untuk informasi selengkapnya, lihat Meningkatkan model Amazon Lookout for Vision.

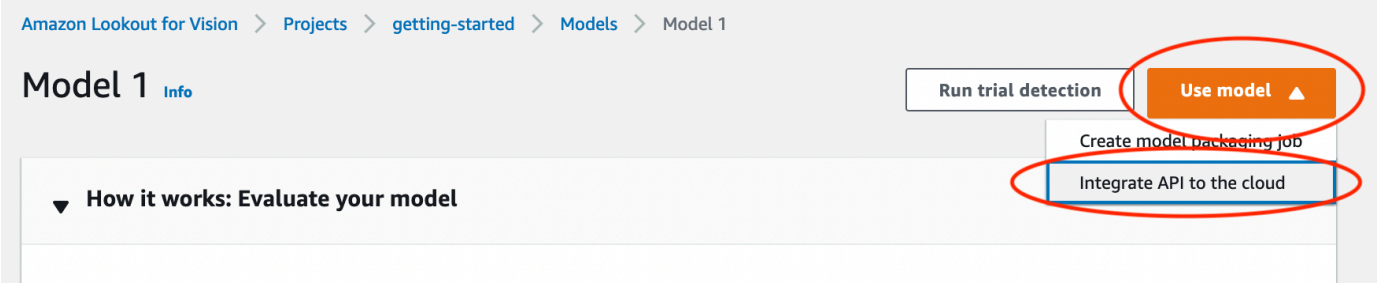
Langkah 3: Mulai model

Pada langkah ini, Anda mulai hosting model sehingga siap untuk menganalisis gambar. Untuk informasi selengkapnya, lihat Menjalankan model Amazon Lookout for Vision Anda yang terlatih.

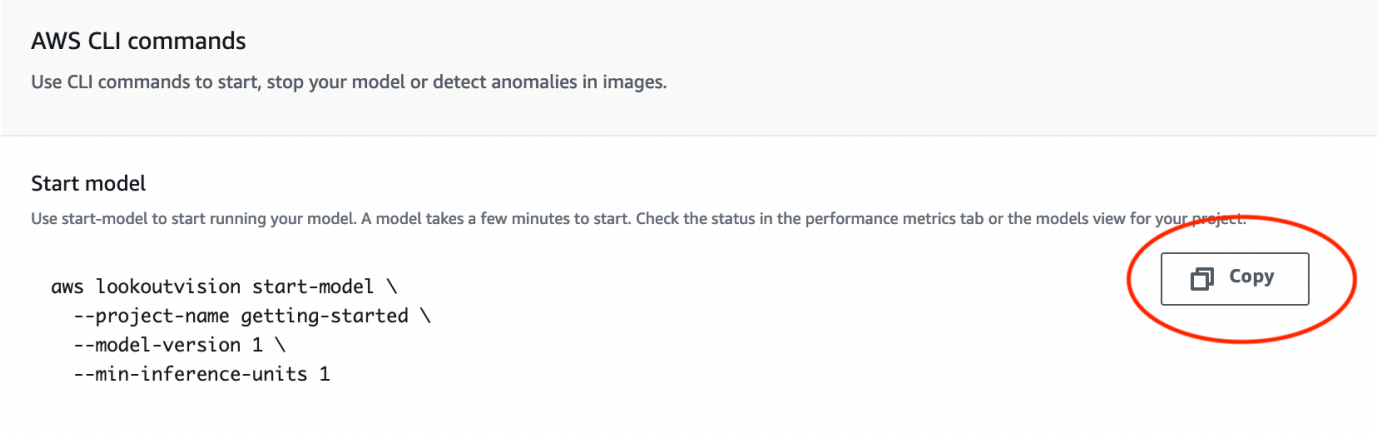
> **Note**
>
> Anda dikenai biaya untuk jumlah waktu yang digunakan untuk menjalankan model Anda. Anda menghentikan model Anda diLangkah 5: Hentikan model.

Untuk memulai model.

1. Pada halaman detail model, pilih Gunakan model, lalu pilih Integrasikan API ke cloud.



2. Di bagian AWS CLIperintah, salin `start-model` AWS CLI perintah.



3. Pastikan bahwa AWS CLI dikonfigurasi untuk berjalan di AWS Wilayah yang sama di mana Anda menggunakan konsol Amazon Lookout for Vision. Untuk mengubah AWS Wilayah yang AWS CLI digunakan, lihatInstal AWS SDKS.
4. Di prompt perintah, mulai model dengan memasukkan perintah, mulai model dengan memasukkan perintah, mulai model dengan memasukkan `start-model` perintah. Jika Anda menggunakan `lookoutvision` profil untuk mendapatkan kredensil, tambahkan parameter. `--profile lookoutvision-access` Misalnya:

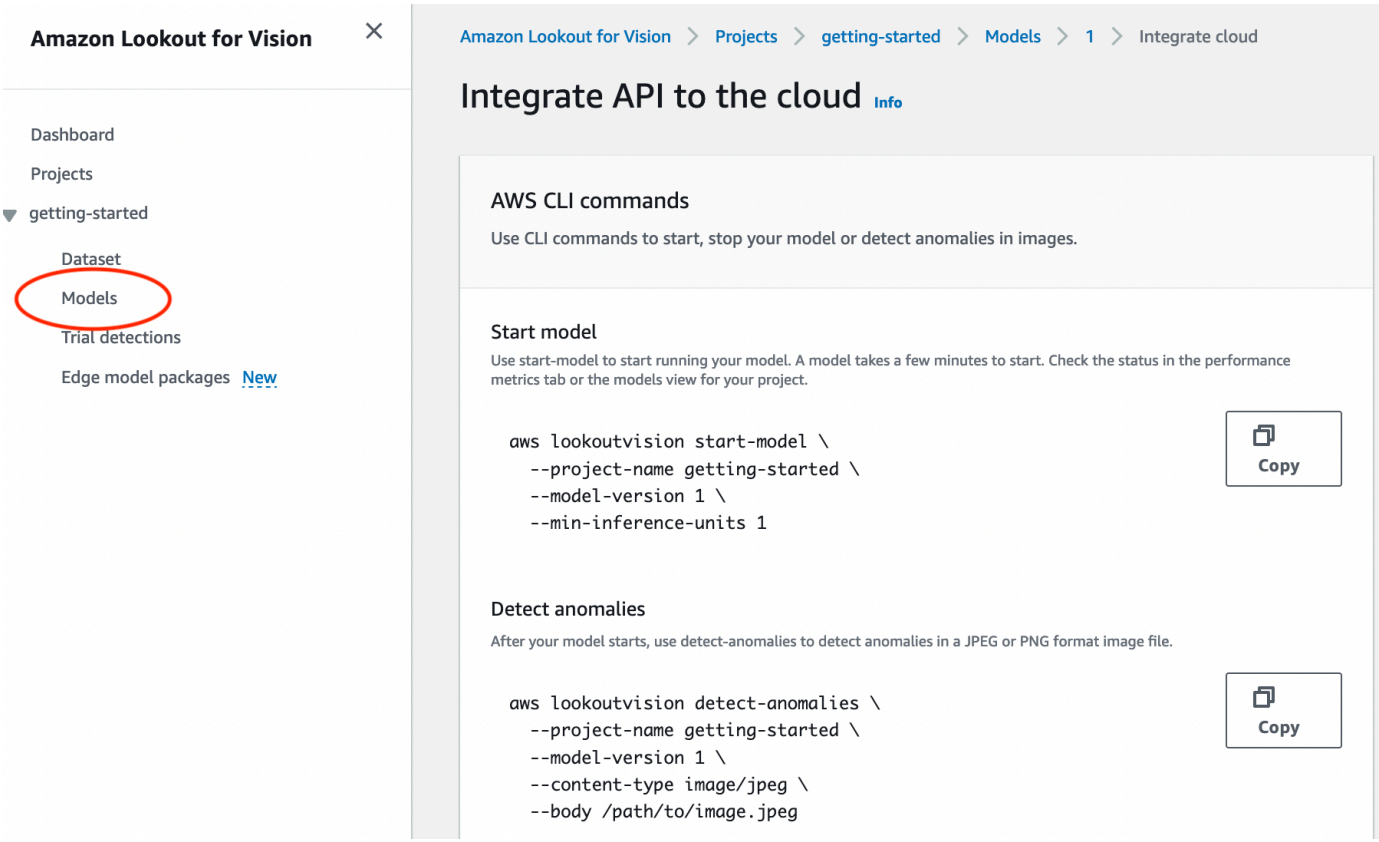
```
aws lookoutvision start-model \
  --project-name getting-started \
  --model-version 1 \
```

```
  --min-inference-units 1 \
  --profile lookoutvision-access
```

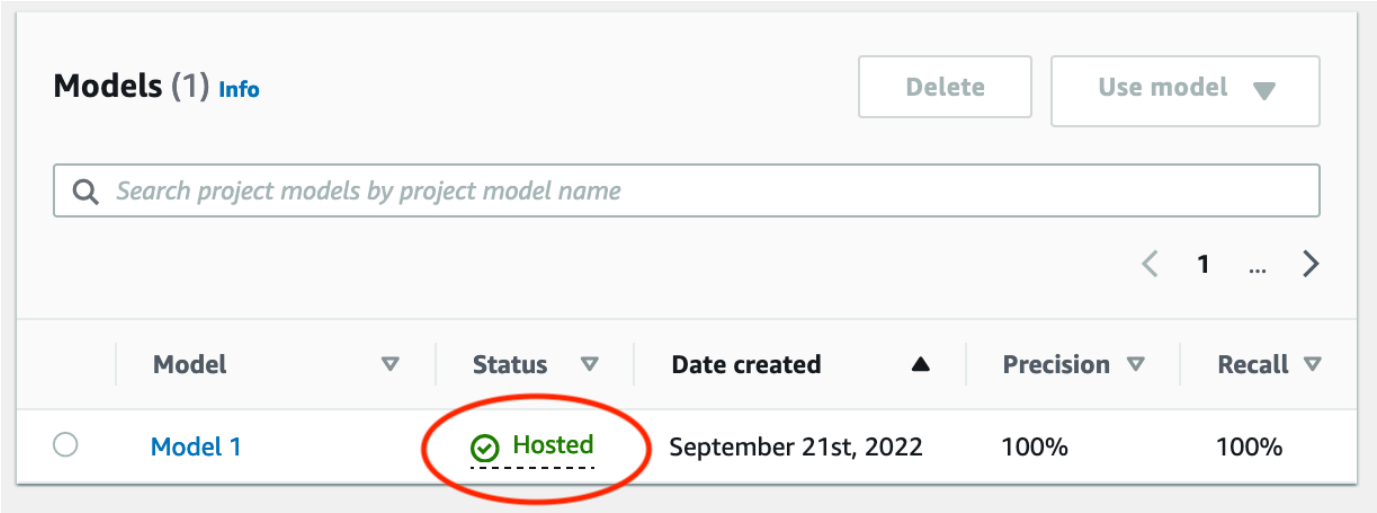
Jika panggilan berhasil, output berikut akan ditampilkan:

```
{
    "Status": "STARTING_HOSTING"
}
```

5. Di konsol, pilih Model dalam panel navigasi.



6. Tunggu sampai status model (Model 1) di kolom Status menampilkan Hosted. Jika sebelumnya Anda telah melatih model dalam proyek, tunggu hingga versi model terbaru selesai.

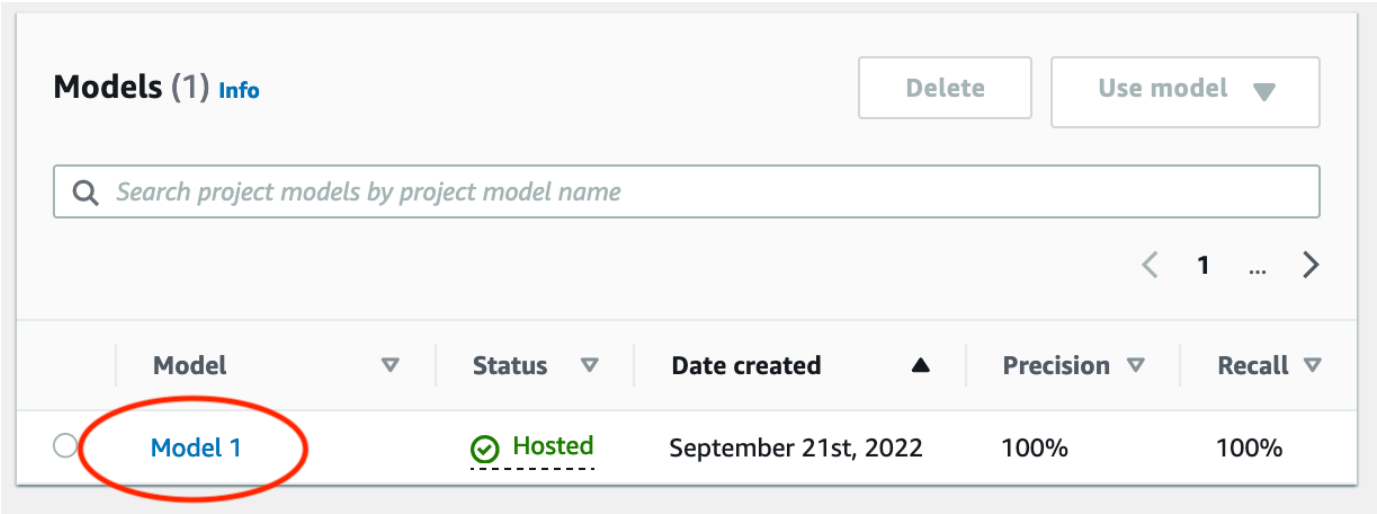


Langkah 4: Menganalisis gambar

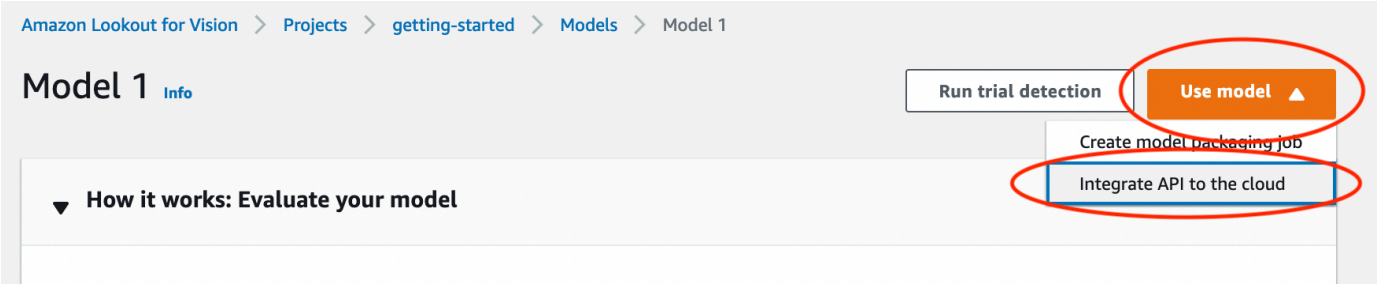
Pada langkah ini, Anda menganalisis gambar dengan model Anda. Kami menyediakan contoh gambar yang dapat Anda gunakan di `test-images` folder memulai di repositori dokumentasi Lookout for Vision di komputer Anda. Untuk informasi selengkapnya, lihat Mendeteksi anomali dalam gambar.

Untuk menganalisis gambar

1. Pada halaman Model, pilih model Model 1.



2. Pada halaman detail model, pilih Gunakan model, lalu pilih Integrasikan API ke cloud.



3. Di bagian AWS CLIperintah, salin `detect-anomalies` AWS CLI perintah.

4. Pada prompt perintah, analisis gambar anomali dengan memasukkan `detect-anomalies` perintah dari langkah sebelumnya. Untuk `--body` parameter, tentukan gambar anomali dari `test-images` folder memulai di komputer Anda. Jika Anda menggunakan `lookoutvision` profil untuk mendapatkan kredensil, tambahkan parameter. `--profile lookoutvision-access` Misalnya:

```
aws lookoutvision detect-anomalies \
  --project-name getting-started \
  --model-version 1 \
  --content-type image/jpeg \
  --body /path/to/test-images/test-anomaly-1.jpg \
  --profile lookoutvision-access
```

Outputnya akan serupa dengan yang berikut ini:

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.983975887298584,
        "Anomalies": [
            {
```

```
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.9818974137306213,
                    "Color": "#FFFFFF"
                }
            },
            {
                "Name": "cracked",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.018102575093507767,
                    "Color": "#23A436"
                }
            }
        ],
        "AnomalyMask": "iVBORw0KGgoAAAANSUhEUgAAAkAAAAMACA......"
    }
}
```

5. Di output, perhatikan hal berikut:

   - `IsAnomalous`adalah Boolean untuk klasifikasi diprediksi. `true`jika gambarnya anomali, jika tidak. `false`
   - `Confidence`adalah nilai float yang mewakili keyakinan yang dimiliki Amazon Lookout for Vision dalam prediksi. 0 adalah kepercayaan terendah, 1 adalah kepercayaan tertinggi.
   - `Anomalies`adalah daftar anomali yang ditemukan pada citra. `Name`adalah label anomali. `PixelAnomaly`termasuk luas persentase total anomali (`TotalPercentageArea`) dan warna (`Color`) untuk label anomali. Daftar ini juga mencakup anomali “latar belakang” yang mencakup area di luar anomali yang ditemukan pada gambar.
   - `AnomalyMask`adalah gambar topeng yang menunjukkan lokasi anomali pada gambar yang dianalisis.

   Anda dapat menggunakan informasi dalam respons untuk menampilkan campuran gambar yang telah dianalisis dan topeng anomali, seperti yang ditunjukkan pada contoh berikut. Untuk kode sampel, lihat Menampilkan informasi klasifikasi dan segmentasi.


Classification:
Prediction: Anomalous
Confidence: 99.9%
Segmentation:
Anomaly: cracked. Area: 6.2%

6. Pada prompt perintah, analisis gambar normal dari `test-images` folder memulai. Jika Anda menggunakan `lookoutvision` profil untuk mendapatkan kredensil, tambahkan parameter. `--profile lookoutvision-access` Misalnya:

```
aws lookoutvision detect-anomalies \
  --project-name getting-started \
  --model-version 1 \
  --content-type image/jpeg \
  --body /path/to/test-images/test-normal-1.jpg \
  --profile lookoutvision-access
```

Outputnya akan serupa dengan yang berikut ini:

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": false,
        "Confidence": 0.9916400909423828,
        "Anomalies": [
            {
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 1.0,
                    "Color": "#FFFFFF"
                }
            }
        ],
        "AnomalyMask": "iVBORw0KGgoAAAANSUhEUgAAAkAAAA....."
    }
}
```

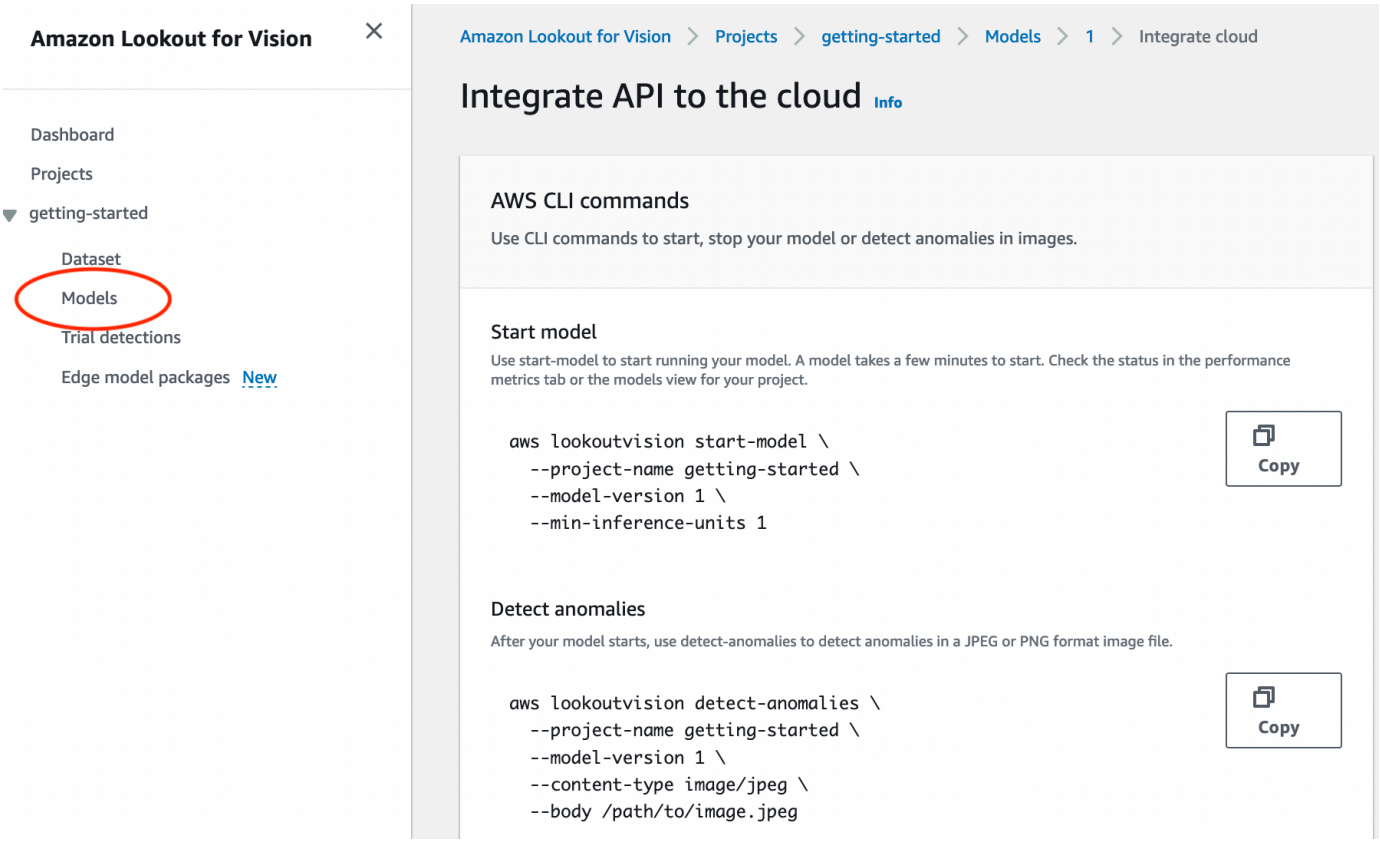
7. Dalam output, perhatikan bahwa `false` nilai untuk `IsAnomalous` mengklasifikasikan gambar sebagai tidak memiliki anomali. Gunakan `Confidence` untuk membantu menentukan kepercayaan diri Anda dalam klasifikasi. Juga, `Anomalies` array hanya memiliki label `background` anomali.

Langkah 5: Hentikan model

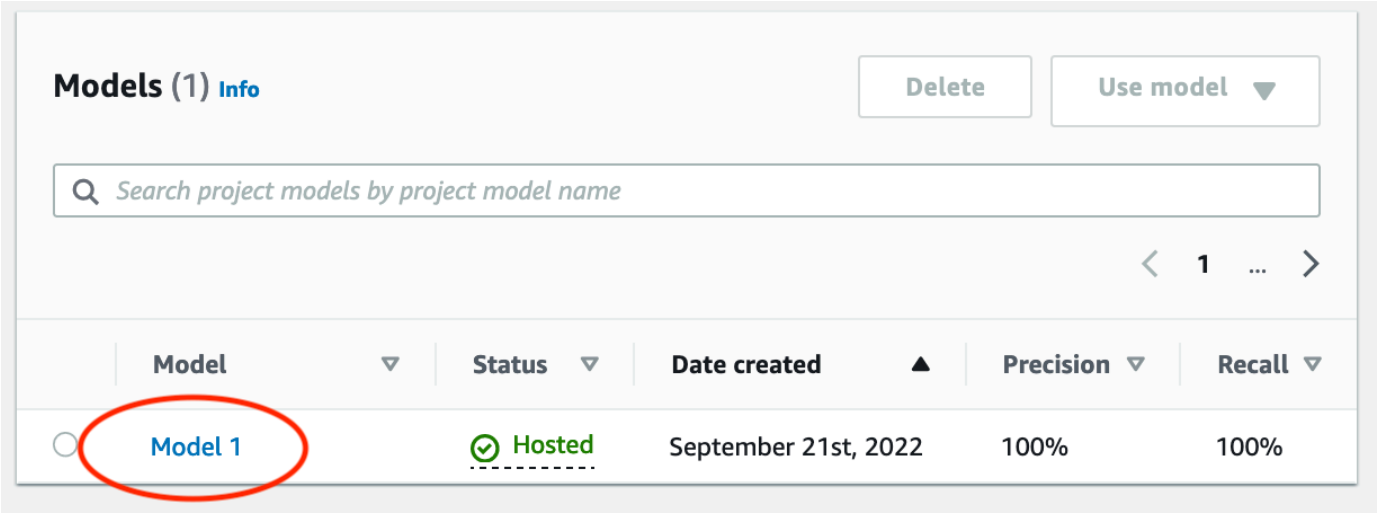
Pada langkah ini, Anda berhenti hosting model. Anda dikenai biaya untuk jumlah waktu yang digunakan untuk menjalankan model Anda. Jika Anda tidak menggunakan model, Anda harus menghentikannya. Anda dapat memulai ulang model saat Anda membutuhkannya. Untuk informasi selengkapnya, lihat Memulai model Amazon Lookout for Vision Anda.

Untuk menghentikan model.

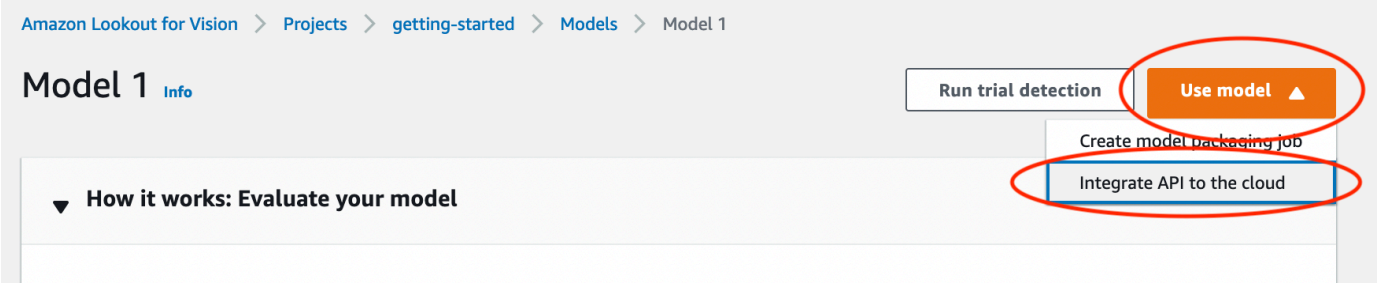
1. Pilih Model di panel navigasi.



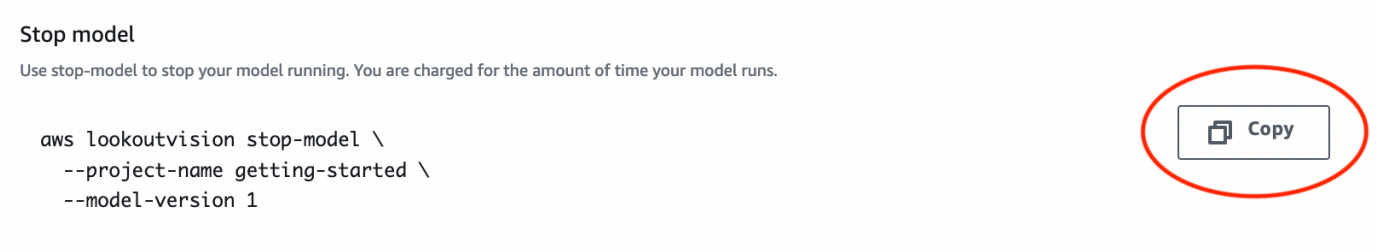
2. Di halaman Model, pilih model Model 1.



3. Pada halaman detail model, pilih Gunakan model, lalu pilih Integrasikan API ke cloud.



4. Di bagian AWS CLIperintah, salin `stop-model` AWS CLI perintah.



5. Pada prompt perintah, hentikan model dengan memasukkan `stop-model` AWS CLI perintah dari langkah sebelumnya. Jika Anda menggunakan `lookoutvision` profil untuk mendapatkan kredensil, tambahkan parameter. `--profile lookoutvision-access` Misalnya:

```
aws lookoutvision stop-model \
  --project-name getting-started \
  --model-version 1 \
  --profile lookoutvision-access
```

Jika panggilan berhasil, output berikut akan ditampilkan:

```
{
    "Status": "STOPPING_HOSTING"
}
```

6. Kembali ke konsol, pilih Model di halaman navigasi kiri.
7. Model telah berhenti ketika status model di kolom Status Pelatihan selesai.

Langkah selanjutnya

Bila Anda siap buat model dengan gambar Anda sendiri, mulailah dengan mengikuti petunjuk diMembuat proyek Anda. Instruksi mencakup langkah-langkah untuk membuat model dengan konsol Amazon Lookout for Vision dan dengan AWS SDK.

Jika Anda ingin mencoba set data contoh lainnya, lihat. Contoh kode dan kumpulan data

Membuat model Amazon Lookout for Vision

Model Amazon Lookout for Vision adalah model pembelajaran mesin yang memprediksi adanya anomali pada gambar baru dengan menemukan pola dalam gambar yang digunakan untuk melatih model. Bagian ini menunjukkan cara membuat dan melatih model. Setelah Anda melatih model Anda, Anda mengevaluasi kinerjanya. Untuk informasi selengkapnya, lihat Meningkatkan model Amazon Lookout for Vision.

Sebelum Anda membuat model pertama Anda, kami sarankan Anda membaca Memahami Amazon Lookout for Vision danMemulai dengan Amazon Lookout for Vision. Jika Anda menggunakan AWS SDK, bacaHubungi operasi Amazon Lookout for Vision.

Topik

- Membuat proyek Anda
- Membuat dataset Anda
- Pelabelan gambar
- Melatih model Anda
- Pelatihan model pemecahan masalah

Membuat proyek Anda

Proyek Amazon Lookout for Vision adalah pengelompokan sumber daya yang dibutuhkan untuk membuat dan mengelola model Lookout for Vision. Sebuah proyek mengelola hal-hal berikut:

- Dataset — Gambar dan label gambar yang digunakan untuk melatih model. Untuk informasi selengkapnya, lihat Membuat dataset Anda.
- Model — Perangkat lunak yang Anda latih untuk mendeteksi anomali. Anda dapat memiliki beberapa versi model. Untuk informasi selengkapnya, lihat Melatih model Anda.

Kami menyarankan Anda menggunakan proyek untuk kasus penggunaan tunggal, seperti mendeteksi anomali dalam satu jenis bagian mesin.

> **Note**
>
> Anda dapat menggunakan AWS CloudFormation untuk menyediakan dan mengonfigurasi proyek Amazon Lookout for Vision. Untuk informasi selengkapnya, lihat Membuat sumber daya Amazon Lookout for VisionAWS CloudFormation.

Untuk melihat proyek Anda, lihat Melihat proyek Anda atau bukaMenggunakan dasbor Amazon Lookout for Vision. Untuk menghapus model, lihat Menghapus model.

Topik

- Membuat proyek (konsol)
- Membuat proyek (SDK)

Membuat proyek (konsol)

Prosedur berikut menunjukkan cara membuat proyek menggunakan konsol.

Untuk membuat proyek (konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Di panel navigasi kiri, pilih Proyek.
3. Pilih Buat proyek.
4. Dalam Nama proyek, masukkan nama untuk proyek Anda.
5. Pilih Buat proyek. Halaman detail untuk proyek Anda ditampilkan.
6. Ikuti langkah-langkah Membuat dataset Anda untuk membuat kumpulan data Anda.

Membuat proyek (SDK)

Anda menggunakan CreateProjectoperasi untuk membuat proyek Amazon Lookout for Vision. Tanggapan dari `CreateProject` termasuk nama proyek dan Nama Sumber Daya Amazon (ARN) proyek. Setelah itu, hubungi CreateDatasetuntuk menambahkan pelatihan dan kumpulan data pengujian ke proyek Anda. Untuk informasi selengkapnya, lihat Membuat kumpulan data dengan file manifes (SDK).

Untuk melihat proyek yang telah Anda buat dalam sebuah proyek, hubungi`ListProjects`. Untuk informasi selengkapnya, lihat Melihat proyek Anda.

Untuk membuat proyek (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode contoh berikut untuk membuat model.

   CLI

   Ubah nilai `project-name` ke nama yang ingin Anda gunakan untuk proyek.

   ```
   aws lookoutvision create-project --project-name project name \
     --profile lookoutvision-access
   ```

   Python

   Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

   ```
       @staticmethod
       def create_project(lookoutvision_client, project_name):
           """
           Creates a new Lookout for Vision project.

           :param lookoutvision_client: A Boto3 Lookout for Vision client.
           :param  project_name: The name for the new project.
           :return project_arn: The ARN of the new project.
           """
           try:
               logger.info("Creating project: %s", project_name)
               response =
   lookoutvision_client.create_project(ProjectName=project_name)
               project_arn = response["ProjectMetadata"]["ProjectArn"]
               logger.info("project ARN: %s", project_arn)
           except ClientError:
               logger.exception("Couldn't create project %s.", project_name)
               raise
           else:
               return project_arn
   ```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
/**
 * Creates an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return ProjectMetadata Metadata information about the created project.
 */
public static ProjectMetadata createProject(LookoutVisionClient lfvClient,
 String projectName)
                throws LookoutVisionException {

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

        CreateProjectResponse response =
 lfvClient.createProject(createProjectRequest);

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

        return response.projectMetadata();

}
```

3. Ikuti langkah-langkah Membuat kumpulan data menggunakan file manifes Amazon SageMaker Ground Truth untuk membuat kumpulan data Anda.

Membuat dataset Anda

Dataset berisi gambar dan label yang ditetapkan yang Anda gunakan untuk melatih dan menguji model. Anda membuat kumpulan data untuk project Anda dengan konsol Amazon Lookout for Vision

atau dengan operasi. CreateDataset Gambar dataset harus diberi label sesuai dengan jenis model yang ingin Anda buat (klasifikasi gambar atau segmentasi gambar).

Topik

Mempersiapkan gambar untuk dataset

Anda memerlukan koleksi gambar untuk membuat kumpulan data. Gambar Anda harus berupa file format PNG atau JPEG. Jumlah dan jenis gambar yang Anda butuhkan tergantung pada apakah proyek Anda memiliki satu set data tunggal atau kumpulan data pelatihan dan pengujian terpisah.

Proyek kumpulan data tunggal

Untuk membuat model klasifikasi gambar, Anda memerlukan yang berikut ini untuk memulai pelatihan:

- Setidaknya 20 gambar objek normal.
- Setidaknya 10 gambar objek anomali.

Untuk membuat model segmentasi gambar, Anda memerlukan yang berikut ini untuk memulai pelatihan:

- Setidaknya 20 gambar dari setiap jenis anomali.
- Setiap gambar anomali (gambar dengan tipe anomali yang ada) harus hanya memiliki satu jenis anomali.
- Setidaknya 20 gambar objek normal.

Pisahkan proyek kumpulan data pelatihan dan pengujian

Untuk membuat model klasifikasi gambar, Anda memerlukan yang berikut:

- Setidaknya 10 gambar objek normal dalam dataset pelatihan.
- Setidaknya 10 gambar objek normal dalam dataset pengujian.
- Setidaknya 10 gambar objek anomali dalam dataset uji.

Untuk membuat model segmentasi gambar, Anda memerlukan yang berikut ini:

- Setiap dataset membutuhkan setidaknya 10 gambar dari setiap jenis anomali.
- Setiap gambar anomali (gambar dengan tipe anomali yang ada) harus mengandung hanya satu jenis anomali.
- Setiap dataset harus memiliki setidaknya 10 gambar objek normal.

Untuk membuat model berkualitas lebih tinggi, gunakan lebih dari jumlah minimum gambar. Jika Anda membuat model segmentasi, sebaiknya sertakan gambar dengan beberapa jenis anomali, tetapi ini tidak diperhitungkan dalam jumlah minimum yang dibutuhkan Lookout for Vision untuk memulai pelatihan.

Gambar Anda harus dari satu jenis objek. Selain itu, Anda harus memiliki kondisi pengambilan gambar yang konsisten, seperti pemosisian kamera, pencahayaan, dan pose objek.

Semua gambar dalam kumpulan data pelatihan dan pengujian harus memiliki dimensi yang sama. Kemudian, gambar yang Anda analisis dengan model terlatih Anda harus memiliki dimensi yang sama dengan gambar kumpulan data pelatihan dan pengujian. Untuk informasi selengkapnya, lihat Mendeteksi anomali dalam gambar.

Semua gambar pelatihan dan tes harus berupa gambar yang unik, lebih disukai dari objek unik. Gambar normal harus menangkap variasi normal dari objek yang sedang dianalisis. Gambar anomali harus menangkap beragam sampel anomali.

Amazon Lookout for Vision memberikan contoh gambar yang dapat Anda gunakan. Untuk informasi selengkapnya, lihat Set data klasifikasi gambar.

Untuk batas gambar, lihatQuotas.

Membuat dataset

Saat Anda membuat kumpulan data untuk proyek Anda, Anda memilih konfigurasi kumpulan data awal proyek Anda. Anda juga memilih dari mana Lookout for Vision mengimpor gambar.

Memilih konfigurasi kumpulan data untuk proyek Anda

Saat Anda membuat kumpulan data pertama dalam proyek Anda, Anda memilih salah satu konfigurasi kumpulan data berikut:

- Dataset tunggal — Sebuah proyek dataset tunggal menggunakan satu set data untuk melatih dan menguji model Anda. Menggunakan satu kumpulan data menyederhanakan pelatihan dengan membiarkan Amazon Lookout for Vision memilih gambar pelatihan dan pengujian. Selama pelatihan, Amazon Lookout for Vision, secara internal membagi kumpulan data menjadi kumpulan data pelatihan dan kumpulan data pengujian. Anda tidak memiliki akses ke kumpulan data terpisah. Sebaiknya gunakan satu proyek kumpulan data untuk sebagian besar skenario.
- Pisahkan kumpulan data pelatihan dan pengujian — Jika Anda menginginkan kontrol yang lebih baik atas pelatihan, pengujian, dan penyetelan kinerja, Anda dapat mengonfigurasi proyek Anda agar memiliki kumpulan data pelatihan dan pengujian yang terpisah. Gunakan kumpulan data pengujian terpisah jika Anda ingin mengontrol gambar yang digunakan untuk pengujian, atau jika Anda sudah memiliki kumpulan gambar tolok ukur yang ingin Anda gunakan.

Anda dapat menambahkan kumpulan data pengujian ke proyek kumpulan data tunggal yang ada. Dataset tunggal kemudian menjadi kumpulan data pelatihan. Jika Anda menghapus kumpulan data pengujian dari proyek dengan kumpulan data pelatihan dan pengujian terpisah, proyek menjadi proyek kumpulan data tunggal. Untuk informasi selengkapnya, lihat Menghapus dataset.

Mengimpor gambar

Saat Anda membuat kumpulan data, Anda memilih tempat untuk mengimpor gambar. Bergantung pada bagaimana Anda mengimpor gambar, gambar mungkin sudah diberi label. Jika gambar tidak diberi label setelah membuat kumpulan data, lihat. Pelabelan gambar

Anda membuat kumpulan data dan mengimpor gambarnya dengan salah satu cara berikut:

- Impor gambar dari komputer lokal Anda. Gambar tidak diberi label. Anda menambahkan atau memberi label dengan menggunakan konsol Lookout for Vision.
- Impor gambar dari ember S3. Amazon Lookout for Vision dapat mengklasifikasikan gambar dengan menggunakan nama folder untuk memberi label pada gambar. Gunakan `normal` untuk gambar normal. Gunakan `anomaly` untuk gambar anomali. Anda tidak dapat secara otomatis menetapkan label segmentasi.
- Impor file manifes Amazon SageMaker Ground Truth, yang menyertakan gambar berlabel. Anda dapat membuat dan mengimpor file manifes Anda sendiri. Jika Anda memiliki banyak gambar,

pertimbangkan untuk menggunakan layanan pelabelan SageMaker Ground Truth. Anda kemudian mengimpor file manifes keluaran dari pekerjaan Amazon SageMaker Ground Truth. Jika perlu, Anda dapat menggunakan konsol Lookout for Vision untuk menambah atau mengubah label.

Jika Anda menggunakan AWS SDK, Anda membuat kumpulan data dengan file manifes Amazon SageMaker Ground Truth. Untuk informasi selengkapnya, lihat Membuat kumpulan data menggunakan file manifes Amazon SageMaker Ground Truth.

Jika, setelah membuat kumpulan data Anda, gambar Anda diberi label, Anda dapat melatih modelnya. Jika gambar tidak diberi label, tambahkan label sesuai dengan jenis model yang ingin Anda buat. Untuk informasi selengkapnya, lihat Pelabelan gambar.

Anda dapat menambahkan lebih banyak gambar ke kumpulan data yang ada. Untuk informasi selengkapnya, lihat Menambahkan gambar ke kumpulan data Anda.

Membuat kumpulan data menggunakan gambar yang disimpan di komputer lokal Anda

Anda dapat membuat kumpulan data dengan menggunakan gambar yang dimuat langsung dari komputer Anda. Anda dapat mengunggah hingga 30 gambar sekaligus. Dalam prosedur ini, Anda dapat membuat proyek kumpulan data tunggal, atau proyek dengan kumpulan data pelatihan dan pengujian terpisah.

> **Note**
>
> Jika Anda baru saja selesaiMembuat proyek Anda, konsol harus menampilkan dasbor proyek Anda dan Anda tidak perlu melakukan langkah 1 - 3.

Untuk membuat kumpulan data menggunakan gambar di komputer lokal (konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Di panel navigasi kiri, pilih Proyek.
3. Di halaman Proyek, pilih proyek yang ingin Anda tambahkan dataset.
4. Pada halaman detail proyek, pilih Buat kumpulan data.
5. Pilih tab Set data tunggal atau tab Kumpulan data pelatihan dan uji terpisah dan ikuti langkah-langkahnya.

Single dataset

a. Di bagian konfigurasi Dataset, pilih Buat kumpulan data tunggal.
b. Di bagian Konfigurasi sumber gambar, pilih Unggah gambar dari komputer Anda.
c. Pilih Buat kumpulan data.
d. Pada halaman dataset, pilih Tambahkan gambar.
e. Pilih gambar yang ingin Anda unggah ke kumpulan data dari file komputer Anda. Anda dapat menyeret gambar atau memilih gambar yang ingin Anda unggah dari komputer lokal Anda.
f. Pilih Unggah gambar.

Separate training and test datasets

a. Di bagian konfigurasi Dataset, pilih Buat kumpulan data pelatihan dan kumpulan data pengujian.
b. Di bagian Detail kumpulan data pelatihan, pilih Unggah gambar dari komputer Anda.
c. Di bagian Uji detail kumpulan data, pilih Unggah gambar dari komputer Anda.

> **Note**
>
> Kumpulan data pelatihan dan pengujian Anda dapat memiliki sumber gambar yang berbeda.

d. Pilih Buat kumpulan data. Halaman kumpulan data muncul dengan tab Pelatihan dan tab Uji untuk kumpulan data masing-masing.
e. Pilih Tindakan, lalu pilih Tambahkan gambar ke kumpulan data pelatihan.
f. Pilih gambar yang ingin Anda unggah ke kumpulan data. Anda dapat menyeret gambar atau memilih gambar yang ingin Anda unggah dari komputer lokal Anda.
g. Pilih Unggah gambar.
h. Ulangi langkah 5e - 5g. Untuk langkah 5e, pilih Tindakan dan kemudian pilih Tambahkan gambar untuk menguji kumpulan data.

6. Ikuti langkah-langkah Pelabelan gambar untuk memberi label pada gambar Anda.
7. Ikuti langkah-langkah Melatih model Anda untuk melatih model Anda.

Membuat kumpulan data menggunakan gambar yang disimpan di bucket Amazon S3

Anda dapat membuat kumpulan data menggunakan gambar yang disimpan di bucket Amazon S3. Dengan opsi ini, Anda dapat menggunakan struktur folder di bucket Amazon S3 Anda untuk mengklasifikasikan gambar Anda secara otomatis. Anda dapat menyimpan gambar di bucket konsol atau bucket Amazon S3 lainnya di akun Anda.

Menyiapkan folder untuk pelabelan otomatis

Selama pembuatan kumpulan data, Anda dapat memilih untuk menetapkan nama label ke gambar berdasarkan nama folder yang berisi gambar. Folder harus merupakan turunan dari jalur folder Amazon S3 yang Anda tentukan di URI S3 saat Anda membuat kumpulan data.

Berikut ini adalah `train` folder untuk gambar contoh Memulai. Jika Anda menentukan lokasi folder Amazon S3 sebagai`S3-bucket/circuitboard/train/`, gambar di folder `normal` diberi label. `Normal` Gambar dalam folder `anomaly` diberi label`Anomaly`. Nama-nama folder anak yang lebih dalam tidak digunakan untuk memberi label pada gambar.

```
S3-bucket
  ### circuitboard
    ### train
        ### anomaly
            ### train-anomaly_1.jpg
            ### train-anomaly_2.jpg
            ### .
            ### .
        ### normal
            ### train-normal_1.jpg
            ### train-normal_2.jpg
            ### .
            ### .
```

Membuat kumpulan data menggunakan gambar dari bucket Amazon S3

Prosedur berikut membuat kumpulan data menggunakan contoh klasifikasi gambar yang disimpan dalam bucket Amazon S3. Untuk menggunakan gambar Anda sendiri, buat struktur folder yang dijelaskan diMenyiapkan folder untuk pelabelan otomatis.

Prosedur ini juga menunjukkan cara membuat proyek kumpulan data tunggal, atau proyek yang menggunakan kumpulan data pelatihan dan pengujian terpisah.

Jika Anda tidak memilih untuk memberi label secara otomatis pada gambar Anda, Anda perlu memberi label pada gambar setelah kumpulan data dibuat. Untuk informasi selengkapnya, lihat Mengklasifikasikan gambar (konsol).

> **Note**
>
> Jika Anda baru saja selesaiMembuat proyek Anda, konsol harus menampilkan dasbor proyek Anda dan Anda tidak perlu melakukan langkah 1 - 4.

Untuk membuat kumpulan data menggunakan gambar yang disimpan di bucket Amazon S3

1. Jika Anda belum melakukannya, unggah gambar yang memulai ke bucket Amazon S3 Anda. Untuk informasi selengkapnya, lihat Set data klasifikasi gambar.
2. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
3. Di panel navigasi kiri, pilih Proyek.
4. Di halaman Proyek, pilih proyek yang ingin Anda tambahkan dataset. Halaman detail untuk proyek Anda ditampilkan.
5. Pilih Buat kumpulan data. Halaman Create dataset ditampilkan.

   > **Tip**
   >
   > Jika Anda mengikuti petunjuk Memulai, pilih Buat kumpulan data pelatihan dan kumpulan data pengujian.

6. Pilih tab Set data tunggal atau tab Kumpulan data pelatihan dan uji terpisah dan ikuti langkah-langkahnya.

   Single dataset

   a. Di bagian konfigurasi Dataset, pilih Buat kumpulan data tunggal.

   b. Masukkan informasi untuk langkah 7 - 9 di bagian Konfigurasi sumber gambar.

Separate training and test datasets

a. Di bagian konfigurasi Dataset, pilih Buat kumpulan data pelatihan dan kumpulan data pengujian.
b. Untuk kumpulan data pelatihan Anda, masukkan informasi untuk langkah 7 - 9 di bagian Detail kumpulan data Pelatihan.
c. Untuk kumpulan data pengujian Anda, masukkan informasi untuk langkah 7 - 9 di bagian Detail kumpulan data Uji.

> **Note**
>
> Kumpulan data pelatihan dan pengujian Anda dapat memiliki sumber gambar yang berbeda.

7. Pilih Impor gambar dari ember Amazon S3.
8. Di URI S3, masukkan lokasi bucket Amazon S3 dan jalur folder. Ubah `bucket` ke nama bucket Amazon S3 Anda.

   a. Jika Anda membuat proyek kumpulan data tunggal atau kumpulan data pelatihan, masukkan yang berikut ini:

   ```
   s3://bucket/circuitboard/train/
   ```

   b. Jika Anda membuat kumpulan data pengujian, masukkan yang berikut ini:

   ```
   s3://bucket/circuitboard/test/
   ```

9. Pilih Secara otomatis melampirkan label ke gambar berdasarkan folder.
10. Pilih Buat kumpulan data. Halaman dataset terbuka dengan gambar berlabel Anda.
11. Ikuti langkah-langkah Melatih model Anda untuk melatih model Anda.

Membuat kumpulan data menggunakan file manifes Amazon SageMaker Ground Truth

File manifes berisi informasi tentang gambar dan label gambar yang dapat Anda gunakan untuk melatih dan menguji model. Anda dapat menyimpan file manifes di bucket Amazon S3 dan

menggunakannya untuk membuat kumpulan data. Anda dapat membuat file manifes Anda sendiri atau Anda dapat menggunakan file manifes yang ada, seperti output dari pekerjaan Amazon SageMaker Ground Truth.

Topik

Menggunakan pekerjaan Amazon Sagemaker Ground Truth

Pelabelan gambar dapat memakan waktu yang signifikan. Misalnya, dibutuhkan waktu 10 detik untuk menggambar topeng secara akurat di sekitar anomali. Jika Anda memiliki 100-an gambar, mungkin perlu beberapa jam untuk melabelinya. Sebagai alternatif untuk memberi label pada gambar sendiri, pertimbangkan untuk menggunakan Amazon SageMaker Ground Truth.

Dengan Amazon SageMaker Ground Truth, Anda dapat menggunakan pekerja dari Amazon Mechanical Turk, perusahaan vendor yang Anda pilih, atau tenaga kerja pribadi internal untuk membuat kumpulan gambar berlabel. Untuk informasi selengkapnya, lihat Menggunakan Amazon SageMaker Ground Truth untuk Label Data.

Ada biaya untuk menggunakan Amazon Mechanical Turk. Juga, Mungkin perlu beberapa hari untuk menyelesaikan pekerjaan pelabelan Amazon Ground Truth. Jika biaya menjadi masalah, atau jika Anda perlu melatih model Anda dengan cepat, kami sarankan Anda menggunakan konsol Amazon Lookout for Vision untuk memberi label pada gambar Anda.

Anda dapat menggunakan pekerjaan pelabelan Amazon SageMaker Ground Truth untuk memberi label gambar yang cocok untuk model klasifikasi gambar dan model segmentasi gambar. Setelah pekerjaan selesai, Anda menggunakan file manifes keluaran untuk membuat kumpulan data Amazon Lookout for Vision.

Klasifikasi gambar

Untuk memberi label gambar untuk model klasifikasi gambar, buat pekerjaan pelabelan untuk tugas Klasifikasi Gambar (Label Tunggal).

Segmentasi gambar

Untuk memberi label gambar untuk model segmentasi gambar, buat pekerjaan pelabelan untuk tugas Klasifikasi Gambar (Label Tunggal). Kemudian, rantai pekerjaan untuk membuat pekerjaan pelabelan untuk tugas Segmentasi Semantik Gambar.

Anda juga dapat menggunakan pekerjaan pelabelan untuk membuat file manifes sebagian untuk model segmentasi gambar. Misalnya, Anda dapat mengklasifikasikan gambar dengan tugas Klasifikasi Gambar (Label Tunggal). Setelah membuat kumpulan data Lookout for Vision dengan output pekerjaan, gunakan konsol Amazon Lookout for Vision untuk menambahkan masker segmentasi dan label anomali ke gambar kumpulan data.

Melabeli gambar dengan Amazon SageMaker Ground Truth

Prosedur berikut menunjukkan cara memberi label gambar dengan tugas pelabelan gambar Amazon SageMaker Ground Truth. Prosedur ini membuat file manifes klasifikasi gambar dan secara opsional menghubungkan tugas pelabelan gambar untuk membuat file manifes segmentasi gambar. Jika Anda ingin proyek Anda memiliki kumpulan data pengujian terpisah, ulangi prosedur ini untuk membuat file manifes untuk kumpulan data pengujian.

Untuk memberi label pada gambar dengan Amazon SageMaker Ground Truth (Konsol)

1. Buat pekerjaan Ground Truth untuk tugas Klasifikasi Gambar (Single Label) dengan mengikuti petunjuk di Create a Labeling Job (Console).

   a. Untuk langkah 10, pilih Gambar dari menu tarik-turun kategori Tugas, dan pilih Klasifikasi Gambar (Label Tunggal) sebagai jenis tugas.
   b. Untuk langkah 16, di bagian alat pelabelan klasifikasi gambar (Single Label), tambahkan dua label: normal dan anomali.
2. Tunggu hingga tenaga kerja selesai mengklasifikasikan gambar Anda.
3. Jika Anda membuat kumpulan data untuk model segmentasi gambar, lakukan hal berikut. Jika tidak, lanjutkan ke langkah 4.

   a. Di konsol Amazon SageMaker Ground Truth, buka halaman Pekerjaan Pelabelan.
   b. Pilih pekerjaan yang Anda buat sebelumnya. Ini memungkinkan menu Tindakan.
   c. Dari menu Tindakan, pilih Rantai. Halaman detail pekerjaan terbuka.
   d. Dalam tipe tugas, pilih segmentasi semantik.
   e. Pilih Selanjutnya.
   f. Di bagian alat pelabelan segmentasi semantik, tambahkan label anomali untuk setiap jenis anomali yang Anda ingin model Anda temukan.
   g. Pilih Buat.
   h. Tunggu sampai tenaga kerja memberi label pada gambar Anda.
4. Buka konsol Ground Truth dan buka halaman Pekerjaan Pelabelan.

5. Jika Anda membuat model klasifikasi gambar, pilih pekerjaan yang Anda buat di langkah 1. Jika Anda membuat model segmentasi gambar, pilih pekerjaan yang dibuat pada langkah 3.
6. Dalam Pelabelan ringkasan pekerjaan buka lokasi S3 di lokasi dataset Output. Perhatikan lokasi file manifes, yang seharusnya`s3://`*`output-dataset-location`*`/manifests/output/output.manifest`.
7. Ulangi prosedur ini jika Anda ingin membuat file manifes untuk kumpulan data pengujian. Jika tidak, ikuti petunjuk di Membuat dataset untuk membuat kumpulan data dengan file manifes.

Membuat dataset

Gunakan prosedur ini untuk membuat kumpulan data dalam proyek Lookout for Vision dengan file manifes yang Anda catat di langkah 6. Melabeli gambar dengan Amazon SageMaker Ground Truth File manifes membuat kumpulan data pelatihan untuk satu proyek kumpulan data. Jika ingin project memiliki kumpulan data pengujian terpisah, Anda dapat menjalankan tugas Amazon SageMaker Ground Truth lainnya untuk membuat file manifes untuk kumpulan data pengujian. Atau Anda dapat membuat file manifes sendiri. Anda juga dapat mengimpor gambar ke kumpulan data pengujian dari bucket Amazon S3 atau dari komputer lokal Anda. (Gambar mungkin perlu diberi label sebelum Anda dapat melatih modelnya).

Prosedur ini mengasumsikan bahwa proyek Anda tidak memiliki kumpulan data apa pun.

Untuk membuat kumpulan data dengan Lookout for Vision (Konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi kiri, pilih Proyek.
4. Pilih proyek yang ingin Anda tambahkan untuk digunakan dengan file manifes.
5. Di bagian Cara kerjanya, pilih Buat kumpulan data.
6. Pilih tab Set data tunggal atau tab Kumpulan data pelatihan dan uji terpisah dan ikuti langkah-langkahnya.

   Single dataset

   1. Pilih Buat satu set data.
   2. Di bagian Konfigurasi sumber gambar, pilih Impor gambar yang diberi label oleh SageMaker Ground Truth.

3. Untuk lokasi file.manifest, masukkan lokasi file manifes yang Anda catat di langkah 6. Melabeli gambar dengan Amazon SageMaker Ground Truth

Separate training and test datasets

1. Pilih Buat kumpulan data pelatihan dan kumpulan data pengujian.
2. Di bagian Detail kumpulan data Pelatihan, pilih Impor gambar berlabel SageMaker Ground Truth.
3. Di lokasi file.manifest, lokasi file manifes yang Anda catat di langkah 6 dari. Melabeli gambar dengan Amazon SageMaker Ground Truth
4. Di bagian Test dataset details, pilih Impor gambar berlabel SageMaker Ground Truth.
5. Di lokasi file.manifest, lokasi file manifes yang Anda catat di langkah 6 dari. Melabeli gambar dengan Amazon SageMaker Ground Truth Ingatlah bahwa Anda memerlukan file manifes terpisah untuk kumpulan data pengujian.

7. Pilih Kirim.
8. Ikuti langkah-langkah Melatih model Anda untuk melatih model Anda.

Membuat file manifes

Anda dapat membuat kumpulan data dengan mengimpor file manifes format SageMaker Ground Truth. Jika gambar Anda diberi label dalam format yang bukan file manifes SageMaker Ground Truth, gunakan informasi berikut untuk membuat file manifes format SageMaker Ground Truth.

File manifes dalam format baris JSON di mana setiap baris adalah objek JSON lengkap yang mewakili informasi pelabelan untuk gambar. Ada berbagai format untuk klasifikasi gambar dan segmentasi gambar. File manifes harus dikodekan menggunakan pengkodean UTF-8.

> **Note**
>
> Contoh baris JSON di bagian ini diformat agar mudah dibaca.

Gambar yang direferensikan oleh file manifes harus ditempatkan di bucket Amazon S3 yang sama. File manifes dapat berada di ember yang berbeda. Anda menentukan lokasi gambar di `source-ref` bidang garis JSON.

Anda dapat membuat file manifes dengan menggunakan kode. Notebook Python Lookout for Vision Lab Amazon menunjukkan cara membuat file manifes klasifikasi gambar untuk contoh gambar sirkuit. Atau, Anda dapat menggunakan kode contoh Datasets di Repositori Contoh AWS Kode. Anda dapat dengan mudah membuat file manifes dengan menggunakan file Comma Separated Values (CSV). Untuk informasi selengkapnya, lihat Membuat file manifes klasifikasi dari file CSV.

Topik

Mendefinisikan garis JSON untuk klasifikasi gambar

Anda menentukan baris JSON untuk setiap gambar yang ingin Anda gunakan dalam file manifes Amazon Lookout for Vision. Jika Anda ingin membuat model klasifikasi, garis JSON harus menyertakan klasifikasi gambar yang normal atau anomali. Sebuah baris JSON dalam format SageMaker Ground Truth Classification Job Output. File manifes terbuat dari satu atau lebih baris JSON, satu untuk setiap gambar yang ingin Anda impor.

Untuk membuat file manifes untuk gambar baris

1. Buat file teks kosong.
2. Tambahkan baris JSON untuk setiap gambar yang ingin Anda impor. Setiap baris JSON akan terlihat mirip dengan yang berikut:

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

3. Simpan file tersebut.

> **Note**
>
> Anda dapat menggunakan ekstensi `.manifest`, tetapi tidak diperlukan.

4. Buat kumpulan data menggunakan file manifes yang Anda buat. Untuk informasi selengkapnya, lihat Membuat file manifes.

Klasifikasi garis JSON

Di bagian ini, Anda mempelajari cara membuat garis JSON yang mengklasifikasikan gambar sebagai normal atau anomali.

Garis JSON anomali

Baris JSON berikut menunjukkan gambar yang diberi label sebagai anomali. Perhatikan bahwa nilai `class-name` adalah`anomaly`.

```
{
    "source-ref": "s3: //bucket/image/anomaly/abnormal-1.jpg",
    "anomaly-label-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/auto-label",
        "class-name": "anomaly",
        "human-annotated": "yes",
        "creation-date": "2020-11-10T03:37:09.600",
        "type": "groundtruth/image-classification"
    },
    "anomaly-label": 1
}
```

Garis JSON normal

Baris JSON berikut menunjukkan gambar berlabel normal. Perhatikan bahwa nilai `class-name` adalah`normal`.

```
{
```

```
    "source-ref": "s3: //bucket/image/normal/2020-10-20_12-14-55_613.jpeg",
    "anomaly-label-metadata": {
        "confidence": 1,
        "job-name": "labeling-job/auto-label",
        "class-name": "normal",
        "human-annotated": "yes",
        "creation-date": "2020-11-10T03:37:09.603",
        "type": "groundtruth/image-classification"
    },
    "anomaly-label": 0
}
```

Kunci dan nilai baris JSON

Informasi berikut menjelaskan kunci dan nilai dalam baris Amazon Lookout for Vision JSON.

sumber-ref

(Wajib) Lokasi Amazon S3 dari gambar. Formatnya adalah "`s3://BUCKET/OBJECT_PATH`". Gambar dalam kumpulan data yang diimpor harus disimpan dalam bucket Amazon S3 yang sama.

label anomali

(Wajib) Atribut label. Gunakan kunci`anomaly-label`, atau nama kunci lain yang Anda pilih. Nilai kunci (`0`dalam contoh sebelumnya) diperlukan oleh Amazon Lookout for Vision, tetapi tidak digunakan. Manifes keluaran yang dibuat oleh Amazon Lookout for Vision mengubah nilai 1 menjadi gambar anomali dan nilai untuk gambar normal. `0` Nilai `class-name` menentukan apakah gambar itu normal atau anomali.

Harus ada metadata terkait yang diidentifikasi dengan nama bidang dengan -metadata ditambahkan. Misalnya, "`anomaly-label-metadata`".

anomali-label-metadata

(Wajib) Metadata tentang atribut label. Nama bidang harus sama dengan atribut label dengan -metadata ditambahkan.

kepercayaan diri

(Opsional) Saat ini tidak digunakan oleh Amazon Lookout for Vision. Jika Anda menentukan nilai, gunakan nilai1.

nama-pekerjaan

(Opsional) Nama yang Anda pilih untuk pekerjaan yang memproses gambar.

nama kelas

(Wajib) Jika gambar berisi konten normal, tentukan`normal`, jika tidak tentukan`anomaly`. Jika nilai `class-name` adalah nilai lain, gambar ditambahkan ke kumpulan data sebagai gambar yang tidak berlabel. Untuk memberi label pada gambar, lihatMenambahkan gambar ke kumpulan data Anda.

beranotasi manusia

(Wajib) Tentukan`"yes"`, jika anotasi diselesaikan oleh manusia. Jika tidak, tentukan`"no"`.

kreasi-tanggal

(Opsional) Tanggal dan waktu Universal Terkoordinasi (UTC) saat label dibuat.

jenis

(Wajib) Jenis pemrosesan yang harus diterapkan pada gambar. Untuk label anomali tingkat gambar, nilainya adalah. `"groundtruth/image-classification"`

Mendefinisikan garis JSON untuk segmentasi gambar

Anda menentukan baris JSON untuk setiap gambar yang ingin Anda gunakan dalam file manifes Amazon Lookout for Vision. Jika Anda ingin membuat model segmentasi, Garis JSON harus menyertakan informasi segmentasi dan klasifikasi untuk gambar. File manifes terbuat dari satu atau lebih baris JSON, satu untuk setiap gambar yang ingin Anda impor.

Untuk membuat file manifes untuk gambar tersegmentasi

1. Buat file teks kosong.
2. Tambahkan baris JSON untuk setiap gambar yang ingin Anda impor. Setiap baris JSON akan terlihat mirip dengan yang berikut:

```
{"source-ref":"s3://path-to-image","anomaly-label":1,"anomaly-label-metadata":
{"class-name":"anomaly","creation-date":"2021-10-12T14:16:45.668","human-
annotated":"yes","job-name":"labeling-job/classification-job","type":"groundtruth/
image-classification","confidence":1},"anomaly-mask-ref":"s3://path-to-
image","anomaly-mask-ref-metadata":{"internal-color-map":{"0":{"class-
name":"BACKGROUND","hex-color":"#ffffff","confidence":0.0},"1":{"class-
name":"scratch","hex-color":"#2ca02c","confidence":0.0},"2":{"class-
```

```
name":"dent","hex-color":"#1f77b4","confidence":0.0}},"type":"groundtruth/
semantic-segmentation","human-annotated":"yes","creation-
date":"2021-11-23T20:31:57.758889","job-name":"labeling-job/segmentation-job"}}
```

3. Simpan file tersebut.

> **Note**
>
> Anda dapat menggunakan ekstensi `.manifest`, tetapi tidak diperlukan.

4. Buat kumpulan data menggunakan file manifes yang Anda buat. Untuk informasi selengkapnya, lihat Membuat file manifes.

Segmentasi garis JSON

Di bagian ini, Anda mempelajari cara membuat garis JSON yang mencakup informasi segmentasi dan klasifikasi untuk gambar.

Baris JSON berikut menunjukkan gambar dengan informasi segmentasi dan klasifikasi. `anomaly-label-metadata`berisi informasi klasifikasi. `anomaly-mask-ref`dan `anomaly-mask-ref-metadata` berisi informasi segmentasi.

```
{
    "source-ref": "s3://path-to-image",
    "anomaly-label": 1,
    "anomaly-label-metadata": {
        "class-name": "anomaly",
        "creation-date": "2021-10-12T14:16:45.668",
        "human-annotated": "yes",
        "job-name": "labeling-job/classification-job",
        "type": "groundtruth/image-classification",
        "confidence": 1
    },
    "anomaly-mask-ref": "s3://path-to-image",
    "anomaly-mask-ref-metadata": {
        "internal-color-map": {
            "0": {
                "class-name": "BACKGROUND",
                "hex-color": "#ffffff",
                "confidence": 0.0
            },
```

```
            "1": {
                "class-name": "scratch",
                "hex-color": "#2ca02c",
                "confidence": 0.0
            },
            "2": {
                "class-name": "dent",
                "hex-color": "#1f77b4",
                "confidence": 0.0
            }
        },
        "type": "groundtruth/semantic-segmentation",
        "human-annotated": "yes",
        "creation-date": "2021-11-23T20:31:57.758889",
        "job-name": "labeling-job/segmentation-job"
    }
}
```

Kunci dan nilai baris JSON

Informasi berikut menjelaskan kunci dan nilai dalam baris Amazon Lookout for Vision JSON.

sumber-ref

(Wajib) Lokasi Amazon S3 dari gambar. Formatnya adalah `"s3://BUCKET/OBJECT_PATH"`. Gambar dalam kumpulan data yang diimpor harus disimpan dalam bucket Amazon S3 yang sama.

label anomali

(Wajib) Atribut label. Gunakan kunci`anomaly-label`, atau nama kunci lain yang Anda pilih. Nilai kunci (`1`dalam contoh sebelumnya) diperlukan oleh Amazon Lookout for Vision, tetapi tidak digunakan. Manifes keluaran yang dibuat oleh Amazon Lookout for Vision mengubah nilai 1 menjadi gambar anomali dan nilai untuk gambar normal. `0` Nilai `class-name` menentukan apakah gambar itu normal atau anomali.

Harus ada metadata terkait yang diidentifikasi dengan nama bidang dengan -metadata ditambahkan. Misalnya, `"anomaly-label-metadata"`.

anomali-label-metadata

(Wajib) Metadata tentang atribut label. Berisi informasi klasifikasi. Nama bidang harus sama dengan atribut label dengan -metadata ditambahkan.

kepercayaan diri

(Opsional) Saat ini tidak digunakan oleh Amazon Lookout for Vision. Jika Anda menentukan nilai, gunakan nilai1.

nama-pekerjaan

(Opsional) Nama yang Anda pilih untuk pekerjaan yang memproses gambar.

nama kelas

(Wajib) Jika gambar berisi konten normal, tentukan`normal`, jika tidak tentukan`anomaly`. Jika nilai `class-name` adalah nilai lain, gambar ditambahkan ke kumpulan data sebagai gambar yang tidak berlabel. Untuk memberi label pada gambar, lihatMenambahkan gambar ke kumpulan data Anda.

beranotasi manusia

(Wajib) Tentukan`"yes"`, jika anotasi diselesaikan oleh manusia. Jika tidak, tentukan`"no"`.

kreasi-tanggal

(Opsional) Tanggal dan waktu Universal Terkoordinasi (UTC) saat label dibuat.

jenis

(Wajib) Jenis pemrosesan yang harus diterapkan pada gambar. Gunakan nilainya`"groundtruth/image-classification"`.

anomali-topeng-ref

(Wajib) Lokasi Amazon S3 dari gambar topeng. Gunakan `anomaly-mask-ref` untuk nama kunci atau gunakan nama kunci pilihan Anda. Kuncinya harus diakhiri dengan`-ref`. Gambar topeng harus berisi topeng berwarna untuk setiap jenis anomali. `internal-color-map` Formatnya adalah `"s3://`*`BUCKET`*`/`*`OBJECT_PATH`*`"`. Gambar dalam kumpulan data yang diimpor harus disimpan dalam bucket Amazon S3 yang sama. Gambar topeng harus berupa gambar format Portable Network Graphic (PNG).

anomali-topeng-ref-metadata

(Wajib) Metadata segmentasi untuk gambar. Gunakan `anomaly-mask-ref-metadata` untuk nama kunci atau gunakan nama kunci pilihan Anda. Nama kunci harus diakhiri dengan`-ref-metadata`.

peta warna internal

(Wajib) Peta warna yang dipetakan ke jenis anomali individu. Warna harus sesuai dengan warna pada gambar topeng (`anomaly-mask-ref`).

kunci

(Wajib) Kunci ke peta. Entri `0` harus berisi nama kelas BACKGROUND yang mewakili area di luar anomali pada gambar.

nama kelas

(Wajib) Nama jenis anomali, seperti goresan atau penyok.

hex-warna

(Wajib) Warna hex untuk jenis anomali, seperti. `#2ca02c` Warnanya harus sesuai dengan warna`anomaly-mask-ref`. Nilai untuk tipe `BACKGROUND` anomali selalu. `#ffffff`

kepercayaan

(Wajib) Saat ini tidak digunakan oleh Amazon Lookout for Vision, tetapi nilai float diperlukan.

beranotasi manusia

(Wajib) Tentukan`"yes"`, jika anotasi diselesaikan oleh manusia. Jika tidak, tentukan`"no"`.

kreasi-tanggal

(Opsional) Tanggal dan waktu Universal Terkoordinasi (UTC) saat informasi segmentasi dibuat.

jenis

(Wajib) Jenis pemrosesan yang harus diterapkan pada gambar. Gunakan nilainya`"groundtruth/semantic-segmentation"`.

Membuat file manifes klasifikasi dari file CSV

Contoh skrip Python ini menyederhanakan pembuatan file manifes klasifikasi dengan menggunakan file Comma Separated Values (CSV) untuk memberi label gambar. Anda membuat file CSV.

File manifes menjelaskan gambar yang digunakan untuk melatih model. File manifes terdiri dari satu atau lebih baris JSON. Setiap baris JSON menggambarkan satu gambar. Untuk informasi selengkapnya, lihat Mendefinisikan garis JSON untuk klasifikasi gambar.

File CSV mewakili data tabular di beberapa baris dalam file teks. Bidang pada baris dipisahkan dengan koma. Untuk informasi selengkapnya, lihat nilai yang dipisahkan koma. Untuk skrip ini, setiap baris dalam file CSV Anda menyertakan lokasi S3 gambar dan klasifikasi anomali untuk gambar (atau). `normal` `anomaly` Setiap baris memetakan ke JSON Line dalam file manifes.

Misalnya, File CSV berikut menjelaskan beberapa gambar dalam contoh gambar.

```
s3://s3bucket/circuitboard/train/anomaly/train-anomaly_1.jpg,anomaly
s3://s3bucket/circuitboard/train/anomaly/train-anomaly_10.jpg,anomaly
s3://s3bucket/circuitboard/train/anomaly/train-anomaly_11.jpg,anomaly
s3://s3bucket/circuitboard/train/normal/train-normal_1.jpg,normal
s3://s3bucket/circuitboard/train/normal/train-normal_10.jpg,normal
s3://s3bucket/circuitboard/train/normal/train-normal_11.jpg,normal
```

Script menghasilkan JSON Lines untuk setiap baris. Sebagai contoh, berikut ini adalah JSON Line untuk baris pertama (`s3://s3bucket/circuitboard/train/anomaly/train-anomaly_1.jpg,anomaly`).

```
{"source-ref": "s3://s3bucket/csv_test/train_anomaly_1.jpg","anomaly-label":
 1,"anomaly-label-metadata": {"confidence": 1,"job-name": "labeling-job/anomaly-
classification","class-name": "anomaly","human-annotated": "yes","creation-date":
 "2022-02-04T22:47:07","type": "groundtruth/image-classification"}}
```

Jika file CSV Anda tidak menyertakan jalur Amazon S3 untuk gambar, gunakan `--s3-path` argumen baris perintah untuk menentukan jalur Amazon S3 ke gambar.

Sebelum membuat file manifes, skrip memeriksa gambar duplikat dalam file CSV dan klasifikasi gambar apa pun yang tidak atau. `normal` `anomaly` Jika duplikat kesalahan klasifikasi gambar atau gambar ditemukan, skrip melakukan hal berikut:

- Merekam entri gambar pertama yang valid untuk semua gambar dalam file CSV yang tidak digandakan.
- Merekam kejadian duplikat gambar dalam file kesalahan.
- Merekam klasifikasi gambar yang tidak `normal` atau `anomaly` dalam file kesalahan.
- Tidak membuat file manifes.

File kesalahan mencakup nomor baris di mana gambar duplikat atau kesalahan klasifikasi ditemukan dalam file CSV input. Gunakan file CSV kesalahan untuk memperbarui file CSV input dan kemudian jalankan skrip lagi. Atau, gunakan file CSV kesalahan untuk memperbarui file CSV yang tidak

digandakan, yang hanya berisi entri gambar unik dan gambar tanpa kesalahan klasifikasi gambar. Jalankan kembali skrip dengan file CSV deduplikat yang diperbarui.

Jika tidak ada duplikat atau kesalahan yang ditemukan dalam file CSV input, skrip menghapus file CSV gambar yang tidak digandakan dan file kesalahan, karena kosong.

Dalam prosedur ini, Anda membuat file CSV dan menjalankan skrip Python untuk membuat file manifes. Script telah diuji dengan Python versi 3.7.

Untuk membuat file manifes dari file CSV

1. Buat file CSV dengan bidang berikut di setiap baris (satu baris per gambar). Jangan menambahkan baris header ke file CSV.

Bidang 1	Bidang 2
Nama gambar atau jalur Amazon S3 pada gambar. Misalnya, `s3://s3bucket/circuitboard/train/anomaly/train-anomaly_10.jpg` . Anda tidak dapat memiliki campuran gambar dengan jalur Amazon S3 dan gambar tanpa.	Klasifikasi anomali untuk gambar (`normal`atau`anomaly`).

   Misalnya `s3://s3bucket/circuitboard/train/anomaly/image_10.jpg,anomaly` atau `image_11.jpg,normal`
2. Simpan file CSV.
3. Jalankan skrip Python berikut. Berikan argumen berikut:

   - `csv_file`— File CSV yang Anda buat di langkah 1.
   - (Opsional) `--s3-path` *`s3://path_to_folder/`* - Jalur Amazon S3 untuk ditambahkan ke nama file gambar (bidang 1). Gunakan `--s3-path` jika gambar di bidang 1 belum berisi jalur S3.

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

```
Shows how to create an Amazon Lookout for Vision manifest file from a CSV file.
The CSV file format is image location,anomaly classification (normal or anomaly)
For example:
s3://s3bucket/circuitboard/train/anomaly/train_11.jpg,anomaly
s3://s3bucket/circuitboard/train/normal/train_1.jpg,normal

If necessary, use the bucket argument to specify the Amazon S3 bucket folder for
 the images.
"""

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

logger = logging.getLogger(__name__)


def check_errors(csv_file):
    """
    Checks for duplicate images and incorrect classifications in a CSV file.
    If duplicate images or invalid anomaly assignments are found, an errors CSV
 file
    and deduplicated CSV file are created. Only the first
    occurrence of a duplicate is recorded. Other duplicates are recorded in the
 errors file.
    :param csv_file: The source CSV file
    :return: True if errors or duplicates are found, otherwise false.
    """

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

    errors_found = False
    errors_file = f"{os.path.splitext(csv_file)[0]}_errors.csv"
    deduplicated_file = f"{os.path.splitext(csv_file)[0]}_deduplicated.csv"

    with open(csv_file, 'r', encoding="UTF-8") as input_file,\
            open(deduplicated_file, 'w', encoding="UTF-8") as dedup,\
            open(errors_file, 'w', encoding="UTF-8") as errors:

        reader = csv.reader(input_file, delimiter=',')
        dedup_writer = csv.writer(dedup)
```

```
        error_writer = csv.writer(errors)
        line = 1
        entries = set()
        for row in reader:

            # Skip empty lines.
            if not ''.join(row).strip():
                continue

            # Record any incorrect classifications.
            if not row[1].lower() == "normal" and not row[1].lower() == "anomaly":
                error_writer.writerow(
                    [line, row[0], row[1], "INVALID_CLASSIFICATION"])
                errors_found = True

            # Write first image entry to dedup file and record duplicates.
            key = row[0]
            if key not in entries:
                dedup_writer.writerow(row)
                entries.add(key)
            else:
                error_writer.writerow([line, row[0], row[1], "DUPLICATE"])
                errors_found = True
            line += 1

    if errors_found:
        logger.info("Errors found check %s.", errors_file)
    else:
        os.remove(errors_file)
        os.remove(deduplicated_file)

    return errors_found


def create_manifest_file(csv_file, manifest_file, s3_path):
    """
    Read a CSV file and create an Amazon Lookout for Vision 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 Amazon S3 path to the folder that contains the images.
    """
    logger.info("Processing CSV file %s.", csv_file)
```

```
    image_count = 0
    anomalous_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 the CSV file.
        for row in image_classifications:
            # Skip empty lines.
            if not ''.join(row).strip():
                continue

            source_ref = str(s3_path) + row[0]
            classification = 0

            if row[1].lower() == 'anomaly':
                classification = 1
                anomalous_count += 1

           # Create the JSON line.
            json_line = {}
            json_line['source-ref'] = source_ref
            json_line['anomaly-label'] = str(classification)

            metadata = {}
            metadata['confidence'] = 1
            metadata['job-name'] = "labeling-job/anomaly-classification"
            metadata['class-name'] = row[1]
            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['anomaly-label-metadata'] = metadata

            output_file.write(json.dumps(json_line))
            output_file.write('\n')
            image_count += 1

    logger.info("Finished creating manifest file %s.\n"
                "Images: %s\nAnomalous: %s",
```

```
                manifest_file,
                image_count,
                anomalous_count)
    return image_count, anomalous_count


def add_arguments(parser):
    """
    Add 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 Amazon S3 bucket and folder path for the images."
        " If not supplied, column 1 is assumed to include the Amazon 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 = ""

        csv_file = args.csv_file
        csv_file_no_extension = os.path.splitext(csv_file)[0]
        manifest_file = csv_file_no_extension + '.manifest'

        # Create manifest file if there are no duplicate images.
        if check_errors(csv_file):
```

```
            print(f"Issues found. Use {csv_file_no_extension}_errors.csv "\
                "to view duplicates and errors.")
            print(f"{csv_file}_deduplicated.csv contains the first"\
                "occurrence of a duplicate.\n"
                  "Update as necessary with the correct information.")
            print(f"Re-run the script with
 {csv_file_no_extension}_deduplicated.csv")
        else:
            print('No duplicates found. Creating manifest file.')

            image_count, anomalous_count = create_manifest_file(csv_file,
 manifest_file, s3_path)

            print(f"Finished creating manifest file: {manifest_file} \n")

            normal_count = image_count-anomalous_count
            print(f"Images processed: {image_count}")
            print(f"Normal: {normal_count}")
            print(f"Anomalous: {anomalous_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. Jika gambar duplikat terjadi atau kesalahan klasifikasi terjadi:

   a. Gunakan file kesalahan untuk memperbarui file CSV yang didedupulisasi atau file CSV input.

   b. Jalankan skrip lagi dengan file CSV deduplikat yang diperbarui atau file CSV input yang diperbarui.

5. Jika Anda berencana menggunakan kumpulan data pengujian, ulangi langkah 1—4 untuk membuat file manifes untuk kumpulan data pengujian Anda.

6. Jika perlu, salin gambar dari komputer Anda ke jalur bucket Amazon S3 yang Anda tentukan di kolom 1 file CSV (atau ditentukan dalam `--s3-path` baris perintah). Untuk menyalin gambar, masukkan perintah berikut pada prompt perintah.

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

7. Ikuti petunjuk di Membuat kumpulan data dengan file manifes (konsol) untuk membuat kumpulan data. Jika Anda menggunakan AWS SDK, lihatMembuat kumpulan data dengan file manifes (SDK).

Membuat kumpulan data dengan file manifes (konsol)

Prosedur berikut menunjukkan cara membuat kumpulan data pelatihan atau pengujian dengan mengimpor file manifes SageMaker format yang disimpan dalam bucket Amazon S3.

Setelah membuat kumpulan data, Anda dapat menambahkan lebih banyak gambar ke kumpulan data, atau menambahkan label ke gambar. Untuk informasi selengkapnya, lihat Menambahkan gambar ke kumpulan data Anda.

Untuk membuat kumpulan data menggunakan file manifes format SageMaker Ground Truth (konsol)

1. Buat, atau gunakan file manifes format Ground SageMaker Truth yang kompatibel dengan Amazon Lookout for Vision yang sudah ada. Untuk informasi selengkapnya, lihat Membuat file manifes.
2. Masuk ke AWS Management Console dan buka konsol Amazon S3 di https://console.aws.amazon.com/s3/.
3. Di bucket Amazon S3, buat folder untuk menyimpan file manifes Anda.
4. Unggah file manifes Anda ke folder yang baru saja Anda buat.
5. Di bucket Amazon S3, buat folder untuk menyimpan gambar Anda.
6. Unggah gambar Anda ke folder yang baru saja Anda buat.

   > **Important**
   >
   > Nilai `source-ref` bidang di setiap baris JSON harus dipetakan ke gambar di folder.

7. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
8. Pilih Mulai.
9. Di panel navigasi kiri, pilih Proyek.
10. Pilih proyek yang ingin Anda tambahkan untuk digunakan dengan file manifes.
11. Di bagian Cara kerjanya, pilih Buat kumpulan data.
12. Pilih tab Set data tunggal atau tab Kumpulan data pelatihan dan uji terpisah dan ikuti langkah-langkahnya.

Single dataset

1. Pilih Buat satu set data.
2. Di bagian Konfigurasi sumber gambar, pilih Impor gambar yang diberi label oleh SageMaker Ground Truth.
3. Untuk lokasi file.manifest, masukkan lokasi file manifes Anda.

Separate training and test datasets

1. Pilih Buat kumpulan data pelatihan dan kumpulan data pengujian.
2. Di bagian Detail kumpulan data Pelatihan, pilih Impor gambar berlabel SageMaker Ground Truth.
3. Di lokasi file.manifest, masukkan lokasi file manifes pelatihan Anda.
4. Di bagian Test dataset details, pilih Impor gambar berlabel SageMaker Ground Truth.
5. Di lokasi file.manifest, masukkan lokasi file manifes pengujian Anda.

> **Note**
>
> Kumpulan data pelatihan dan pengujian Anda dapat memiliki sumber gambar yang berbeda.

13. Pilih Kirim.
14. Ikuti langkah-langkah Melatih model Anda untuk melatih model Anda.

Amazon Lookout for Vision membuat kumpulan data di folder bucket Amazon S3. `datasets` `.manifest`File asli Anda tetap tidak berubah.

Membuat kumpulan data dengan file manifes (SDK)

Anda menggunakan CreateDatasetoperasi untuk membuat kumpulan data yang terkait dengan proyek Amazon Lookout for Vision.

Jika Anda ingin menggunakan satu set data untuk pelatihan dan pengujian, buat satu set data dengan `DatasetType` nilai yang disetel ke. `train` Selama pelatihan, kumpulan data dibagi secara internal untuk membuat kumpulan data pelatihan dan pengujian. Anda tidak memiliki akses ke kumpulan data pelatihan dan pengujian terpisah. Jika Anda menginginkan kumpulan data pengujian

terpisah, lakukan panggilan kedua `CreateDataset` dengan set `DatasetType` `test` nilai. Selama pelatihan, kumpulan data pelatihan dan pengujian digunakan untuk melatih dan menguji model.

Anda dapat menggunakan `DatasetSource` parameter secara opsional untuk menentukan lokasi file manifes format SageMaker Ground Truth yang digunakan untuk mengisi kumpulan data. Dalam hal ini, panggilan ke `CreateDataset` asinkron. Untuk memeriksa status terkini, panggil `DescribeDataset`. Untuk informasi selengkapnya, lihat Melihat kumpulan data Anda. Jika terjadi kesalahan validasi selama impor, nilai disetel ke CREATE_FAILED dan pesan status () `StatusMessage` disetel. `Status`

> **Tip**
>
> Jika Anda membuat kumpulan data dengan kumpulan data contoh memulai, gunakan file manifes (`getting-started/dataset-files/manifests/train.manifest`) tempat skrip dibuat. Langkah 1: Buat file manifes dan unggah gambar
> Jika Anda membuat kumpulan data dengan gambar contoh papan sirkuit, Anda memiliki dua opsi:
>
> 1. Buat file manifes menggunakan kode. Notebook Python Lookout for Vision Lab Amazon menunjukkan cara membuat file manifes untuk gambar contoh papan sirkuit. Atau, gunakan kode contoh Datasets di Repositori Contoh AWS Kode.
> 2. Jika Anda sudah menggunakan konsol Amazon Lookout for Vision untuk membuat kumpulan data dengan gambar contoh papan sirkuit, gunakan kembali file manifes yang dibuat untuk Anda oleh Amazon Lookout for Vision. Lokasi file manifes pelatihan dan pengujian adalah`s3://`*`bucket`*`/datasets/`*`project name`*`/`*`train or test`*`/manifests/output/output.manifest`.

Jika Anda tidak menentukan`DatasetSource`, dataset kosong akan dibuat. Dalam hal ini, panggilan ke `CreateDataset` sinkron. Nanti, Anda dapat memberi label gambar ke kumpulan data dengan memanggil UpdateDataset Entri. Untuk kode sampel, lihat Menambahkan lebih banyak gambar (SDK).

Jika Anda ingin mengganti kumpulan data, pertama-tama hapus kumpulan data yang ada dengan DeleteDatasetlalu buat kumpulan data baru dari jenis kumpulan data yang sama dengan memanggil. `CreateDataset` Untuk informasi selengkapnya, lihat Menghapus dataset.

Setelah Anda membuat kumpulan data, Anda dapat membuat model. Untuk informasi selengkapnya, lihat Melatih model (SDK).

Anda dapat melihat gambar berlabel (baris JSON) dalam kumpulan data dengan memanggil Entri. ListDataset Anda dapat menambahkan gambar berlabel dengan meneleponUpdateDatasetEntries.

Untuk melihat informasi tentang kumpulan data pengujian dan pelatihan, lihat. Melihat kumpulan data Anda

Untuk membuat dataset (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode contoh berikut untuk membuat dataset.

   CLI

   Ubah nilai berikut:

   - project-namedengan nama proyek yang ingin Anda kaitkan dengan dataset.
   - dataset-typeuntuk jenis dataset yang ingin Anda buat (trainatautest).
   - dataset-sourceke lokasi Amazon S3 dari file manifes.
   - Bucketdengan nama bucket Amazon S3 yang berisi file manifes.
   - Keyke jalur dan nama file dari file manifes di bucket Amazon S3.

```
aws lookoutvision create-dataset --project-name project\
  --dataset-type train or test\
  --dataset-source '{ "GroundTruthManifest": { "S3Object": { "Bucket": "bucket",
 "Key": "manifest file" } } }' \
  --profile lookoutvision-access
```

   Python

   Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
    @staticmethod
    def create_dataset(lookoutvision_client, project_name, manifest_file,
 dataset_type):
        """
        Creates a new Lookout for Vision dataset
```

```
        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        :param project_name: The name of the project in which you want to
                             create a dataset.
        :param bucket: The bucket that contains the manifest file.
        :param manifest_file: The path and name of the manifest file.
        :param dataset_type: The type of the dataset (train or test).
        """
        try:
            bucket, key = manifest_file.replace("s3://", "").split("/", 1)
            logger.info("Creating %s dataset type...", dataset_type)
            dataset = {
                "GroundTruthManifest": {"S3Object": {"Bucket": bucket, "Key":
 key}}
            }
            response = lookoutvision_client.create_dataset(
                ProjectName=project_name,
                DatasetType=dataset_type,
                DatasetSource=dataset,
            )
            logger.info("Dataset Status: %s", response["DatasetMetadata"]
["Status"])
            logger.info(
                "Dataset Status Message: %s",
                response["DatasetMetadata"]["StatusMessage"],
            )
            logger.info("Dataset Type: %s", response["DatasetMetadata"]
["DatasetType"])

            # Wait until either created or failed.
            finished = False
            status = ""
            dataset_description = {}
            while finished is False:
                dataset_description = lookoutvision_client.describe_dataset(
                    ProjectName=project_name, DatasetType=dataset_type
                )
                status = dataset_description["DatasetDescription"]["Status"]

                if status == "CREATE_IN_PROGRESS":
                    logger.info("Dataset creation in progress...")
                    time.sleep(2)
                elif status == "CREATE_COMPLETE":
                    logger.info("Dataset created.")
```

```
                    finished = True
                else:
                    logger.info(
                        "Dataset creation failed: %s",
                        dataset_description["DatasetDescription"]
["StatusMessage"],
                    )
                    finished = True

            if status != "CREATE_COMPLETE":
                message = dataset_description["DatasetDescription"]
["StatusMessage"]
                logger.exception("Couldn't create dataset: %s", message)
                raise Exception(f"Couldn't create dataset: {message}")

        except ClientError:
            logger.exception("Service error: Couldn't create dataset.")
            raise
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
/**
 * Creates an Amazon Lookout for Vision dataset from a manifest file.
 * Returns after Lookout for Vision creates the dataset.
 *
 * @param lfvClient    An Amazon Lookout for Vision client.
 * @param projectName  The name of the project in which you want to create a
 *                     dataset.
 * @param datasetType  The type of dataset that you want to create (train or
 *                     test).
 * @param bucket       The S3 bucket that contains the manifest file.
 * @param manifestFile The name and location of the manifest file within the S3
 *                     bucket.
 * @return DatasetDescription The description of the created dataset.
 */
public static DatasetDescription createDataset(LookoutVisionClient lfvClient,
                String projectName,
                String datasetType,
                String bucket,
```

```
                String manifestFile)
                throws LookoutVisionException, InterruptedException {

        logger.log(Level.INFO, "Creating {0} dataset for project {1}",
                        new Object[] { projectName, datasetType });

        // Build the request. If no bucket supplied, setup for empty dataset
creation.
        CreateDatasetRequest createDatasetRequest = null;

        if (bucket != null && manifestFile != null) {

                InputS3Object s3Object = InputS3Object.builder()
                                .bucket(bucket)
                                .key(manifestFile)
                                .build();

                DatasetGroundTruthManifest groundTruthManifest =
DatasetGroundTruthManifest.builder()
                                .s3Object(s3Object)
                                .build();

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

                createDatasetRequest = CreateDatasetRequest.builder()
                                .projectName(projectName)
                                .datasetType(datasetType)
                                .datasetSource(datasetSource)
                                .build();
        } else {
                createDatasetRequest = CreateDatasetRequest.builder()
                                .projectName(projectName)
                                .datasetType(datasetType)
                                .build();
        }

        lfvClient.createDataset(createDatasetRequest);

        DatasetDescription datasetDescription = null;

        boolean finished = false;
```

```
        // Wait until dataset is created, or failure occurs.
        while (!finished) {

                datasetDescription = describeDataset(lfvClient, projectName,
datasetType);

                switch (datasetDescription.status()) {
                        case CREATE_COMPLETE:
                                logger.log(Level.INFO, "{0}dataset created for
project {1}",
                                                new Object[] { datasetType,
projectName });
                                finished = true;
                                break;
                        case CREATE_IN_PROGRESS:
                                logger.log(Level.INFO, "{0} dataset creating for
project {1}",
                                                new Object[] { datasetType,
projectName });

                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case CREATE_FAILED:
                                logger.log(Level.SEVERE,
                                                "{0} dataset creation failed for
project {1}. Error {2}",
                                                new Object[] { datasetType,
projectName,

datasetDescription.statusAsString() });
                                finished = true;
                                break;
                        default:
                                logger.log(Level.SEVERE, "{0} error when
creating {1} dataset for project {2}",
                                                new Object[] { datasetType,
projectName,

datasetDescription.statusAsString() });
                                finished = true;
                                break;
```

```
                }
        }

        logger.log(Level.INFO, "Dataset info. Status: {0}\n Message: {1} }",
                        new Object[] { datasetDescription.statusAsString(),
                                        datasetDescription.statusMessage() });

        return datasetDescription;

}
```

3. Latih model Anda dengan mengikuti langkah-langkah diMelatih model (SDK).

Pelabelan gambar

Anda dapat menggunakan konsol Amazon Lookout for Vision untuk menambahkan atau memodifikasi label yang ditetapkan ke gambar dalam kumpulan data Anda. Jika Anda menggunakan SDK, label adalah bagian dari file manifes yang Anda berikan CreateDataset. Anda dapat memperbarui label untuk gambar dengan memanggil UpdateDatasetEntri. Untuk kode sampel, lihat Menambahkan lebih banyak gambar (SDK).

Memilih jenis model

Label yang Anda tetapkan ke gambar menentukan jenis model yang dibuat Lookout for Vision. Jika proyek Anda memiliki kumpulan data pengujian terpisah, beri label pada gambar dengan cara yang sama.

Model klasifikasi gambar

Untuk membuat model klasifikasi gambar, Anda mengklasifikasikan setiap gambar sebagai normal atau anomali. Untuk setiap gambar, lakukanMengklasifikasikan gambar (konsol).

Model segmentasi gambar

Untuk membuat model segmentasi gambar, Anda mengklasifikasikan setiap gambar sebagai normal atau anomali. Untuk setiap gambar anomali, Anda juga menentukan topeng piksel untuk setiap area anomali pada gambar dan label anomali untuk jenis anomali dalam topeng piksel. Misalnya, topeng biru pada gambar berikut menandai lokasi jenis anomali goresan pada mobil. Anda dapat menentukan lebih dari satu jenis label anomali dalam sebuah gambar. Untuk setiap gambar, lakukanSegmentasi gambar (konsol).

Mengklasifikasikan gambar (konsol)

Anda menggunakan konsol Lookout for Vision untuk mengklasifikasikan gambar dalam kumpulan data sebagai normal atau anomali. Gambar yang tidak diklasifikasi tidak digunakan untuk melatih model Anda.

Jika Anda membuat model segmentasi gambar, lewati prosedur ini dan lakukanSegmentasi gambar (konsol), yang mencakup langkah-langkah untuk mengklasifikasikan gambar.

> **Note**
>
> Jika Anda baru saja selesaiMembuat dataset Anda, konsol saat ini harus menampilkan dasbor model Anda dan Anda tidak perlu melakukan langkah 1 - 4.

Untuk mengklasifikasikan gambar Anda (konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Di panel navigasi kiri, pilih Proyek.
3. Di halaman Proyek, pilih proyek yang ingin Anda gunakan.
4. Di panel navigasi kiri proyek Anda, pilih Dataset.
5. Jika Anda memiliki kumpulan data pelatihan dan pengujian terpisah, pilih tab untuk kumpulan data yang ingin Anda gunakan.
6. Pilih Mulai pelabelan.
7. Pilih Pilih semua gambar di halaman ini.
8. Jika gambar normal, pilih Klasifikasi seperti biasa, jika tidak pilih Klasifikasi sebagai anomali. Sebuah label muncul di bawah setiap gambar.

9. Jika Anda perlu mengubah label untuk gambar, lakukan hal berikut:

    a. Pilih Anomali atau Normal di bawah gambar.

    b. Jika Anda tidak dapat menentukan label yang benar untuk gambar, perbesar gambar dengan memilih gambar di galeri.

> **Note**
>
> Anda dapat memfilter label gambar dengan memilih label yang diinginkan, atau status label, di bagian Filter.

10. Ulangi langkah 7-9 pada setiap halaman seperlunya sampai semua gambar dalam kumpulan data diberi label dengan benar.
11. Pilih Simpan perubahan.
12. Jika Anda telah selesai memberi label pada gambar Anda, Anda dapat melatih model Anda.

Segmentasi gambar (konsol)

Jika Anda membuat model segmentasi gambar, Anda harus mengklasifikasikan gambar sebagai normal atau anomali. Anda juga harus menambahkan informasi segmentasi ke gambar anomali. Untuk menentukan informasi segmentasi, pertama-tama Anda menentukan label anomali untuk setiap jenis anomali, seperti penyok atau goresan, yang ingin ditemukan oleh model Anda. Kemudian Anda menentukan topeng anomali dan label anomali untuk setiap anomali pada gambar anomali dalam kumpulan data Anda.

> **Note**
>
> Jika Anda membuat model klasifikasi gambar, Anda tidak perlu mengelompokkan gambar dan Anda tidak perlu menentukan label anomali.

Topik

- Menentukan label anomali
- Melabeli gambar
- Segmentasi gambar dengan alat anotasi

Menentukan label anomali

Anda menentukan label anomali untuk setiap jenis anomali yang ada di gambar kumpulan data.

Tentukan label anomali

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Di panel navigasi kiri, pilih Proyek.
3. Di halaman Proyek, pilih proyek yang ingin Anda gunakan.
4. Di panel navigasi kiri proyek Anda, pilih Dataset.
5. Dalam label anomali pilih Tambahkan label anomali. Jika sebelumnya Anda telah menambahkan label anomali, pilih Kelola.
6. Dalam kotak dialog, lakukan hal berikut:
    a. Masukkan label anomali yang ingin Anda tambahkan dan pilih Tambahkan label anomali.
    b. Ulangi langkah sebelumnya sampai Anda memasukkan setiap label anomali yang Anda ingin model Anda temukan.
    c. (Opsional) Pilih ikon edit untuk mengubah nama label.
    d. (Opsional) Pilih ikon hapus untuk menghapus label anomali baru. Anda tidak dapat menghapus jenis anomali yang saat ini digunakan oleh kumpulan data Anda.
7. Pilih Konfirmasi untuk menambahkan label anomali baru ke kumpulan data.

Setelah Anda menentukan label anomali, beri label gambar dengan melakukan. Melabeli gambar

Melabeli gambar

Untuk memberi label gambar untuk segmentasi gambar, klasifikasikan gambar sebagai normal atau anomali. Kemudian, gunakan alat anotasi untuk mengelompokkan gambar dengan menggambar topeng yang menutupi area untuk setiap jenis anomali yang ada dalam gambar.

Untuk memberi label pada gambar

1. Jika Anda memiliki kumpulan data pelatihan dan pengujian terpisah, pilih tab untuk kumpulan data yang ingin Anda gunakan.
2. Jika Anda belum melakukannya, tentukan jenis anomali untuk kumpulan data Anda dengan melakukan. Menentukan label anomali

3. Pilih Mulai pelabelan.
4. Pilih Pilih semua gambar di halaman ini.
5. Jika gambar normal, pilih Klasifikasi seperti biasa, jika tidak pilih Klasifikasi sebagai anomali.
6. Untuk mengubah label untuk satu gambar, pilih Normal atau Anomali di bawah gambar.

> **Note**
>
> Anda dapat memfilter label gambar dengan memilih label yang diinginkan, atau status label, di bagian Filter. Anda dapat mengurutkan berdasarkan skor kepercayaan di bagian Opsi penyortiran.

7. Untuk setiap gambar anomali, pilih gambar untuk membuka alat anotasi. Tambahkan informasi segmentasi dengan melakukanSegmentasi gambar dengan alat anotasi.
8. Pilih Simpan perubahan.
9. Jika Anda telah selesai memberi label pada gambar Anda, Anda dapat melatih model Anda.

Segmentasi gambar dengan alat anotasi

Anda menggunakan alat anotasi untuk mengelompokkan gambar dengan menandai area anomali dengan topeng.

Untuk menyegmentasikan gambar dengan alat anotasi

1. Buka alat anotasi dengan memilih gambar di galeri dataset. Jika perlu, pilih Mulai pelabelan untuk masuk ke mode pelabelan.
2. Di bagian Label anomali pilih label anomali yang ingin Anda tandai. Jika perlu, pilih Tambahkan label anomali untuk menambahkan label anomali baru.
3. Pilih alat menggambar di bagian bawah halaman dan gambar topeng yang menutupi area anomali dengan erat untuk label anomali. Gambar berikut adalah contoh topeng yang menutupi anomali dengan erat.

Berikut ini adalah contoh topeng buruk yang tidak menutupi anomali dengan erat.

4. Jika Anda memiliki lebih banyak gambar untuk disegmentasikan, pilih Berikutnya dan ulangi langkah 2 dan 3.
5. Pilih Kirim dan tutup untuk menyelesaikan segmentasi gambar.

Melatih model Anda

Setelah Anda membuat kumpulan data dan memberi label gambar, Anda dapat melatih model Anda. Sebagai bagian dari proses pelatihan, kumpulan data pengujian digunakan. Jika Anda memiliki proyek kumpulan data tunggal, gambar dalam kumpulan data secara otomatis dibagi menjadi kumpulan data pengujian dan kumpulan data pelatihan sebagai bagian dari proses pelatihan. Jika proyek Anda memiliki pelatihan dan kumpulan data pengujian, mereka digunakan untuk melatih dan menguji kumpulan data secara terpisah.

Setelah pelatihan selesai, Anda dapat mengevaluasi kinerja model dan melakukan perbaikan yang diperlukan. Untuk informasi selengkapnya, lihat Meningkatkan model Amazon Lookout for Vision.

Untuk melatih model Anda, Amazon Lookout for Vision membuat salinan pelatihan sumber dan gambar uji Anda. Secara default, gambar yang disalin dienkripsi dengan kunci yang dimiliki dan dikelola AWS. Anda juga dapat memilih untuk menggunakan kunci AWS Key Management Service (KMS) milik Anda sendiri. Untuk informasi selengkapnya, lihat Konsep AWS Key Management Service. Citra sumber Anda tidak terpengaruh.

Anda dapat menetapkan metadata ke model Anda dalam bentuk tag. Untuk informasi selengkapnya, lihat Model penandaan.

Setiap kali Anda melatih model, versi baru dari model dibuat. Jika Anda tidak lagi memerlukan versi model, Anda dapat menghapusnya. Untuk informasi selengkapnya, lihat Menghapus model.

Anda dikenakan biaya untuk jumlah waktu yang diperlukan untuk berhasil melatih model Anda. Untuk informasi selengkapnya, lihat Jam Pelatihan.

Untuk melihat model yang ada dalam sebuah proyek,Melihat model Anda.

> **Note**
>
> Jika Anda baru saja menyelesaikan Membuat dataset Anda atauMenambahkan gambar ke kumpulan data Anda. Konsol saat ini harus menampilkan dasbor model Anda dan Anda tidak perlu melakukan langkah 1 - 4.

Topik

- Melatih model (konsol)
- Melatih model (SDK)

Melatih model (konsol)

Prosedur berikut menunjukkan cara melatih model Anda menggunakan konsol.

Untuk melatih model Anda (konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Di panel navigasi kiri, pilih Proyek.
3. Di halaman Proyek, pilih proyek yang berisi model yang ingin Anda latih.
4. Pada halaman detail proyek, pilih Model kereta. Tombol model Kereta tersedia jika Anda memiliki cukup gambar berlabel untuk melatih model. Jika tombol tidak tersedia, tambahkan lebih banyak gambar sampai Anda memiliki cukup gambar berlabel.
5. (Opsional) Jika Anda ingin menggunakan kunci enkripsi AWS KMS Anda sendiri, lakukan hal berikut:
   a. Di Enkripsi data gambar pilih Sesuaikan pengaturan enkripsi (lanjutan).
   b. Di encryption.aws_kms_key masukkan Nama Sumber Daya Amazon (ARN) kunci Anda, atau pilih kunci AWS KMS yang ada. Untuk membuat kunci baru, pilih Buat kunci AWS IMS.
6. (Opsional) jika Anda ingin menambahkan tag ke model Anda lakukan hal berikut:
   a. Di bagian Tag, pilih Tambahkan tag baru.
   b. Masukkan yang berikut ini:
      i. Nama kunci di Key.
      ii. Nilai kunci dalam Nilai.
   c. Untuk menambahkan lebih banyak tag, ulangi langkah 6a dan 6b.
   d. (Opsional) Jika Anda ingin menghapus tag, pilih Hapus di samping tag yang ingin Anda hapus. Jika Anda menghapus tag yang disimpan sebelumnya, tag tersebut akan dihapus saat Anda menyimpan perubahan.
7. Pilih model Kereta.
8. Di Apakah Anda ingin melatih model Anda? kotak dialog, pilih Model kereta.
9. Dalam tampilan Model, Anda dapat melihat bahwa pelatihan telah dimulai dan Anda dapat memeriksa status saat ini dengan melihat `Status` kolom untuk versi model. Pelatihan model membutuhkan waktu beberapa saat untuk diselesaikan.

10. Saat pelatihan selesai, Anda dapat mengevaluasi kinerjanya. Untuk informasi selengkapnya, lihat Meningkatkan model Amazon Lookout for Vision.

Melatih model (SDK)

Anda menggunakan CreateModeloperasi untuk memulai pelatihan, pengujian, dan evaluasi model. Amazon Lookout for Vision melatih model menggunakan kumpulan data pelatihan dan pengujian yang terkait dengan proyek. Untuk informasi selengkapnya, lihat Membuat proyek (SDK).

Setiap kali Anda meneleponCreateModel, versi baru dari model dibuat. Tanggapan dari CreateModel termasuk versi model.

Anda dikenakan biaya untuk setiap pelatihan model yang berhasil. Gunakan parameter ClientToken input untuk membantu mencegah pengisian daya karena pengulangan pelatihan model yang tidak perlu atau tidak disengaja oleh pengguna Anda. ClientTokenadalah parameter input idempoten yang memastikan CreateModel hanya selesai sekali untuk satu set parameter tertentu — Panggilan berulang ke CreateModel dengan ClientToken nilai yang sama memastikan bahwa pelatihan tidak diulang. Jika Anda tidak memberikan nilaiClientToken, AWS SDK yang Anda gunakan menyisipkan nilai untuk Anda. Ini mencegah percobaan ulang setelah kesalahan jaringan memulai beberapa pekerjaan pelatihan, tetapi Anda harus memberikan nilai Anda sendiri untuk kasus penggunaan Anda sendiri. Untuk informasi lebih lanjut, lihat CreateModel.

Membutuhkan waktu beberapa saat untuk menyelesaikan pelatihan. Untuk memeriksa status saat ini, panggil DescribeModel dan teruskan nama proyek (ditentukan dalam panggilan keCreateProject) dan versi model. statusBidang menunjukkan status pelatihan model saat ini. Untuk kode sampel, lihat Melihat model Anda (SDK).

Jika pelatihan berhasil, Anda dapat mengevaluasi model. Untuk informasi selengkapnya, lihat Meningkatkan model Amazon Lookout for Vision.

Untuk melihat model yang telah Anda buat dalam sebuah proyek, hubungiListModels. Untuk kode sampel, lihat Melihat model Anda.

Untuk melatih model (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode contoh berikut untuk melatih model.

CLI

Ubah nilai berikut:

- `project-name`dengan nama proyek yang berisi model yang ingin Anda buat.
- `output-config`ke lokasi di mana Anda ingin menyimpan hasil pelatihan. Ganti nilai-nilai berikut:
  - `output bucket`dengan nama bucket Amazon S3 tempat Amazon Lookout for Vision menyimpan hasil pelatihan.
  - `output folder`dengan nama folder tempat Anda ingin menyimpan hasil pelatihan.
  - `Key`dengan nama kunci tag.
  - `Value`dengan nilai untuk diasosiasikan`tag_key`.

```
aws lookoutvision create-model --project-name "project name"\
  --output-config '{ "S3Location": { "Bucket": "output bucket", "Prefix":
 "output folder" } }'\
  --tags '[{"Key":"Key","Value":"Value"}]' \
  --profile lookoutvision-access
```

Python

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
    @staticmethod
    def create_model(
        lookoutvision_client,
        project_name,
        training_results,
        tag_key=None,
        tag_key_value=None,
    ):
        """
        Creates a version of a Lookout for Vision model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project in which you want to create
 a
```

```
                                model.
        :param training_results: The Amazon S3 location where training results
 are stored.
        :param tag_key: The key for a tag to add to the model.
        :param tag_key_value - A value associated with the tag_key.
        return: The model status and version.
        """
        try:
            logger.info("Training model...")
            output_bucket, output_folder = training_results.replace("s3://",
 "").split(
                "/", 1
            )
            output_config = {
                "S3Location": {"Bucket": output_bucket, "Prefix": output_folder}
            }
            tags = []
            if tag_key is not None:
                tags = [{"Key": tag_key, "Value": tag_key_value}]

            response = lookoutvision_client.create_model(
                ProjectName=project_name, OutputConfig=output_config, Tags=tags
            )

            logger.info("ARN: %s", response["ModelMetadata"]["ModelArn"])
            logger.info("Version: %s", response["ModelMetadata"]
["ModelVersion"])
            logger.info("Started training...")

            print("Training started. Training might take several hours to
 complete.")

            # Wait until training completes.
            finished = False
            status = "UNKNOWN"
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name,
                    ModelVersion=response["ModelMetadata"]["ModelVersion"],
                )
                status = model_description["ModelDescription"]["Status"]

                if status == "TRAINING":
                    logger.info("Model training in progress...")
```

```
                    time.sleep(600)
                    continue

                if status == "TRAINED":
                    logger.info("Model was successfully trained.")
                else:
                    logger.info(
                        "Model training failed: %s ",
                        model_description["ModelDescription"]["StatusMessage"],
                    )
                finished = True
        except ClientError:
            logger.exception("Couldn't train model.")
            raise
        else:
            return status, response["ModelMetadata"]["ModelVersion"]
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
/**
 * Creates an Amazon Lookout for Vision model. The function returns after model
 * training completes. Model training can take multiple hours to complete.
 * You are charged for the amount of time it takes to successfully train a
 model.
 * Returns after Lookout for Vision creates the dataset.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to create a
 *                    model.
 * @param description A description for the model.
 * @param bucket      The S3 bucket in which Lookout for Vision stores the
 *                    training results.
 * @param folder      The location of the training results within the S3
 *                    bucket.
 * @return ModelDescription The description of the created model.
 */
public static ModelDescription createModel(LookoutVisionClient lfvClient, String
 projectName,
                String description, String bucket, String folder)
```

```
                throws LookoutVisionException, InterruptedException {

        logger.log(Level.INFO, "Creating model for project: {0}.", new Object[]
{ projectName });

        // Setup input parameters.
        S3Location s3Location = S3Location.builder()
                        .bucket(bucket)
                        .prefix(folder)
                        .build();

        OutputConfig config = OutputConfig.builder()
                        .s3Location(s3Location)
                        .build();

        CreateModelRequest createModelRequest = CreateModelRequest.builder()
                        .projectName(projectName)
                        .description(description)
                        .outputConfig(config)
                        .build();

        // Create and train the model.
        CreateModelResponse response =
lfvClient.createModel(createModelRequest);

        String modelVersion = response.modelMetadata().modelVersion();
        boolean finished = false;
        DescribeModelResponse descriptionResponse = null;

        // Wait until training finishes or fails.

        do {
                DescribeModelRequest describeModelRequest =
DescribeModelRequest.builder()
                                .projectName(projectName)
                                .modelVersion(modelVersion)
                                .build();

                descriptionResponse =
lfvClient.describeModel(describeModelRequest);

                switch (descriptionResponse.modelDescription().status()) {
                        case TRAINED:
```

```
                                logger.log(Level.INFO, "Model training completed
for project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                finished = true;
                                break;

                        case TRAINING:
                                logger.log(Level.INFO,
                                                "Model training in progress for
project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                TimeUnit.SECONDS.sleep(60);

                                break;

                        case TRAINING_FAILED:
                                logger.log(Level.SEVERE,
                                                "Model training failed for for
project {0} version {1}.",
                                                new Object[] { projectName,
modelVersion });
                                finished = true;
                                break;

                        default:
                                logger.log(Level.SEVERE,
                                                "Unexpected error when training
model project {0} version {1}: {2}.",
                                                new Object[] { projectName,
modelVersion,

descriptionResponse.modelDescription()

.status() });
                                finished = true;
                                break;

                }
        } while (!finished);

        return descriptionResponse.modelDescription();
```

```
}
```

3. Saat pelatihan selesai, Anda dapat mengevaluasi kinerjanya. Untuk informasi selengkapnya, lihat Meningkatkan model Amazon Lookout for Vision.

Pelatihan model pemecahan masalah

Masalah dengan file manifes atau gambar pelatihan Anda dapat menyebabkan pelatihan model gagal. Sebelum melatih ulang model Anda, periksa potensi masalah berikut.

Warna label anomali tidak cocok dengan warna anomali pada gambar topeng

Jika Anda melatih model segmentasi gambar, warna label anomali dalam file manifes harus cocok dengan warna yang ada di gambar topeng. Baris JSON untuk gambar dalam file manifes memiliki metadata (`internal-color-map`) yang memberi tahu Amazon Lookout for Vision warna mana yang sesuai dengan label anomali. Misalnya, warna untuk label `scratch` anomali pada baris JSON berikut adalah. `#2ca02c`

```
{
    "source-ref": "s3://path-to-image",
    "anomaly-label": 1,
    "anomaly-label-metadata": {
        "class-name": "anomaly",
        "creation-date": "2021-10-12T14:16:45.668",
        "human-annotated": "yes",
        "job-name": "labeling-job/classification-job",
        "type": "groundtruth/image-classification",
        "confidence": 1
    },
    "anomaly-mask-ref": "s3://path-to-image",
    "anomaly-mask-ref-metadata": {
        "internal-color-map": {
            "0": {
                "class-name": "BACKGROUND",
                "hex-color": "#ffffff",
                "confidence": 0.0
            },
            "1": {
                "class-name": "scratch",
```

```
                "hex-color": "#2ca02c",
                "confidence": 0.0
            },
            "2": {
                "class-name": "dent",
                "hex-color": "#1f77b4",
                "confidence": 0.0
            }
        },
        "type": "groundtruth/semantic-segmentation",
        "human-annotated": "yes",
        "creation-date": "2021-11-23T20:31:57.758889",
        "job-name": "labeling-job/segmentation-job"
    }
}
```

Jika warna pada gambar topeng tidak cocok dengan nilai`hex-color`, pelatihan gagal dan Anda perlu memperbarui file manifes.

Untuk memperbarui nilai warna dalam file manifes

1. Menggunakan editor teks, buka file manifes yang Anda gunakan untuk membuat kumpulan data.
2. Untuk setiap baris JSON (gambar), periksa apakah warna (`hex-color`) di dalam `internal-color-map` bidang cocok dengan warna untuk label anomali pada gambar topeng.

   Anda bisa mendapatkan lokasi gambar topeng dari *`anomaly-mask`*`-ref` lapangan. Unduh gambar ke komputer Anda dan gunakan kode berikut untuk mendapatkan warna dalam gambar.

   ```
   from PIL import Image
   img = Image.open('path to local copy of mask file')
   colors = img.convert('RGB').getcolors() #this converts the mode to RGB
   for color in colors:
       print('#%02x%02x%02x' % color[1])
   ```

3. Untuk setiap gambar dengan penetapan warna yang salah, perbarui `hex-color` bidang di baris JSON untuk gambar.
4. Simpan file manifes pembaruan.
5. Hapus kumpulan data yang ada dari proyek.

6. Buat kumpulan data baru dalam proyek dengan file manifes yang diperbarui.
7. Latih modelnya.

Atau, untuk langkah 5 dan 6, Anda dapat memperbarui gambar individual dalam kumpulan data dengan memanggil operasi UpdateDatasetEntri dan menyediakan baris JSON yang diperbarui untuk gambar yang ingin Anda perbarui. Untuk kode sampel, lihat Menambahkan lebih banyak gambar (SDK).

Gambar topeng tidak dalam format PNG

Jika Anda melatih model segmentasi gambar, gambar topeng harus dalam format PNG. Jika Anda membuat kumpulan data dari file manifes, pastikan gambar topeng yang Anda referensikan *anomaly-mask*`-ref` adalah format PNG. Jika gambar topeng tidak dalam format PNG, Anda perlu mengonversinya ke format PNG. Tidak cukup mengganti nama ekstensi untuk file gambar menjadi. `.png`

Gambar topeng yang Anda buat di konsol Amazon Lookout for Vision atau dengan pekerjaan Ground SageMaker Truth dibuat dalam format PNG. Anda tidak perlu mengubah format gambar-gambar ini.

Untuk memperbaiki gambar topeng format non-PNG dalam file manifes

1. Menggunakan editor teks, buka file manifes yang Anda gunakan untuk membuat kumpulan data.
2. Untuk setiap baris JSON (gambar) pastikan gambar bahwa *anomaly-mask*`-ref` referensi gambar format PNG. Untuk informasi selengkapnya, lihat Membuat file manifes.
3. Simpan file manifes yang diperbarui.
4. Hapus kumpulan data yang ada dari proyek.
5. Buat kumpulan data baru dalam proyek dengan file manifes yang diperbarui.
6. Latih modelnya.

Label segmentasi atau klasifikasi tidak akurat atau hilang

Label yang hilang atau tidak akurat dapat menyebabkan pelatihan gagal atau membuat model yang berkinerja buruk. Kami menyarankan Anda memberi label pada semua gambar dalam kumpulan data Anda. Jika Anda tidak memberi label pada semua gambar dan pelatihan model gagal, atau model Anda berkinerja buruk, tambahkan lebih banyak gambar.

Periksa hal-hal berikut:

- Jika Anda membuat model segmentasi, masker harus menutupi anomali pada gambar kumpulan data Anda dengan ketat. Untuk memeriksa topeng di kumpulan data Anda, lihat gambar di galeri kumpulan data proyek. Jika perlu, gambar ulang topeng gambar. Untuk informasi selengkapnya, lihat Segmentasi gambar (konsol).

- Pastikan bahwa gambar anomali dalam gambar dataset Anda diklasifikasikan. Jika Anda membuat model segmentasi gambar, pastikan gambar anomali memiliki label anomali dan topeng gambar.

  Penting untuk diingat jenis model (segmentasi atau klasifikasi) yang Anda buat. Model klasifikasi tidak memerlukan topeng gambar pada gambar anomali. Jangan menambahkan masker ke gambar kumpulan data yang ditujukan untuk model klasifikasi.

  Untuk memperbarui label yang hilang

  1. Buka galeri dataset proyek.
  2. Filter gambar yang tidak berlabel untuk melihat gambar mana yang tidak memiliki label.
  3. Lakukan salah satu hal berikut ini:
     - Jika Anda membuat model klasifikasi gambar, klasifikasikan setiap gambar yang tidak berlabel.
     - Jika Anda membuat model segmentasi gambar, klasifikasikan dan segmentasikan setiap gambar yang tidak berlabel.
  4. Jika Anda membuat model segmentasi gambar, tambahkan masker ke gambar anomali rahasia yang tidak memiliki topeng.
  5. Latih modelnya.

Jika Anda memilih untuk tidak memperbaiki label yang buruk atau hilang, sebaiknya Anda menambahkan lebih banyak gambar berlabel atau menghapus gambar yang terpengaruh dari kumpulan data. Anda dapat menambahkan lebih banyak dari konsol atau dengan menggunakan operasi UpdateDatasetEntri. Untuk informasi selengkapnya, lihat Menambahkan gambar ke kumpulan data Anda.

Jika Anda memilih untuk menghapus gambar, Anda harus membuat ulang kumpulan data tanpa gambar yang terpengaruh, karena Anda tidak dapat menghapus gambar dari kumpulan data. Untuk informasi selengkapnya, lihat Menghapus gambar dari kumpulan data Anda.

Meningkatkan model Amazon Lookout for Vision

Selama latihan, Lookout for Vision menguji model Anda dengan set data pengujian dan menggunakan hasilnya untuk membuat metrik kinerja. Anda dapat menggunakan metrik kinerja untuk mengevaluasi performa model. Jika perlu, Anda dapat mengambil langkah-langkah untuk meningkatkan set data Anda dan kemudian melatih ulang model Anda.

Jika Anda puas dengan performa model Anda, Anda dapat mulai menggunakannya. Untuk informasi selengkapnya, lihat Menjalankan model Amazon Lookout for Vision Anda yang terlatih.

Topik

Langkah 1: Evaluasi performa model Anda

Anda dapat mengakses metrik kinerja dari konsol dan dari DescribeModeloperasi. Amazon Lookout for Vision menyediakan metrik kinerja ringkasan untuk kumpulan data pengujian dan hasil prediksi untuk semua gambar individual. Jika model Anda adalah model segmentasi, konsol juga menampilkan metrik ringkasan untuk setiap label anomali.

Untuk melihat metrik performa dan menguji prediksi gambar di konsol, lihatMelihat metrik kinerja (konsol). Untuk informasi tentang mengakses metrik kinerja dan prediksi gambar uji dengan`DescribeModel` operasi, lihatMelihat metrik kinerja (SDK).

Metrik klasifikasi citra

Amazon Lookout for Vision menyediakan metrik ringkasan berikut untuk klasifikasi yang dibuat model selama pengujian:

- Presisi
- Ingat
- Skor F1

Metrik model segmentasi gambar

Jika model adalah model segmentasi gambar, Amazon Lookout for Vision menyediakan metrik klasifikasi gambar ringkasan dan metrik kinerja ringkasan untuk setiap label anomali:

- Skor F1
- Persimpangan Rata-rata di atas Union (IoU)

Presisi

Metrik presisi menjawab pertanyaan - Ketika model memprediksi bahwa gambar mengandung anomali, seberapa sering prediksi itu benar?

Presisi adalah metrik yang berguna untuk situasi di mana biaya positif palsu tinggi. Misalnya, biaya melepas bagian mesin yang tidak rusak dari mesin rakitan.

Amazon Lookout for Vision memberikan nilai metrik presisi ringkasan untuk seluruh kumpulan data pengujian.

Presisi adalah fraksi dari anomali yang diprediksi dengan benar (true positive) atas semua anomali yang diprediksi (true and false positive). Rumus untuk presisi adalah sebagai berikut.

Nilai presisi = benar positif/(benar positif+positif palsu)

Nilai yang mungkin untuk rentang presisi dari 0-1. Konsol Amazon Lookout for Vision menampilkan presisi sebagai nilai persentase (0-100).

Nilai presisi yang lebih tinggi menunjukkan bahwa lebih banyak anomali yang diprediksi benar. Misalnya, model Anda memprediksi bahwa 100 gambar itu anomali. Jika 85 prediksi benar (positif sejati) dan 15 salah (positif palsu), presisi dihitung sebagai berikut:

85 benar positif/ (85 benar positif+15 positif palsu) = 0,85 nilai presisi

Namun, jika model hanya memprediksi 40 gambar dengan benar dari 100 prediksi anomali, nilai presisi yang dihasilkan lebih rendah pada 0,40 (yaitu, 40/(40+ 60) = 0,40). Dalam hal ini, model Anda membuat prediksi yang lebih salah daripada prediksi yang benar. Untuk memperbaikinya, pertimbangkan untuk melakukan perbaikan pada model Anda. Untuk informasi selengkapnya, lihat Langkah 2: Meningkatkan model Anda.

Untuk informasi selengkapnya, lihat Precision dan recall.

Ingat

Metrik penarikan menjawab pertanyaan - Dari jumlah total gambar anomali dalam kumpulan data uji, berapa banyak yang diprediksi dengan benar sebagai anomali?

Metrik penarikan berguna untuk situasi di mana biaya negatif palsu tinggi. Misalnya, ketika biaya tidak menghapus bagian yang rusak tinggi. Amazon Lookout for Vision menyediakan nilai metrik penarikan ringkasan untuk seluruh kumpulan data pengujian.

Ingat adalah fraksi dari gambar uji anomali yang terdeteksi dengan benar. Ini adalah ukuran seberapa sering model dapat memprediksi gambar anomali dengan benar, ketika itu benar-benar ada dalam gambar set data pengujian Anda. Rumus untuk recall dihitung sebagai berikut:

Ingat nilai = benar positif/(benar positif+negatif palsu)

Rentang untuk penarikan adalah 0-1. Konsol Amazon Lookout for Vision menampilkan penarikan kembali sebagai nilai persentase (0-100).

Nilai penarikan yang lebih tinggi menunjukkan bahwa lebih banyak gambar anomali diidentifikasi dengan benar. Sebagai contoh, perhatikan kumpulan data uji berisi 100 citra anomali. Jika model mendeteksi 90 dari 100 gambar anomali dengan benar, maka ingatnya adalah sebagai berikut:

90 true positives/(90 benar positif+10 negatif palsu) = 0,90 nilai recall

Nilai penarikan 0,90 menunjukkan bahwa model Anda memprediksi dengan benar sebagian besar gambar anomali dalam kumpulan data pengujian. Jika model hanya memprediksi 20 gambar anomali dengan benar, penarikan kembali lebih rendah pada 0,20 (yaitu, 20/(20 + 80) = 0,20).

Dalam kasus ini, Anda harus mempertimbangkan untuk melakukan perbaikan pada model Anda. Untuk informasi selengkapnya, lihat Langkah 2: Meningkatkan model Anda.

Untuk informasi selengkapnya, lihat Precision dan recall.

Skor F1

Amazon Lookout for Vision memberikan skor kinerja model rata-rata untuk kumpulan data pengujian. Secara khusus, kinerja model untuk klasifikasi anomali diukur dengan metrik skor F1, yang merupakan mean harmonik dari skor presisi dan penarikan.

Skor F1 adalah ukuran agregat yang memperhitungkan presisi dan penarikan. Skor performa model adalah nilai antara 0 dan 1. Semakin tinggi nilainya, semakin baik performa model untuk penarikan dan presisi. Misalnya, untuk model dengan presisi 0,9 dan penarikan 1,0, skor F1 adalah 0,947.

Jika model tidak berkinerja baik, misalnya, dengan presisi rendah 0,30 dan penarikan tinggi 1,0, skor F1 adalah 0,46. Demikian pula, jika presisi tinggi (0,95) dan penarikan kembali rendah (0,20), skor F1 adalah 0,33. Dalam kedua kasus, skor F1 rendah, yang menunjukkan masalah dengan model.

Untuk informasi selengkapnya, lihat Skor F1.

Persimpangan Rata-rata di atas Union (IoU)

Persentase rata-rata tumpang tindih antara topeng anomali dalam gambar uji dan topeng anomali yang diprediksi model untuk gambar uji. Amazon Lookout for Vision mengembalikan IoU Rata-rata untuk setiap label anomali dan hanya dikembalikan oleh model segmentasi gambar.

Nilai persentase rendah menunjukkan bahwa model tidak secara akurat mencocokkan topeng yang diprediksi untuk label dengan topeng dalam gambar uji.

Gambar berikut memiliki IoU rendah. Masker oranye adalah prediksi dari model dan tidak menutupi topeng biru yang mewakili topeng dalam gambar uji.

Gambar berikut memiliki IoU yang lebih tinggi. Masker biru (gambar uji) tertutup rapat oleh topeng oranye (topeng yang diprediksi).

Hasil pengujian

Selama pengujian, model memprediksi klasifikasi untuk setiap gambar pengujian dalam set data pengujian. Hasil untuk setiap prediksi dibandingkan dengan label (normal atau anomali) dari gambar uji yang sesuai sebagai berikut:

- Dengan benar memprediksi bahwa gambar itu anomali dianggap positif sejati.
- Salah memprediksi bahwa gambar itu anomali dianggap positif palsu.
- Memprediksi dengan benar bahwa gambar itu normal dianggap negatif sejati.
- Salah memprediksi bahwa gambar normal dianggap negatif palsu.

Jika modelnya adalah model segmentasi, model tersebut juga memprediksi topeng dan label anomali untuk lokasi anomali pada gambar uji.

Amazon Lookout for Vision menggunakan hasil perbandingan untuk menghasilkan metrik kinerja.

Langkah 2: Meningkatkan model Anda

Metrik kinerja mungkin menunjukkan bahwa Anda dapat meningkatkan model Anda. Misalnya, jika model tidak mendeteksi semua anomali dalam kumpulan data pengujian, model Anda memiliki daya ingat rendah (yaitu, metrik penarikan memiliki nilai rendah). Untuk memperbaiki model Anda, pertimbangkan hal berikut:

- Periksa apakah gambar set data pelatihan dan pengujian diberi label dengan benar.
- Kurangi variabilitas kondisi pengambilan gambar seperti pencahayaan dan pose objek, dan latih model Anda pada objek dengan jenis yang sama.
- Pastikan bahwa gambar Anda hanya menampilkan konten yang diperlukan. Misalnya, jika proyek Anda mendeteksi anomali di bagian mesin, pastikan tidak ada objek lain dalam gambar Anda.
- Tambahkan lebih banyak gambar berlabel ke set data kereta dan uji Anda. Jika kumpulan data pengujian Anda memiliki penarikan dan presisi yang sangat baik tetapi model berkinerja buruk saat diterapkan, kumpulan data pengujian Anda mungkin tidak cukup representatif dan Anda perlu memperpanjangnya.
- Jika kumpulan data pengujian Anda menghasilkan penarikan dan presisi yang buruk, pertimbangkan seberapa baik anomali dan kondisi pengambilan gambar dalam kumpulan data pelatihan dan pengujian cocok. Jika gambar latihan Anda tidak mewakili anomali dan kondisi yang

diharapkan, tetapi gambar dalam gambar pengujian, tambahkan gambar ke set data pelatihan pelatihan dengan anomali dan kondisi yang diharapkan. Jika gambar set data pengujian tidak dalam kondisi yang diharapkan, tetapi gambar latihannya, perbarui set data pengujian.

Untuk informasi selengkapnya, lihat Menambahkan lebih banyak gambar. Cara alternatif untuk menambahkan gambar berlabel ke set data latihan Anda adalah dengan menjalankan tugas deteksi uji coba dan memverifikasi hasilnya. Anda kemudian dapat menambahkan gambar terverifikasi ke set data pelatihan. Untuk informasi selengkapnya, lihat Memverifikasi model Anda dengan tugas deteksi uji coba.

- Pastikan Anda memiliki gambar normal dan anomali yang cukup beragam dalam kumpulan data pelatihan dan pengujian Anda. Gambar harus mewakili jenis gambar normal dan anomali yang akan ditemui model Anda. Misalnya, saat menganalisis papan sirkuit, gambar normal Anda harus mewakili variasi posisi dan penyolderan komponen, seperti resistor dan transistor. Gambar anomali harus mewakili berbagai jenis anomali yang mungkin ditemui sistem, seperti komponen yang salah tempat atau hilang.
- Jika model Anda memiliki IoU Rata-rata rendah untuk jenis anomali yang terdeteksi, periksa output topeng dari model segmentasi. Untuk beberapa kasus penggunaan, seperti goresan, model mungkin mengeluarkan goresan yang sangat dekat dengan goresan groundtruth pada gambar uji, tetapi dengan tumpang tindih piksel rendah. Misalnya, dua garis parallel yang berjarak 1 piksel. Dalam kasus tersebut, Rata-rata IOU adalah indikator yang tidak dapat diandalkan untuk mengukur keberhasilan prediksi.
- Jika ukuran gambar kecil, atau resolusi gambar rendah, pertimbangkan untuk menangkap gambar pada resolusi yang lebih tinggi. Dimensi gambar dapat berkisar dari 64 x 64 piksel hingga 4096 piksel X 4096 piksel.
- Jika ukuran anomali kecil, pertimbangkan untuk membagi gambar menjadi ubin terpisah dan gunakan gambar ubin untuk pelatihan dan pengujian. Ini memungkinkan model melihat cacat pada ukuran yang lebih besar dalam gambar.

Setelah Anda meningkatkan set data pelatihan dan pengujian, latih ulang dan evaluasi ulang model Anda. Untuk informasi selengkapnya, lihat Melatih model Anda.

Jika metrik menunjukkan bahwa model Anda memiliki kinerja yang dapat diterima, Anda dapat memverifikasi kinerjanya dengan menambahkan hasil tugas deteksi uji coba ke set data pengujian. Setelah pelatihan ulang, metrik kinerja harus mengkonfirmasi metrik kinerja dari pelatihan sebelumnya. Untuk informasi selengkapnya, lihat Memverifikasi model Anda dengan tugas deteksi uji coba.

Melihat metrik kinerja

Anda bisa mendapatkan metrik kinerja dari konsol dan dengan memanggil`DescribeModel` operasi.

Topik

- Melihat metrik kinerja (konsol)
- Melihat metrik kinerja (SDK)

Melihat metrik kinerja (konsol)

Setelah latihan selesai, konsol menampilkan metrik kinerja.

Konsol Amazon Lookout for Vision menunjukkan metrik kinerja berikut untuk klasifikasi yang dibuat selama pengujian:

- Presisi
- Ingat
- Skor F1

Jika model adalah model segmentasi, konsol juga menampilkan metrik kinerja berikut untuk setiap label anomali:

- Jumlah gambar uji di mana label anomali ditemukan.
- Skor F1
- Persimpangan Rata-rata di atas Union (IoU)

Bagian ikhtisar hasil pengujian menunjukkan kepada Anda prediksi total yang benar dan salah untuk gambar dalam kumpulan data pengujian. Anda juga dapat melihat penetapan label yang diprediksi dan aktual untuk masing-masing gambar dalam kumpulan data pengujian.

Prosedur berikut menunjukkan cara mendapatkan metrik kinerja dari tampilan daftar model proyek.

Untuk melihat metrik kinerja (konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.

3. Di panel navigasi sebelah kiri, pilih Proyek.
4. Di tampilan proyek, pilih proyek yang berisi versi model yang ingin Anda lihat.
5. Di panel navigasi sebelah kiri, di bawah nama proyek, pilih Model.
6. Di tampilan daftar model, pilih versi model yang ingin Anda lihat.
7. Pada halaman detail model, lihat metrik kinerja pada tab Metrik kinerja.
8. Perhatikan hal berikut:

    a. Bagian Metrik kinerja model berisi metrik model keseluruhan (presisi, penarikan, skor F1) untuk prediksi klasifikasi yang dibuat model untuk gambar pengujian.

    b. Jika model adalah model segmentasi gambar, bagian Kinerja per label berisi jumlah gambar uji tempat label anomali ditemukan. Anda juga melihat metrik (skor F1, Rata-rata IoU) untuk setiap label anomali.

    c. Bagian ikhtisar hasil pengujian memberikan hasil untuk setiap gambar pengujian yang digunakan Lookout for Vision untuk mengevaluasi model. Ini mencakup hal-hal berikut:

        - Jumlah total prediksi klasifikasi yang benar (true positive) dan salah (false negative) (normal atau anomali) untuk semua gambar uji.
        - Prediksi klasifikasi untuk setiap gambar uji. Jika Anda melihat Benar di bawah gambar, klasifikasi yang diprediksi cocok dengan klasifikasi sebenarnya untuk gambar. Jika tidak, model tidak mengklasifikasikan gambar dengan benar.
        - Dengan model segmentasi gambar, Anda melihat label anomali yang ditetapkan model ke gambar dan topeng pada gambar yang cocok dengan warna label anomali.

Melihat metrik kinerja (SDK)

Anda dapat menggunakan DescribeModeloperasi untuk mendapatkan metrik kinerja ringkasan (klasifikasi) untuk model, manifes evaluasi, dan hasil evaluasi untuk model.

Mendapatkan metrik performa ringkasan

Metrik kinerja ringkasan untuk prediksi klasifikasi yang dibuat oleh model selama pengujian (PresisiIngat,, danSkor F1) ditampilkan di`Performance` bidang yang dikembalikan oleh panggilan ke`DescribeModel`.

```
        "Performance": {
            "F1Score": 0.8,
```

```
            "Recall": 0.8,
            "Precision": 0.9
        },
```

`Performance`Bidang tidak menyertakan metrik kinerja label anomali yang dikembalikan oleh model segmentasi. Anda bisa mendapatkannya dari`EvaluationResult` lapangan. Untuk informasi selengkapnya, lihat Meninjau hasil evaluasi.

Untuk informasi tentang metrik kinerja ringkasan, lihatLangkah 1: Evaluasi performa model Anda. Untuk kode sampel, lihat Melihat model Anda (SDK).

Menggunakan manifes evaluasi

Manifes evaluasi menyediakan metrik prediksi pengujian untuk masing-masing gambar yang digunakan untuk menguji model. Untuk setiap gambar dalam kumpulan data pengujian, baris JSON berisi informasi pengujian asli (ground truth) dan prediksi model untuk gambar. Amazon Lookout for Vision menyimpan manifes evaluasi di bucket Amazon S3. Anda bisa mendapatkan lokasi dari`EvaluationManifest` lapangan dalam respons dari`DescribeModel` operasi.

```
"EvaluationManifest": {
            "Bucket": "lookoutvision-us-east-1-nnnnnnnnnn",
            "Key": "my-sdk-project-model-output/EvaluationManifest-my-sdk-
project-1.json"
        }
```

Format nama file adalah`EvaluationManifest-`*`project name`*`.json`. Untuk kode sampel, lihat Melihat model Anda.

Dalam contoh baris JSON berikut,`class-name` adalah kebenaran dasar untuk isi gambar. Dalam contoh ini gambar berisi anomali. `confidence`Bidang menunjukkan keyakinan yang dimiliki Amazon Lookout for Vision dalam prediksi.

```
{
    "source-ref"*: "s3://customerbucket/path/to/image.jpg",
    "source-ref-metadata": {
        "creation-date": "2020-05-22T21:33:37.201882"
    },

    // Test dataset ground truth
```

```
    "anomaly-label": 1,
    "anomaly-label-metadata": {
        "class-name": "anomaly",
        "type": "groundtruth/image-classification",
        "human-annotated": "yes",
        "creation-date": "2020-05-22T21:33:37.201882",
        "job-name": "labeling-job/anomaly-detection"
    },
    // Anomaly label detected by Lookout for Vision
    "anomaly-label-detected": 1,
    "anomaly-label-detected-metadata": {
        "class-name": "anomaly",
        "confidence": 0.9,
        "type": "groundtruth/image-classification",
        "human-annotated": "no",
        "creation-date": "2020-05-22T21:33:37.201882",
        "job-name": "training-job/anomaly-detection",
        "model-arn": "lookoutvision-some-model-arn",
        "project-name": "lookoutvision-some-project-name",
        "model-version": "lookoutvision-some-model-version"
    }
}
```

Meninjau hasil evaluasi

Hasil evaluasi memiliki metrik kinerja agregat (klasifikasi) berikut untuk seluruh rangkaian gambar pengujian:

- Presisi
- Ingat
- Kurva ROC (tidak ditampilkan di konsol)
- Presisi rata-rata (tidak ditampilkan di konsol)
- Skor F1

Hasil evaluasi juga mencakup jumlah gambar yang digunakan untuk pelatihan dan pengujian model.

Jika model adalah model segmentasi, hasil evaluasi juga mencakup metrik berikut untuk setiap label anomali yang ditemukan dalam kumpulan data pengujian:

- Presisi

- Ingat
- Skor F1
- Persimpangan Rata-rata di atas Union (IoU)

Amazon Lookout for Vision menyimpan hasil evaluasi di bucket Amazon S3. Anda bisa mendapatkan lokasi dengan memeriksa nilai`EvaluationResult` dalam respon dari`DescribeModel` operasi.

```
"EvaluationResult": {
            "Bucket": "lookoutvision-us-east-1-nnnnnnnnnn",
            "Key": "my-sdk-project-model-output/EvaluationResult-my-sdk-project-1.json"
        }
```

Format nama file adalah`EvaluationResult-`*`project name`*`.json`. Sebagai contoh, lihat Melihat model Anda.

Skema berikut menunjukkan hasil evaluasi.

```
   {
        "Version": 1,
        "EvaluationDetails":
        {
            "ModelArn": "string", // The Amazon Resource Name (ARN) of the model
version.
            "EvaluationEndTimestamp": "string",   // The UTC date and time that
evaluation finished.
            "NumberOfTrainingImages": int,        // The number of images that were
successfully used for training.
            "NumberOfTestingImages": int          // The number of images that were
successfully used for testing.
        },
        "AggregatedEvaluationResults":
        {
            "Metrics":
            {                       //Classification metrics.
                "ROCAUC": float,            // ROC area under the curve.
                "AveragePrecision": float,  // The average precision of the model.
                "Precision": float,         // The overall precision of the model.
                "Recall": float,            // The overall recall of the model.
                "F1Score": float,           // The overal F1 score for the model.

                "PixelAnomalyClassMetrics":     //Segmentation metrics.
```

```
                [
                    {
                        "Precision": float,       // The precision for the anomaly
label.
                        "Recall": float,          // The recall for the anomaly label.
                        "F1Score": float,         // The F1 score for the anomaly
label.
                        "AIOU" : float,           // The average Intersection Over
Union for the anomaly label.
                        "ClassName": "string"   // The anomaly label.
                    }
                ]
            }
        }
    }
```

Memverifikasi model Anda dengan tugas deteksi uji coba

Jika Anda ingin memverifikasi atau meningkatkan kualitas model Anda, Anda dapat menjalankan tugas deteksi uji coba. Tugas deteksi percobaan mendeteksi anomali dalam gambar baru yang Anda berikan.

Anda dapat memverifikasi hasil deteksi dan menambahkan gambar terverifikasi ke set data Anda. Jika Anda memiliki kumpulan data pelatihan dan pengujian terpisah, gambar yang diverifikasi akan ditambahkan ke set data pelatihan.

Anda dapat memverifikasi citra dari komputer atau citra lokal yang terletak di bucket Amazon S3. Jika Anda ingin menambahkan gambar terverifikasi ke set data, gambar yang terletak di bucket S3 harus berada dalam bucket S3 yang sama dengan gambar di kumpulan data Anda.

> **Note**
>
> Untuk menjalankan tugas deteksi uji coba, pastikan bucket S3 Anda mengaktifkan versi. Untuk informasi selengkapnya, lihat Menggunakan versi. Bucket konsol dibuat dengan versi diaktifkan.

Secara default gambar Anda dienkripsi dengan kunci yang dimiliki dan dikelola AWS. Anda juga dapat memilih untuk menggunakan kunci AWS Key Management Service (KMS). Untuk informasi selengkapnya, lihat Konsep AWS Key Management Service.

Topik

Menjalankan tugas deteksi percobaan

Lakukan langkah-langkah berikut untuk menjalankan tugas deteksi uji coba.

Untuk menjalankan deteksi uji coba (konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi sebelah kiri, pilih Proyek.
4. Di tampilan proyek, pilih proyek yang berisi versi model yang ingin Anda lihat.
5. Di panel navigasi sebelah kiri, di bawah nama proyek, pilih Deteksi uji coba.
6. Dalam tampilan deteksi uji coba, pilih Jalankan deteksi uji coba.
7. Pada halaman Jalankan deteksi uji coba, masukkan nama untuk tugas deteksi uji coba di Nama tugas.
8. Di Pilih model, pilih versi model yang ingin Anda gunakan.
9. Impor gambar sesuai dengan sumber gambar sebagai berikut:
   - Jika Anda mengimpor gambar sumber dari bucket Amazon S3, masukkan URI S3.

     > **Tip**
     >
     > Jika Anda menggunakan gambar contoh Memulai, gunakan folder extra_images. URI Amazon S3 adalah`s3://your bucket/circuitboard/extra_images`.

   - Jika Anda mengunggah gambar dari komputer, tambahkan gambar setelah Anda memilih Deteksi anomali.
10. (Opsional) Jika Anda ingin menggunakan kunci enkripsi AWS KMS Anda sendiri, lakukan hal berikut:

    a. Untuk enkripsi data gambar, pilih Sesuaikan pengaturan enkripsi (lanjutan).

b. Di encryption.aws_kms_key, masukkan Amazon Resource Name (ARN) dari kunci Anda, atau pilih kunci AWS KMS yang ada. Untuk membuat kunci baru, pilih Buat kunci AWS IMS.

11. Pilih Deteksi anomali, lalu pilih Jalankan deteksi uji coba untuk memulai tugas deteksi uji coba.
12. Periksa status saat ini dalam tampilan deteksi uji coba. Deteksi uji coba mungkin membutuhkan waktu beberapa saat untuk menyelesaikan.

Memverifikasi hasil deteksi uji coba

Memverifikasi hasil deteksi uji coba dapat membantu Anda meningkatkan model Anda.

Jika metrik kinerja buruk, tingkatkan model Anda dengan menjalankan deteksi uji coba, lalu tambahkan gambar terverifikasi ke set data (set data pelatihan, jika Anda memiliki kumpulan data terpisah).

Jika metrik kinerja model bagus, tetapi hasil deteksi uji coba buruk, Anda dapat meningkatkan model dengan menambahkan gambar terverifikasi ke set data (set data pelatihan). Jika Anda memiliki kumpulan data pengujian terpisah, pertimbangkan untuk menambahkan lebih banyak gambar ke set data pengujian.

Setelah Anda menambahkan gambar terverifikasi ke set data Anda, latih ulang dan evaluasi ulang model Anda. Untuk informasi selengkapnya, lihat Melatih model Anda.

Untuk memverifikasi hasil deteksi uji coba

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Di panel navigasi sebelah kiri, pilih Proyek.
3. Di halaman Proyek, pilih proyek yang ingin Anda gunakan. Dasbor untuk proyek Anda ditampilkan.
4. Di panel navigasi sebelah kiri, pilih Deteksi uji coba.
5. Pilih deteksi uji coba yang ingin Anda verifikasi.
6. Pada halaman deteksi uji coba, pilih Verifikasi prediksi mesin.
7. Pilih Pilih semua gambar di halaman ini.
8. Jika prediksi benar, pilih Verifikasi sebagai benar. Jika tidak, pilih Verifikasi sebagai salah. Skor kepercayaan prediksi dan prediksi ditampilkan di bawah setiap gambar.
9. Untuk mengubah label untuk citra, Anda dapat melakukan hal-hal berikut:

a. Pilih Benar atau Salah di bawah gambar.

b. Jika Anda tidak dapat menentukan label yang benar untuk gambar, perbesar gambar dengan memilih gambar di galeri.

> **Note**
>
> Anda dapat memfilter label gambar dengan memilih label yang diinginkan, atau status label, di bagian Filter. Anda dapat mengurutkan berdasarkan skor kepercayaan di bagian Sorting options.

10. Jika model Anda adalah model segmentasi dan label topeng atau anomali untuk gambar salah, pilih Area anomali di bawah gambar dan buka alat anotasi. Perbarui informasi segmentasi dengan melakukanMemperbaiki label segmentasi dengan alat anotasi.
11. Ulangi langkah 7-10 pada setiap halaman seperlunya sampai semua gambar telah diverifikasi.
12. Pilih Tambahkan gambar terverifikasi ke set data. Jika Anda memiliki kumpulan data terpisah, gambar akan ditambahkan ke set data pelatihan.
13. Latih kembali model Anda. Untuk informasi selengkapnya, lihat the section called “Melatih model Anda”.

Memperbaiki label segmentasi dengan alat anotasi

Anda menggunakan alat anotasi untuk mengelompokkan gambar dengan menandai area anomali dengan topeng.

Untuk memperbaiki label segmentasi untuk gambar dengan alat anotasi

1. Buka alat anotasi dengan memilih area anomali di bawah gambar di galeri kumpulan data.
2. Jika label anomali untuk masker tidak benar, pilih topeng dan kemudian pilih label anomali yang benar di bawah label Anomali. Jika perlu, pilih Tambahkan label anomali untuk menambahkan label anomali baru.
3. Jika topeng tidak benar, pilih alat gambar di bagian bawah halaman dan gambar masker yang menutupi area anomali untuk label anomali. Gambar berikut adalah contoh topeng yang menutupi anomali dengan erat.

Berikut ini adalah contoh topeng yang buruk yang tidak menutupi anomali.

4. Jika Anda memiliki lebih banyak gambar untuk diperbaiki, pilih Berikutnya dan ulangi langkah 2 dan 3.
5. Pilih Kirim dan tutup untuk menyelesaikan memperbarui gambar.

Menjalankan model Amazon Lookout for Vision Anda yang terlatih

Untuk mendeteksi anomali dalam gambar dengan model Anda, Anda harus terlebih dahulu memulai model Anda dengan operasi. StartModel Konsol Amazon Lookout for Vision AWS CLI menyediakan perintah yang dapat Anda gunakan untuk memulai dan menghentikan model Anda. Bagian ini mencakup contoh kode yang dapat Anda gunakan.

Setelah model Anda dimulai, Anda dapat menggunakan `DetectAnomalies` operasi untuk mendeteksi anomali dalam gambar. Untuk informasi selengkapnya, lihat Mendeteksi anomali dalam gambar.

Topik

Unit inferensi

Saat memulai model, Amazon Lookout for Vision menyediakan minimal satu sumber daya komputasi, yang dikenal sebagai unit inferensi. Anda menentukan jumlah unit inferensi yang akan digunakan dalam parameter `MinInferenceUnits` input ke `StartModel` API. Alokasi default untuk model adalah 1 unit inferensi.

> **Important**
>
> Anda dikenakan biaya untuk jumlah jam yang dijalankan model Anda dan untuk jumlah unit inferensi yang digunakan model Anda saat berjalan, berdasarkan cara Anda mengonfigurasi pengoperasian model Anda. Misalnya, jika Anda memulai model dengan dua unit inferensi dan menggunakan model selama 8 jam, Anda akan dikenakan biaya selama 16 jam inferensi (8 jam waktu berjalan * dua unit inferensi). Untuk informasi selengkapnya, lihat Amazon Lookout for Vision Pricing. Jika Anda tidak secara eksplisit menghentikan model Anda dengan menelepon StopModel, Anda dikenakan biaya bahkan jika Anda tidak secara aktif menganalisis gambar dengan model Anda.

Transaksi per detik (TPS) yang didukung oleh unit inferensi tunggal dipengaruhi oleh hal-hal berikut:

- Algoritma yang digunakan Lookout for Vision untuk melatih model. Saat Anda melatih model, beberapa model dilatih. Lookout for Vision memilih model dengan kinerja terbaik berdasarkan ukuran dataset dan komposisi gambar normal dan anomali.
- Gambar beresolusi lebih tinggi membutuhkan lebih banyak waktu untuk analisis.
- Gambar berukuran lebih kecil (diukur dalam MB) dianalisis lebih cepat daripada gambar yang lebih besar.

Mengelola throughput dengan unit inferensi

Anda dapat menambah atau mengurangi throughput model Anda tergantung pada permintaan pada aplikasi Anda. Untuk meningkatkan throughput, gunakan unit inferensi tambahan. Setiap unit inferensi tambahan meningkatkan kecepatan pemrosesan Anda dengan satu unit inferensi. Untuk informasi tentang menghitung jumlah unit inferensi yang Anda butuhkan, lihat Menghitung unit inferensi untuk Amazon Rekognition Custom Labels dan Amazon Lookout for Vision model. Jika Anda ingin mengubah throughput model yang didukung, Anda memiliki dua opsi:

Menambahkan atau menghapus unit inferensi secara manual

Hentikan model dan kemudian restart dengan jumlah unit inferensi yang diperlukan. Kerugian dengan pendekatan ini adalah model tidak dapat menerima permintaan saat memulai ulang dan tidak dapat digunakan untuk menangani lonjakan permintaan. Gunakan pendekatan ini jika model Anda memiliki throughput yang stabil dan kasus penggunaan Anda dapat mentolerir waktu henti 10-20 menit. Contohnya adalah jika Anda ingin melakukan batch panggilan ke model Anda menggunakan jadwal mingguan.

Unit inferensi skala otomatis

Jika model Anda harus mengakomodasi lonjakan permintaan, Amazon Lookout for Vision dapat secara otomatis menskalakan jumlah unit inferensi yang digunakan model Anda. Seiring meningkatnya permintaan, Amazon Lookout for Vision menambahkan unit inferensi tambahan ke model dan menghapusnya saat permintaan menurun.

Untuk membiarkan Lookout for Vision secara otomatis menskalakan unit inferensi untuk model, mulai model dan atur jumlah maksimum unit inferensi yang dapat digunakan dengan menggunakan parameter. `MaxInferenceUnits` Menetapkan jumlah maksimum unit inferensi memungkinkan Anda mengelola biaya menjalankan model dengan membatasi jumlah unit inferensi yang tersedia

untuknya. Jika Anda tidak menentukan jumlah maksimum unit, Lookout for Vision tidak akan secara otomatis menskalakan model Anda, hanya menggunakan jumlah unit inferensi yang Anda mulai. Untuk informasi mengenai jumlah maksimum unit inferensi, lihat Service Quotas.

Anda juga dapat menentukan jumlah minimum unit inferensi dengan menggunakan `MinInferenceUnits` parameter. Ini memungkinkan Anda menentukan throughput minimum untuk model Anda, di mana satu unit inferensi mewakili 1 jam waktu pemrosesan.

> **Note**
>
> Anda tidak dapat mengatur jumlah maksimum unit inferensi dengan konsol Lookout for Vision. Sebagai gantinya, tentukan parameter `MaxInferenceUnits` input ke `StartModel` operasi.

Lookout for Vision menyediakan metrik CloudWatch Amazon Logs berikut yang dapat Anda gunakan untuk menentukan status penskalaan otomatis saat ini untuk model.

Metrik	Deskripsi
`DesiredInferenceUnits`	Jumlah unit inferensi di mana Lookout for Vision ditingkatkan atau turun.
`InServiceInferenceUnits`	Jumlah unit inferensi yang digunakan model.

Jika `DesiredInferenceUnits` =`InServiceInferenceUnits`, Lookout for Vision saat ini tidak menskalakan jumlah unit inferensi.

Jika `DesiredInferenceUnits` >`InServiceInferenceUnits`, Lookout for Vision meningkatkan nilai. `DesiredInferenceUnits`

Jika `DesiredInferenceUnits` <`InServiceInferenceUnits`, Lookout for Vision menurunkan nilai. `DesiredInferenceUnits`

Untuk informasi selengkapnya mengenai metrik yang ditampilkan oleh Lookout for Vision dan dimensi pemfilteran, lihat Memantau Lookout for Vision dengan Amazon. CloudWatch

Untuk mengetahui jumlah maksimum unit inferensi yang Anda minta untuk model, panggil DescribeModeldan periksa `MaxInferenceUnits` bidang dalam respons.

Availability Zone

Amazon Lookout for Vision; mendistribusikan unit inferensi di beberapa Availability Zone dalam AWS suatu Wilayah untuk meningkatkan ketersediaan. Untuk informasi selengkapnya, lihat Availability Zone. Untuk membantu melindungi model produksi Anda dari pemadaman Availability Zone dan kegagalan unit inferensi, mulailah model produksi Anda dengan setidaknya dua unit inferensi.

Jika terjadi pemadaman Availability Zone, semua unit inferensi di Availability Zone tidak tersedia dan kapasitas model berkurang. Panggilan ke DetectAnomaliesdidistribusikan kembali di seluruh unit inferensi yang tersisa. Panggilan tersebut berhasil jika tidak melebihi Transaksi Per Detik (TPS) yang didukung dari unit inferensi yang tersisa. Setelah AWS memperbaiki Availability Zone, unit inferensi dimulai ulang, dan kapasitas penuh dipulihkan.

Jika unit inferensi tunggal gagal, Amazon Lookout for Vision secara otomatis memulai unit inferensi baru di Availability Zone yang sama. Kapasitas model dikurangi sampai unit inferensi baru dimulai.

Memulai model Amazon Lookout for Vision Anda

Sebelum Anda dapat menggunakan model Amazon Lookout for Vision untuk mendeteksi anomali, Anda harus memulai model terlebih dahulu. Anda memulai model dengan memanggil StartModelAPI dan meneruskan yang berikut:

- ProjectName— Nama proyek yang berisi model yang ingin Anda mulai.
- ModelVersion— Versi model yang ingin Anda mulai.
- MinInferenceUnits— Jumlah minimum unit inferensi. Untuk informasi selengkapnya, lihat Unit inferensi.
- (Opsional) MaxInferenceUnits— Jumlah maksimum unit inferensi yang dapat digunakan Amazon Lookout for Vision untuk menskalakan model secara otomatis. Untuk informasi selengkapnya, lihat Unit inferensi skala otomatis.

Konsol Amazon Lookout for Vision menyediakan contoh kode yang dapat Anda gunakan untuk memulai dan menghentikan model.

> **Note**
>
> Anda dikenakan biaya untuk jumlah waktu model Anda berjalan. Untuk menghentikan model yang sedang berjalan, lihatMenghentikan model Amazon Lookout for Vision Anda.

Anda dapat menggunakan AWS SDK untuk melihat model yang sedang berjalan di semua AWS Wilayah di mana Lookout for Vision tersedia. Misalnya kode, lihat find_running_models.py.

Topik

Memulai model Anda (konsol)

Konsol Amazon Lookout for Vision menyediakan perintah AWS CLI yang dapat Anda gunakan untuk memulai model. Setelah model dimulai, Anda dapat mulai mendeteksi anomali pada gambar. Untuk informasi selengkapnya, lihat Mendeteksi anomali dalam gambar.

Untuk memulai model (konsol)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.
5. Pada halaman Sumber daya proyek, pilih proyek yang berisi model terlatih yang ingin Anda mulai.
6. Di bagian Model, pilih model yang ingin Anda mulai.
7. Pada halaman detail model, pilih Gunakan model lalu pilih Integrasikan API ke cloud.

   Tip

   Jika Anda ingin menerapkan model Anda ke perangkat tepi, pilih Buat pekerjaan pengemasan model. Untuk informasi selengkapnya, lihat Mengemas model Amazon Lookout for Vision Anda.

8. Di bawah perintah AWS CLI, salin perintah AWS CLI yang memanggil. `start-model`

9. Pada prompt perintah, masukkan `start-model` perintah yang Anda salin pada langkah sebelumnya. Jika Anda menggunakan `lookoutvision` profil untuk mendapatkan kredensialnya, tambahkan parameternya. `--profile lookoutvision-access`
10. Di konsol, pilih Model di halaman navigasi kiri.
11. Periksa kolom Status untuk status model saat ini, Saat status di-host, Anda dapat menggunakan model untuk mendeteksi anomali dalam gambar. Untuk informasi selengkapnya, lihat Mendeteksi anomali dalam gambar.

Memulai model Amazon Lookout for Vision (SDK)

Anda memulai model dengan memanggil StartModeloperasi.

Sebuah model mungkin membutuhkan waktu beberapa saat untuk memulai. Anda dapat memeriksa status saat ini dengan menelepon DescribeModel. Untuk informasi selengkapnya, lihat Melihat model Anda.

Untuk memulai model Anda (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode contoh berikut untuk memulai model.

   CLI

   Ubah nilai berikut:

   - `project-name`dengan nama proyek yang berisi model yang ingin Anda mulai.
   - `model-version`ke versi model yang ingin Anda mulai.
   - `--min-inference-units`untuk jumlah unit inferensi yang ingin Anda gunakan.
   - (Opsional) `--max-inference-units` hingga jumlah maksimum unit inferensi yang dapat digunakan Amazon Lookout for Vision untuk menskalakan model secara otomatis.

```
aws lookoutvision start-model --project-name "project name"\
  --model-version model version\
  --min-inference-units minimum number of units\
  --max-inference-units max number of units \
  --profile lookoutvision-access
```

Python

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
   @staticmethod
   def start_model(
           lookoutvision_client, project_name, model_version,
min_inference_units, max_inference_units = None):
       """
       Starts the hosting of a Lookout for Vision model.

       :param lookoutvision_client: A Boto3 Lookout for Vision client.
       :param project_name:  The name of the project that contains the version
of the
                             model that you want to start hosting.
       :param model_version: The version of 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: (Optional) The maximum number of inference
units that
       Lookout for Vision can use to automatically scale the model.
       """
       try:
           logger.info(
               "Starting model version %s for project %s", model_version,
project_name)

           if max_inference_units is None:
               lookoutvision_client.start_model(
                   ProjectName = project_name,
                   ModelVersion = model_version,
                   MinInferenceUnits = min_inference_units)

           else:
               lookoutvision_client.start_model(
                   ProjectName = project_name,
                   ModelVersion = model_version,
                   MinInferenceUnits = min_inference_units,
                   MaxInferenceUnits = max_inference_units)

           print("Starting hosting...")
```

```
            status = ""
            finished = False

            # Wait until hosted or failed.
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name, ModelVersion=model_version)
                status = model_description["ModelDescription"]["Status"]

                if status == "STARTING_HOSTING":
                    logger.info("Host starting in progress...")
                    time.sleep(10)
                    continue

                if status == "HOSTED":
                    logger.info("Model is hosted and ready for use.")
                    finished = True
                    continue

                logger.info("Model hosting failed and the model can't be used.")
                finished = True

            if status != "HOSTED":
                logger.error("Error hosting model: %s", status)
                raise Exception(f"Error hosting model: {status}")
        except ClientError:
            logger.exception("Couldn't host model.")
            raise
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
/**
 * Starts hosting an Amazon Lookout for Vision model. Returns when the model has
 * started or if hosting fails. You are charged for the amount of time that a
 * model is hosted. To stop hosting a model, use the StopModel operation.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the model that you
 *                    want to host.
```

```
     * @modelVersion The version of the model that you want to host.
     * @minInferenceUnits The number of inference units to use for hosting.
     * @maxInferenceUnits The maximum number of inference units that Lookout for
     *                    Vision can use for automatically scaling the model. If the
     *                    value is null, automatic scaling doesn't happen.
     * @return ModelDescription The description of the model, which includes the
     *         model hosting status.
     */
    public static ModelDescription startModel(LookoutVisionClient lfvClient, String
     projectName, String modelVersion,
            Integer minInferenceUnits, Integer maxInferenceUnits) throws
     LookoutVisionException, InterruptedException {

        logger.log(Level.INFO, "Starting Model version {0} for project {1}.",
                new Object[] { modelVersion, projectName });

        StartModelRequest startModelRequest = null;

        if (maxInferenceUnits == null) {

            startModelRequest =
     StartModelRequest.builder().projectName(projectName).modelVersion(modelVersion)
                    .minInferenceUnits(minInferenceUnits).build();
        } else {
            startModelRequest =
     StartModelRequest.builder().projectName(projectName).modelVersion(modelVersion)

     .minInferenceUnits(minInferenceUnits).maxInferenceUnits(maxInferenceUnits).build();
        }

        // Start hosting the model.
        lfvClient.startModel(startModelRequest);

        DescribeModelRequest describeModelRequest =
     DescribeModelRequest.builder().projectName(projectName)
                .modelVersion(modelVersion).build();

        ModelDescription modelDescription = null;

        boolean finished = false;
        // Wait until model is hosted or failure occurs.
        do {
```

```
       modelDescription =
lfvClient.describeModel(describeModelRequest).modelDescription();

       switch (modelDescription.status()) {

       case HOSTED:
           logger.log(Level.INFO, "Model version {0} for project {1} is
running.",
                   new Object[] { modelVersion, projectName });
           finished = true;
           break;

       case STARTING_HOSTING:
           logger.log(Level.INFO, "Model version {0} for project {1} is
starting.",
                   new Object[] { modelVersion, projectName });

           TimeUnit.SECONDS.sleep(60);

           break;
       case HOSTING_FAILED:
           logger.log(Level.SEVERE, "Hosting failed for model version {0} for
project {1}.",
                   new Object[] { modelVersion, projectName });
           finished = true;
           break;

       default:
           logger.log(Level.SEVERE, "Unexpected error when hosting model
version {0} for project {1}: {2}.",
                   new Object[] { projectName, modelVersion,
modelDescription.status() });
           finished = true;
           break;

       }

   } while (!finished);

   logger.log(Level.INFO, "Finished starting model version {0} for project {1}
status: {2}",
           new Object[] { modelVersion, projectName,
modelDescription.statusMessage() });
```

```
    return modelDescription;

}
```

3. Jika output kodenya`Model is hosted and ready for use`, Anda dapat menggunakan model untuk mendeteksi anomali pada gambar. Untuk informasi selengkapnya, lihat Mendeteksi anomali dalam gambar.

Menghentikan model Amazon Lookout for Vision Anda

Untuk menghentikan model yang sedang berjalan, Anda memanggil `StopModel` operasi dan meneruskan yang berikut ini:

- Proyek — Nama proyek yang berisi model yang ingin Anda hentikan.
- ModelVersion— Versi model yang ingin Anda hentikan.

Konsol Amazon Lookout for Vision menyediakan contoh kode yang dapat Anda gunakan untuk menghentikan model.

> **Note**
>
> Anda dikenakan biaya untuk jumlah waktu model Anda berjalan.

Topik

- Menghentikan model Anda (konsol)
- Menghentikan model Lookout for Vision (SDK) Amazon Anda

Menghentikan model Anda (konsol)

Lakukan langkah-langkah dalam prosedur berikut untuk menghentikan model Anda menggunakan konsol.

Untuk menghentikan model Anda (konsol)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.

2. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.
5. Pada halaman Sumber daya proyek, pilih proyek yang berisi model yang sedang berjalan yang ingin Anda hentikan.
6. Di bagian Model, pilih model yang ingin Anda hentikan.
7. Pada halaman detail model, pilih Gunakan model lalu pilih Integrasikan API ke cloud.
8. Di bawah perintah AWS CLI, salin perintah AWS CLI yang memanggil. `stop-model`
9. Pada prompt perintah, masukkan `stop-model` perintah yang Anda salin pada langkah sebelumnya. Jika Anda menggunakan `lookoutvision` profil untuk mendapatkan kredensialnya, tambahkan parameternya. `--profile lookoutvision-access`
10. Di konsol, pilih Model di halaman navigasi kiri.
11. Periksa kolom Status untuk status model saat ini. Model telah berhenti ketika nilai kolom Status adalah Pelatihan selesai.

Menghentikan model Lookout for Vision (SDK) Amazon Anda

Anda menghentikan model dengan memanggil StopModeloperasi.

Seorang model mungkin membutuhkan waktu beberapa saat untuk berhenti. Untuk memeriksa status saat ini, gunakan`DescribeModel`.

Untuk menghentikan model Anda (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode contoh berikut untuk menghentikan model yang sedang berjalan.

   CLI

   Ubah nilai berikut:

   - `project-name`dengan nama proyek yang berisi model yang ingin Anda hentikan.
   - `model-version`ke versi model yang ingin Anda hentikan.

```
aws lookoutvision stop-model --project-name "project name"\
```

```
    --model-version model version \
  --profile lookoutvision-access
```

Python

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
    @staticmethod
    def stop_model(lookoutvision_client, project_name, model_version):
        """
        Stops a running Lookout for Vision Model.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the version
of
                             the model that you want to stop hosting.
        :param model_version:  The version of the model that you want to stop
hosting.
        """
        try:
            logger.info("Stopping model version %s for %s", model_version,
project_name)
            response = lookoutvision_client.stop_model(
                ProjectName=project_name, ModelVersion=model_version
            )
            logger.info("Stopping hosting...")

            status = response["Status"]
            finished = False

            # Wait until stopped or failed.
            while finished is False:
                model_description = lookoutvision_client.describe_model(
                    ProjectName=project_name, ModelVersion=model_version
                )
                status = model_description["ModelDescription"]["Status"]

                if status == "STOPPING_HOSTING":
                    logger.info("Host stopping in progress...")
                    time.sleep(10)
                    continue
```

```
                if status == "TRAINED":
                    logger.info("Model is no longer hosted.")
                    finished = True
                    continue

                logger.info("Failed to stop model: %s ", status)
                finished = True

            if status != "TRAINED":
                logger.error("Error stopping model: %s", status)
                raise Exception(f"Error stopping model: {status}")
        except ClientError:
            logger.exception("Couldn't stop hosting model.")
            raise
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
/**
 * Stops the hosting an Amazon Lookout for Vision model. Returns when model has
 * stopped or if hosting fails.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the model that you
 *                    want to stop hosting.
 * @modelVersion The version of the model that you want to stop hosting.
 * @return ModelDescription The description of the model, which includes the
 *         model hosting status.
 */

public static ModelDescription stopModel(LookoutVisionClient lfvClient, String
 projectName,
                String modelVersion) throws LookoutVisionException,
 InterruptedException {

        logger.log(Level.INFO, "Stopping Model version {0} for project {1}.",
                        new Object[] { modelVersion, projectName });

        StopModelRequest stopModelRequest = StopModelRequest.builder()
                        .projectName(projectName)
```

```
                        .modelVersion(modelVersion)
                        .build();

        // Stop hosting the model.

        lfvClient.stopModel(stopModelRequest);

        DescribeModelRequest describeModelRequest =
DescribeModelRequest.builder()
                        .projectName(projectName)
                        .modelVersion(modelVersion)
                        .build();

        ModelDescription modelDescription = null;

        boolean finished = false;
        // Wait until model is stopped or failure occurs.
        do {

                modelDescription =
lfvClient.describeModel(describeModelRequest).modelDescription();

                switch (modelDescription.status()) {

                        case TRAINED:
                                logger.log(Level.INFO, "Model version {0} for
project {1} has stopped.",
                                                new Object[] { modelVersion,
projectName });
                                finished = true;
                                break;

                        case STOPPING_HOSTING:
                                logger.log(Level.INFO, "Model version {0} for
project {1} is stopping.",
                                                new Object[] { modelVersion,
projectName });

                                TimeUnit.SECONDS.sleep(60);

                                break;

                        default:
                                logger.log(Level.SEVERE,
```

```
                                                    "Unexpected error when stopping
 model version {0} for project {1}: {2}.",
                                                    new Object[] { projectName,
 modelVersion,

 modelDescription.status() });
                                finished = true;
                                break;

                }

        } while (!finished);

        logger.log(Level.INFO, "Finished stopping model version {0} for project
 {1} status: {2}",
                        new Object[] { modelVersion, projectName,
 modelDescription.statusMessage() });

        return modelDescription;

}
```

Mendeteksi anomali dalam gambar

Untuk mendeteksi anomali dalam gambar dengan model Amazon Lookout for Vision yang terlatih, Anda memanggilDetectAnomaliesoperasi. Hasil dari`DetectAnomalies`termasuk prediksi Boolean yang mengklasifikasikan gambar sebagai mengandung satu atau lebih anomali dan nilai kepercayaan untuk prediksi. Jika modelnya adalah model segmentasi gambar, hasilnya juga mencakup topeng berwarna yang menunjukkan posisi berbagai jenis anomali.

Gambar yang Anda berikan`DetectAnomalies`harus memiliki dimensi lebar dan tinggi yang sama dengan gambar yang Anda gunakan untuk melatih model.

`DetectAnomalies`menerima gambar sebagai gambar format PNG atau JPG. Kami menyarankan agar gambar berada dalam format pengkodean dan kompresi yang sama seperti yang digunakan untuk melatih model. Misalnya, jika Anda melatih model dengan gambar format PNG, hubungi`DetectAnomalies`dengan gambar format PNG.

Sebelum meneleponDetectAnomalies, Anda harus terlebih dahulu memulai model Anda dengan`StartModel`operasi. Untuk informasi selengkapnya, lihat Memulai model Amazon Lookout for Vision Anda. Anda dikenai biaya untuk jumlah waktu, dalam hitungan menit, bahwa model berjalan dan untuk jumlah unit deteksi anomali yang digunakan model Anda. Jika Anda tidak menggunakan model, gunakan`StopModel`operasi untuk menghentikan model Anda. Untuk informasi selengkapnya, lihat Menghentikan model Amazon Lookout for Vision Anda.

Topik

- MeneleponDetectAnomalies
- Memahami respons dariDetectAnomalies
- Menentukan apakah gambar itu anomali
- Menampilkan informasi klasifikasi dan segmentasi
- Menemukan anomali denganAWS Lambdafungsi

MeneleponDetectAnomalies

Untuk meneleponDetectAnomalies, tentukan yang berikut:

- Proyek- Nama proyek yang berisi model yang ingin Anda gunakan.
- ModelVersion- Versi model yang ingin Anda gunakan.

- ContentType- Jenis gambar yang ingin Anda analisis. Nilai yang valid adalah`image/png`(Gambar format PNG) dan`image/jpeg`(Gambar format JPG).
- Tubuh- Byte biner yang tidak dikodekan yang mewakili gambar.

  Gambar harus memiliki dimensi yang sama dengan gambar yang digunakan untuk melatih model.

Contoh berikut menunjukkan bagaimana untuk memanggil`DetectAnomalies`. Anda dapat menggunakan respon fungsi dari contoh Python dan Java untuk memanggil fungsiMenentukan apakah gambar itu anomali.

AWS CLI

Perintah AWS CLI ini menampilkan output JSON untuk operasi CLI `DetectAnomalies`. Ubah nilai parameter input berikut:

- `project name`dengan nama proyek yang ingin Anda gunakan.
- `model version`dengan versi model yang ingin Anda gunakan.
- `content type`dengan jenis gambar yang ingin Anda gunakan. Nilai yang valid adalah`image/png`(Gambar format PNG) dan`image/jpeg`(Gambar format JPG).
- `file name`dengan path dan nama file dari gambar yang ingin Anda gunakan. Pastikan bahwa jenis file cocok dengan nilai`content-type`.

```
aws lookoutvision detect-anomalies  --project-name project name\
    --model-version model version\
    --content-type content type\
    --body file name \
    --profile lookoutvision-access
```

Python

Untuk contoh kode lengkap, lihatGitHub.

```
    def detect_anomalies(lookoutvision_client, project_name, model_version, photo):
        """
        Calls DetectAnomalies using the supplied project, model version, and image.
        :param lookoutvision_client: A Lookout for Vision Boto3 client.
        :param project: The project that contains the model that you want to use.
        :param model_version: The version of the model that you want to use.
```

```
        :param photo: The photo that you want to analyze.
        :return: The DetectAnomalyResult object that contains the analysis results.
        """

        image_type = imghdr.what(photo)
        if image_type == "jpeg":
            content_type = "image/jpeg"
        elif image_type == "png":
            content_type = "image/png"
        else:
            logger.info("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
        with open(photo, "rb") as image:
            response = lookoutvision_client.detect_anomalies(
                ProjectName=project_name,
                ContentType=content_type,
                Body=image.read(),
                ModelVersion=model_version)

        return response['DetectAnomalyResult']
```

Java V2

```
  public static DetectAnomalyResult detectAnomalies(LookoutVisionClient lfvClient,
String projectName,
            String modelVersion,
            String photo) throws IOException, LookoutVisionException {
        /**
         * Creates an Amazon Lookout for Vision dataset from a manifest file.
         * Returns after Lookout for Vision creates the dataset.
         *
         * @param lfvClient    An Amazon Lookout for Vision client.
         * @param projectName  The name of the project in which you want to create a
         *                     dataset.
         * @param modelVersion The version of the model that you want to use.
         *
         * @param photo        The photo that you want to analyze.
         *
         * @return DetectAnomalyResult The analysis result from DetectAnomalies.
```

```
         */

        logger.log(Level.INFO, "Processing local file: {0}", photo);

        // Get image bytes.

        InputStream sourceStream = new FileInputStream(new File(photo));
        SdkBytes imageSDKBytes = SdkBytes.fromInputStream(sourceStream);
        byte[] imageBytes = imageSDKBytes.asByteArray();

        // Get the image type. Can be image/jpeg or image/png.
        String contentType = getImageType(imageBytes);

        // Detect anomalies in the supplied image.
        DetectAnomaliesRequest request =
DetectAnomaliesRequest.builder().projectName(projectName)
                .modelVersion(modelVersion).contentType(contentType).build();

        DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
                RequestBody.fromBytes(imageBytes));

        /*
         * Tip: You can also use the following to analyze a local file.
         * Path path = Paths.get(photo);
         * DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
path);
         */
        DetectAnomalyResult result = response.detectAnomalyResult();

        String prediction = "Prediction: Normal";

        if (Boolean.TRUE.equals(result.isAnomalous())) {
            prediction = "Prediction: Anomalous";
        }

        // Convert confidence to percentage.
        NumberFormat defaultFormat = NumberFormat.getPercentInstance();
        defaultFormat.setMinimumFractionDigits(1);
        String confidence = String.format("Confidence: %s",
defaultFormat.format(result.confidence()));

        // Log classification result.
        String photoPath = "File: " + photo;
        String[] imageLines = { photoPath, prediction, confidence };
```

```
        logger.log(Level.INFO, "Image: {0}\nAnomalous: {1}\nConfidence {2}",
imageLines);

        return result;

    }

    // Gets the image mime type. Supported formats are image/jpeg and image/png.
    private static String getImageType(byte[] image) throws IOException {

        InputStream is = new BufferedInputStream(new ByteArrayInputStream(image));
        String mimeType = URLConnection.guessContentTypeFromStream(is);

        logger.log(Level.INFO, "Image type: {0}", mimeType);

        if (mimeType.equals("image/jpeg") || mimeType.equals("image/png")) {
            return mimeType;
        }
        // Not a supported file type.
        logger.log(Level.SEVERE, "Unsupported image type: {0}", mimeType);
        throw new IOException(String.format("Wrong image type. %s format isn't
supported.", mimeType));
    }
```

Memahami respons dariDetectAnomalies

Tanggapan dari`DetectAnomalies`bervariasi tergantung pada jenis model yang Anda latih (model klasifikasi atau model segmentasi). Dalam kedua kasus, respon adalahDetectAnomalyResultobjek.

Model klasifikasi

Jika model Anda adalahModel klasifikasi gambar, respon dari`DetectAnomalies`berisi berikut ini:

- IsAnomalous- Indikator Boolean bahwa gambar berisi satu atau lebih anomali.
- Keyakinan- Keyakinan yang dimiliki Amazon Lookout for Vision dalam keakuratan prediksi anomali (`IsAnomalous`).`Confidence`adalah nilai floating point antara 0 dan 1. Nilai yang lebih tinggi menunjukkan kepercayaan yang lebih tinggi.
- Sumber— Informasi tentang gambar diteruskan ke`DetectAnomalies`.

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.9996867775917053
    }
}
```

Anda menentukan apakah dalam gambar anomali dengan memeriksa`IsAnomalous`lapangan dan mengkonfirmasikan bahwa`Confidence`nilai cukup tinggi untuk kebutuhan Anda.

Jika Anda menemukan nilai kepercayaan yang dikembalikan oleh`DetectAnomalies`terlalu rendah, pertimbangkan untuk melatih ulang modelnya. Untuk kode sampel, lihat Klasifikasi.

Model segmentasi

Jika model Anda adalahModel segmentasi gambar, responsnya mencakup informasi klasifikasi dan informasi segmentasi, seperti topeng gambar dan jenis anomali. Informasi klasifikasi dihitung secara terpisah dari informasi segmentasi dan Anda tidak boleh mengasumsikan hubungan di antara mereka. Jika Anda tidak mendapatkan informasi segmentasi dalam respons, periksa apakah Anda memiliki versi terbaruAWSSDK diinstal (AWS Command Line Interface, jika Anda menggunakanAWS CLI). Misalnya kode, lihatSegmentasidanMenampilkan informasi klasifikasi dan segmentasi.

- IsAnomalous(klasifikasi) - Indikator Boolean yang mengklasifikasikan gambar sebagai normal atau anomali.
- Keyakinan(klasifikasi) - Keyakinan yang dimiliki Amazon Lookout for Vision dalam keakuratan klasifikasi gambar (`IsAnomalous`).`Confidence`adalah nilai floating point antara 0 dan 1. Nilai yang lebih tinggi menunjukkan kepercayaan yang lebih tinggi.
- Sumber— Informasi tentang gambar diteruskan ke`DetectAnomalies`.
- AnomalyMask(segmentasi) - Masker piksel yang menutupi anomali yang ditemukan pada gambar yang dianalisis. Mungkin ada beberapa anomali pada gambar. Warna peta topeng menunjukkan jenis anomali. Warna topeng dipetakan ke warna yang ditetapkan ke jenis anomali dalam kumpulan data pelatihan. Untuk menemukan jenis anomali dari warna topeng, periksa`Color`di dalam`PixelAnomaly`bidang setiap anomali kembali di`Anomalies`daftar. Untuk kode sampel, lihat Menampilkan informasi klasifikasi dan segmentasi.

- Anomali(segmentasi) - Daftar anomali yang ditemukan dalam gambar. Setiap anomali mencakup tipe anomali (`Name`), dan informasi piksel (`PixelAnomaly`).`TotalPercentageArea`adalah area persentase gambar yang menutupi anomali.`Color`adalah warna topeng untuk anomali.

  Elemen pertama dalam daftar selalu merupakan tipe anomali yang mewakili latar belakang gambar (`BACKGROUND`) dan tidak boleh dianggap sebagai anomali. Amazon Lookout for Vision secara otomatis menambahkan jenis anomali latar belakang ke respons. Anda tidak perlu mendeklarasikan jenis anomali latar belakang dalam kumpulan data Anda.


```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.9996814727783203,
        "Anomalies": [
            {
                "Name": "background",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.998999834060669,
                    "Color": "#FFFFFF"
                }
            },
            {
                "Name": "scratch",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.0004034999874420464,
                    "Color": "#7ED321"
                }
            },
            {
                "Name": "dent",
                "PixelAnomaly": {
                    "TotalPercentageArea": 0.0005966666503809392,
                    "Color": "#4DD8FF"
                }
            }
        ],
        "AnomalyMask": "iVBORw0....."
    }
```

```
}
```

Menentukan apakah gambar itu anomali

Anda dapat menentukan apakah suatu gambar anomali dalam berbagai cara. Metode yang Anda pilih tergantung pada kasus penggunaan dan jenis model Anda. Berikut ini adalah solusi potensional.

Topik

- Klasifikasi
- Segmentasi

Klasifikasi

`IsAnomalous`mengklasifikasikan gambar sebagai anomali, gunakan`Confidence`lapangan untuk membantu memutuskan apakah gambar sebenarnya anomali. Nilai yang lebih tinggi menunjukkan kepercayaan diri yang lebih besar. Misalnya, Anda mungkin memutuskan suatu produk rusak hanya jika kepercayaan lebih dari 80%. Anda dapat mengklasifikasikan gambar yang dianalisis berdasarkan model klasifikasi atau dengan model segmentasi gambar.

Python

Untuk contoh kode lengkap, lihatGitHub.

```
   def reject_on_classification(image, prediction, confidence_limit):
       """
       Returns True if the anomaly confidence is greater than or equal to
       the supplied confidence limit.
       :param image: The name of the image file that was analyzed.
       :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies
       :param confidence_limit: The minimum acceptable confidence. Float value
between 0 and 1.
       :return: True if the error condition indicates an anomaly, otherwise False.
       """

       reject = False

       logger.info("Checking classification for %s", image)
```

```
        if prediction['IsAnomalous'] and prediction['Confidence'] >=
confidence_limit:
            reject = True
            reject_info=(f"Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) is greater"
                    f" than limit ({confidence_limit:.2%})")
            logger.info("%s", reject_info)

        if not reject:
            logger.info("No anomalies found.")
        return reject
```

Java V2

```
   public static boolean rejectOnClassification(String image, DetectAnomalyResult
prediction, float minConfidence) {
        /**
         * Rejects an image based on its anomaly classification and prediction
         * confidence
         *
         * @param image         The file name of the analyzed image.
         * @param prediction    The prediction for an image analyzed with
         *                      DetectAnomalies.
         * @param minConfidence The minimum acceptable confidence for the prediction
         *                      (0-1).
         *
         * @return boolean True if the image is anomalous, otherwise False.
         */

        Boolean reject = false;

        logger.log(Level.INFO, "Checking classification for {0}", image);

        String[] logParameters = { prediction.confidence().toString(),
String.valueOf(minConfidence) };

        if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
            logger.log(Level.INFO, "Rejected: Anomaly confidence {0} is greater than
confidence limit {1}",
                    logParameters);
            reject = true;
        }
```

```
        if (Boolean.FALSE.equals(reject))
            logger.log(Level.INFO, ": No anomalies found.");

        return reject;

    }
```

Segmentasi

Jika model Anda adalah model segmentasi gambar, Anda dapat menggunakan informasi segmentasi untuk menentukan apakah gambar mengandung anomali. Anda juga dapat menggunakan model segmentasi gambar untuk mengklasifikasikan gambar. Misalnya kode yang mendapat dan menampilkan topeng gambar, lihatMenampilkan informasi klasifikasi dan segmentasi

Area anomali

Gunakan cakupan persentase (`TotalPercentageArea`) anomali pada gambar. Misalnya, Anda mungkin memutuskan produk rusak jika area anomali lebih besar dari 1% dari gambar.

Python

Untuk contoh kode lengkap, lihatGitHub.

```
   def reject_on_coverage(image, prediction, confidence_limit, anomaly_label,
coverage_limit):
        """
        Checks if the coverage area of an anomaly is greater than the coverage limit
and if
        the prediction confidence is greater than the confidence limit.
        :param image: The name of the image file that was analyzed.
        :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies
        :param confidence_limit: The minimum acceptable confidence (float 0-1).
        :anomaly_label: The anomaly label for the type of anomaly that you want to
check.
        :coverage_limit: The maximum acceptable percentage coverage of an anomaly
(float 0-1).
        :return: True if the error condition indicates an anomaly, otherwise False.
        """

        reject = False
```

```
        logger.info("Checking coverage for %s", image)

        if prediction['IsAnomalous'] and prediction['Confidence'] >=
 confidence_limit:
            for anomaly in prediction['Anomalies']:
                if (anomaly['Name'] == anomaly_label and
                        anomaly['PixelAnomaly']['TotalPercentageArea'] >
 (coverage_limit)):
                    reject = True
                    reject_info=(f"Rejected: Anomaly confidence
 ({prediction['Confidence']:.2%}) "
                        f"is greater than limit ({confidence_limit:.2%}) and
 {anomaly['Name']} "
                        f"coverage ({anomaly['PixelAnomaly']
['TotalPercentageArea']:.2%}) "
                        f"is greater than limit ({coverage_limit:.2%})")

                    logger.info("%s", reject_info)

        if not reject:
            logger.info("No anomalies found.")

        return reject
```

Java V2

```
   public static Boolean rejectOnCoverage(String image, DetectAnomalyResult
 prediction, float minConfidence,
            String anomalyType, float maxCoverage) {
        /**
         * Rejects an image based on a maximum allowable coverage area for an
 anomaly
         * type.
         *
         * @param image         The file name of the analyzed image.
         * @param prediction    The prediction for an image analyzed with
         *                      DetectAnomalies.
         * @param minConfidence The minimum acceptable confidence for the prediction
         *                      (0-1).
         * @param anomalyTypes  The anomaly type to check.
         * @param maxCoverage   The maximum allowable coverage area of the anomaly
 type.
```

```
         *                      (0-1).
         *
         * @return boolean True if the coverage area of the anomaly type exceeds the
         *         maximum allowed, otherwise False.
         */

        Boolean reject = false;

        logger.log(Level.INFO, "Checking coverage for {0}", image);

        if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
            for (Anomaly anomaly : prediction.anomalies()) {

                if (Objects.equals(anomaly.name(), anomalyType)
                        && anomaly.pixelAnomaly().totalPercentageArea() >=
maxCoverage) {

                    String[] logParameters = { prediction.confidence().toString(),
                            String.valueOf(minConfidence),

String.valueOf(anomaly.pixelAnomaly().totalPercentageArea()),
                            String.valueOf(maxCoverage) };
                    logger.log(Level.INFO,
                            "Rejected: Anomaly confidence {0} is greater than
confidence limit {1} and " +
                                    "{2} anomaly type coverage is higher than
coverage limit {3}\n",
                            logParameters);
                    reject = true;

                }
            }
        }

        if (Boolean.FALSE.equals(reject))
            logger.log(Level.INFO, ": No anomalies found.");

        return reject;
    }
```

Jumlah jenis anomali

Gunakan hitungan jenis anomali yang berbeda (`Name`) ditemukan pada gambar. Misalnya, Anda mungkin memutuskan suatu produk rusak jika ada lebih dari dua jenis anomali yang ada.

Python

Untuk contoh kode lengkap, lihatGitHub.

```
   def reject_on_anomaly_types(image, prediction, confidence_limit,
anomaly_types_limit):
       """
       Checks if the number of anomaly types is greater than than the anomaly types
       limit and if the prediction confidence is greater than the confidence limit.
       :param image: The name of the image file that was analyzed.
       :param prediction: The DetectAnomalyResult object returned from
DetectAnomalies
       :param confidence: The minimum acceptable confidence. Float value between 0
and 1.
       :param anomaly_types_limit: The maximum number of allowable anomaly types
(Integer).
       :return: True if the error condition indicates an anomaly, otherwise False.
       """

       logger.info("Checking number of anomaly types for %s",image)

       reject = False

       if prediction['IsAnomalous'] and prediction['Confidence'] >=
confidence_limit:

           anomaly_types = {anomaly['Name'] for anomaly in prediction['Anomalies']\
               if anomaly['Name'] != 'background'}

           if len (anomaly_types) > anomaly_types_limit:
                   reject = True
                   reject_info = (f"Rejected: Anomaly confidence
({prediction['Confidence']:.2%}) "
                   f"is greater than limit ({confidence_limit:.2%}) and "
                   f"the number of anomaly types ({len(anomaly_types)-1}) is "
                   f"greater than the limit ({anomaly_types_limit})")

                   logger.info("%s", reject_info)
```

```
        if not reject:
            logger.info("No anomalies found.")
        return reject
```

Java V2

```
   public static Boolean rejectOnAnomalyTypeCount(String image, DetectAnomalyResult
prediction,
           float minConfidence, Integer maxAnomalyTypes) {

       /**
        * Rejects an image based on a maximum allowable number of anomaly types.
        *
        * @param image           The file name of the analyzed image.
        * @param prediction      The prediction for an image analyzed with
        *                        DetectAnomalies.
        * @param minConfidence   The minimum acceptable confidence for the
predictio
        *                        (0-1).
        * @param maxAnomalyTypes The maximum allowable number of anomaly types.
        *
        * @return boolean True if the image contains more than the maximum allowed
        *         anomaly types, otherwise False.
        */

       Boolean reject = false;

       logger.log(Level.INFO, "Checking coverage for {0}", image);

       Set<String> defectTypes = new HashSet<>();

       if (Boolean.TRUE.equals(prediction.isAnomalous()) && prediction.confidence()
>= minConfidence) {
           for (Anomaly anomaly : prediction.anomalies()) {
               defectTypes.add(anomaly.name());
           }
           // Reduce defect types by one to account for 'background' anomaly type.
           if ((defectTypes.size() - 1) > maxAnomalyTypes) {
               String[] logParameters = { prediction.confidence().toString(),
                       String.valueOf(minConfidence),
                       String.valueOf(defectTypes.size()),
                       String.valueOf(maxAnomalyTypes) };
```

```
                logger.log(Level.INFO, "Rejected: Anomaly confidence {0} is >=
minimum confidence {1} and " +
                        "the number of anomaly types {2} > the allowable number of
anomaly types {3}\n", logParameters);
                reject = true;
            }

        }

        if (Boolean.FALSE.equals(reject))
            logger.log(Level.INFO, ": No anomalies found.");

        return reject;
    }
```

Menampilkan informasi klasifikasi dan segmentasi

Contoh ini menunjukkan gambar yang dianalisis dan melapisi hasil analisis. Jika respons termasuk topeng anomali, topeng ditampilkan dalam warna jenis anomali terkait.

Untuk menampilkan informasi klasifikasi gambar dan segmentasi gambar

1. Jika Anda belum melakukannya, lakukan hal berikut:

   a. Jika Anda belum melakukannya, instal dan konfigurasikanAWS CLIdanAWSSDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.

   b. Latih model Anda.

   c. Mulai model Anda.

2. Pastikan pengguna meneleponDetectAnomaliesmemiliki akses ke versi model yang ingin Anda gunakan. Untuk informasi selengkapnya, lihat Siapkan izin SDK.

3. Gunakan kode berikut.

   Python

   Kode contoh berikut mendeteksi anomali dalam gambar yang Anda berikan. Contoh mengambil opsi baris perintah berikut:

   - `project`— nama proyek yang ingin Anda gunakan.
   - `version`- versi model, dalam proyek, yang ingin Anda gunakan.

- `image`- jalur dan file file gambar lokal (format JPEG atau PNG).

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
"""
Shows how to detect and show anomalies in an image using a trained Amazon
 Lookout
for Vision model. The script displays the analysed image and overlays mask and
 analysis
output.
"""

import argparse
import logging
import io
import boto3
from PIL import Image, ImageDraw, ImageFont

from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


class ShowAnomalies:
    """
    Class to detect and show anomalies in an image analyzed by detect_anomalies.
    """



    @staticmethod
    def draw_line(draw, text, fnt,  y_coordinate, color):
        """
        Draws a line of text on the supplied drawing surface.
        :param draw: The surface on which to draw the text.
        :param text: The text to draw in the drawing surface.
        :param fnt: The font for the text.
        :param y_coordinate: The y position for the text.
        :param color: The color for the text.
        :returns The y coordinate for the next line of text.
        """
        text_width, text_height = draw.textsize(text, fnt)
        draw.rectangle([(10, y_coordinate), (text_width + 10,
```

```
                                                y_coordinate + text_height)],
 fill="black")
        draw.text((10, y_coordinate), text, fill=color, font=fnt)

        y_coordinate += text_height

        return y_coordinate

    @staticmethod
    def draw_analysis_text(image, analysis):
        """
        Draws classification and segmentation info onto supplied image
        overlay analysis results on an image analyzed by detect_anomalies.
        :param analysis: The response from a call to detect_anomalies.
        :returns Nothing
        """

        ## Calculate a reasonable font size based on image width.
        font_size = int(image.size[0]/32)

        fnt = ImageFont.truetype('/Library/Fonts/Tahoma.ttf', font_size)

        draw = ImageDraw.Draw(image)

        y_coordinate = 0

        # Draw classification information.
        prediction = "Anomalous" if analysis["DetectAnomalyResult"]
["IsAnomalous"] \
            else "Normal"

        confidence = analysis["DetectAnomalyResult"]["Confidence"]
        found_anomalies = analysis["DetectAnomalyResult"]['Anomalies']
        segmentation_info = False

        logger.info("Prediction: %s", format(prediction))
        logger.info("Confidence: %s", format(confidence))

        y_coordinate = 0
        y_coordinate = ShowAnomalies.draw_line(
            draw, "Classification", fnt, y_coordinate, "white")
        y_coordinate = ShowAnomalies.draw_line(
            draw, f"  Prediction: {prediction}", fnt, y_coordinate, "white")
        y_coordinate = ShowAnomalies.draw_line(
```

```
            draw, f"  Confidence: {confidence:.2%}", fnt, y_coordinate, "white")

        # Draw segmentation information, if present.
        if (len(found_anomalies)) > 1:
            logger.info("Anomalies:")

            y_coordinate = ShowAnomalies.draw_line(
                draw, "Segmentation:", fnt, y_coordinate, "white")
            for i in range(1, len(found_anomalies)):

                # Only display info if more than 0% coverage found.
                percent_coverage = found_anomalies[i]['PixelAnomaly']
['TotalPercentageArea']
                if percent_coverage > 0:
                    segmentation_info = True
                    logger.info("  %s", found_anomalies[i]['Name'])
                    logger.info("    Color: %s",
                                found_anomalies[i]['PixelAnomaly']['Color'])
                    logger.info("    Area: %s", percent_coverage)
                    y_coordinate = ShowAnomalies.draw_line(
                        draw,
                        f"  Anomaly: {found_anomalies[i]['Name']}. Area:
 {percent_coverage:.2%}",
                        fnt,
                        y_coordinate,
                        found_anomalies[i]['PixelAnomaly']['Color'])

            if not segmentation_info:
                y_coordinate = ShowAnomalies.draw_line(
                    draw, "No segmentation information found.", fnt,
 y_coordinate, "white")




    @staticmethod
    def show_anomaly_prediction(lookoutvision_client, project_name,
 model_version, photo):
        """
        Detects anomalies in an image (jpg/png) by using your Amazon Lookout for
 Vision
        model. Displays the image and overlays prediction information text.
        :param lookoutvision_client: An Amazon Lookout for Vision Boto3 client.
```

```
        :param project_name: The name of the project that contains the model
that
        you want to use.
        :param model_version: The version of the model that you want to use.
        :param photo: The path and name of the image in which you want to detect
        anomalies.
        """
        try:

            logger.info("Detecting anomalies in %s", photo)

            image = Image.open(photo)
            image_type = Image.MIME[image.format]

            # Check that image type is valid.
            if image_type not in ("image/jpeg", "image/png"):
                logger.info("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()

            # Analyze the image.
            response = lookoutvision_client.detect_anomalies(
                ProjectName=project_name,
                ContentType=image_type,
                Body=image_bytes,
                ModelVersion=model_version
            )

            # Overlay mask onto analyzed image.
            image_mask_bytes = response["DetectAnomalyResult"]["AnomalyMask"]
            image_mask = Image.open(io.BytesIO(image_mask_bytes))

            final_img = Image.blend(image, image_mask, 0.5) \
                if response["DetectAnomalyResult"]["IsAnomalous"] else image

            # Overlay analysis output on image.
            ShowAnomalies.draw_analysis_text(final_img, response)
```

```
            final_img.show()

        except ClientError as err:
            logger.info(format(err))
            raise


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

    parser.add_argument(
        "project", help="The project containing the model that you want to use."
    )
    parser.add_argument(
        "version", help="The version of the model that you want to use."
    )
    parser.add_argument(
        "image",
        help="The file that you want to analyze. "
        "Supply a local file path.",
    )


def main():
    """
    Entrypoint for anomaly detection example.
    """

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

        session = boto3.Session(
            profile_name='lookoutvision-access')

        lookoutvision_client = session.client("lookoutvision")

        parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)

        add_arguments(parser)
```

```
        args = parser.parse_args()

        # Analyze the image and show results.
        ShowAnomalies.show_anomaly_prediction(
            lookoutvision_client, args.project, args.version, args.image
        )

    except ClientError as err:
        print("A service error occured: " +
              format(err.response["Error"]["Message"]))
    except FileNotFoundError as err:
        print("The supplied file couldn't be found: " + err.filename)
    except ValueError as err:
        print("A value error occured. " + format(err))
    else:
        print("Successfully completed analysis.")


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

Java 2

Kode contoh berikut mendeteksi anomali dalam gambar yang Anda berikan. Contoh mengambil opsi baris perintah berikut:

- `project`— nama proyek yang ingin Anda gunakan.
- `version`- versi model, dalam proyek, yang ingin Anda gunakan.
- `image`- jalur dan file file gambar lokal (format JPEG atau PNG).

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


package com.example.lookoutvision;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.SdkBytes;
```

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.lookoutvision.LookoutVisionClient;
import software.amazon.awssdk.services.lookoutvision.model.Anomaly;
import
 software.amazon.awssdk.services.lookoutvision.model.DetectAnomaliesRequest;
import
 software.amazon.awssdk.services.lookoutvision.model.DetectAnomaliesResponse;
import software.amazon.awssdk.services.lookoutvision.model.DetectAnomalyResult;
import
 software.amazon.awssdk.services.lookoutvision.model.LookoutVisionException;

import java.io.BufferedInputStream;
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.net.URLConnection;

import java.text.NumberFormat;
import java.awt.*;
import java.awt.font.LineMetrics;
import java.awt.image.BufferedImage;
import javax.imageio.ImageIO;
import javax.swing.*;

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

// Finds anomalies on a supplied image.
public class ShowAnomalies extends JPanel {
/**
 * Finds and displays anomalies on a supplied image.
 */

    private static final long serialVersionUID = 1L;
    private transient BufferedImage image;
    private transient BufferedImage maskImage;
    private transient Dimension dimension;
    public static final Logger logger =
 Logger.getLogger(ShowAnomalies.class.getName());

    // Constructor. Finds anomalies in a local image file.
```

```
   public ShowAnomalies(LookoutVisionClient lfvClient, String projectName,
String modelVersion,
           String photo) throws IOException, LookoutVisionException {

       logger.log(Level.INFO, "Processing local file: {0}", photo);

       maskImage = null;

       // Get image bytes and buffered image.
       InputStream sourceStream = new FileInputStream(new File(photo));
       SdkBytes imageSDKBytes = SdkBytes.fromInputStream(sourceStream);
       byte[] imageBytes = imageSDKBytes.asByteArray();
       ByteArrayInputStream inputStream = new
ByteArrayInputStream(imageSDKBytes.asByteArray());
       image = ImageIO.read(inputStream);

       // Get the image type. Can be image/jpeg or image/png.
       String contentType = getImageType(imageBytes);

       // Set the size of the window that shows the image.
       setWindowDimensions();

       // Detect anomalies in the supplied image.
       DetectAnomaliesRequest request =
DetectAnomaliesRequest.builder().projectName(projectName)
               .modelVersion(modelVersion).contentType(contentType).build();

       DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
               RequestBody.fromBytes(imageBytes));

       /*
        * Tip: You can also use the following to analyze a local file.
        * Path path = Paths.get(photo);
        * DetectAnomaliesResponse response = lfvClient.detectAnomalies(request,
path);
        */
       DetectAnomalyResult result = response.detectAnomalyResult();


       if (result.anomalyMask() != null){
           SdkBytes maskSDKBytes = result.anomalyMask();

           ByteArrayInputStream maskInputStream = new
ByteArrayInputStream(maskSDKBytes.asByteArray());
```

```
            maskImage = ImageIO.read(maskInputStream);
        }

        drawImageInfo(result);

    }

    // 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;
        dimension.height = (int) dimension.getHeight() / 2;

        if (image.getWidth() < dimension.width || image.getHeight() <
dimension.height) {
            dimension.width = image.getWidth();
            dimension.height = image.getHeight();
        }
        setPreferredSize(dimension);

    }

    private int drawLine(Graphics2D g2d, String line, FontMetrics metrics, int
yPos, Color color) {
    /**
    * Draws a line of text at the spsecified y position and color.
    * confidence
    *
    * @param g2D The Graphics2D object for the image.
    * @param line The line of text to draw.
    * @param metrics The font information to use.
    * @param yPos The y position for the line of text.
    *
    * @return  The yPos for the next line of text.
    */

        int indent = 10;

        // Get text height, width, and descent.
        int textWidth = metrics.stringWidth(line);
        LineMetrics lm = metrics.getLineMetrics(line, g2d);
        int textHeight = (int) lm.getHeight();
        int descent = (int) lm.getDescent();
```

```
        int y2Pos = (yPos + textHeight) - descent;

        // Draw black rectangle.
        g2d.setColor(Color.BLACK);
        g2d.fillRect(indent, yPos, textWidth, textHeight);

        // Draw text.
        g2d.setColor(color);
        g2d.drawString(line, indent, y2Pos);

        yPos += textHeight;

        return yPos;

    }

    public void drawImageInfo(DetectAnomalyResult result) {
    /**
     * Draws the results from DetectAnomalies onto the output image.
     *
     * @param result The response from a call to
     *               DetectAnomalies.
     *
     */

        // Set up drawing.
        Graphics2D g2d = image.createGraphics();

        if (result.anomalyMask() != null){
            Composite composite = g2d.getComposite();
            g2d.setComposite(AlphaComposite.SrcOver.derive(0.5f));
            int x = (image.getWidth() - maskImage.getWidth()) / 2;
            int y = (image.getHeight() - maskImage.getHeight()) / 2;
            g2d.drawImage(maskImage, x, y, null);
            // Set composite for overlaying text.
            g2d.setComposite(composite);
        }

        //Calculate font size based on arbitary 32 pixel image width.
        int fontSize = (image.getWidth() / 32);


        g2d.setFont(new Font("Tahoma", Font.PLAIN, fontSize));
```

```
        Font font = g2d.getFont();
        FontMetrics metrics = g2d.getFontMetrics(font);

        // Get classification information.

        String prediction = "Prediction: Normal";

        if (Boolean.TRUE.equals(result.isAnomalous())) {
            prediction = "Prediction: Anomalous";
        }

        // Convert prediction to percentage.
        NumberFormat defaultFormat = NumberFormat.getPercentInstance();
        defaultFormat.setMinimumFractionDigits(1);
        String confidence = String.format("Confidence: %s",
defaultFormat.format(result.confidence()));

        // Draw classification information.
        int yPos = 0;

        yPos = drawLine(g2d, "Classification:", metrics, yPos, Color.WHITE);
        yPos = drawLine(g2d, prediction, metrics, yPos, Color.WHITE);
        yPos = drawLine(g2d, confidence, metrics, yPos, Color.WHITE);

        // Draw segmentation info.
        yPos = drawLine(g2d, "Segmentation:", metrics, yPos, Color.WHITE);

        // Ignore background label, so size must be > 1
        if (result.anomalies().size() > 1) {
            for (Anomaly anomaly : result.anomalies()) {
                if (anomaly.name().equals("background"))
                    continue;
                String label = String.format("Anomaly: %s. Area: %s",
anomaly.name(),

defaultFormat.format(anomaly.pixelAnomaly().totalPercentageArea()));
                Color anomalyColor =
Color.decode((anomaly.pixelAnomaly().color()));
                yPos = drawLine(g2d, label, metrics, yPos, anomalyColor);

            }

        } else {
            drawLine(g2d, "None found.", metrics, yPos, Color.WHITE);
```

```
        }

        g2d.dispose();

    }

    @Override
    public void paintComponent(Graphics g)
    /**
     * Draws the image and analysis results.
     *
     * @param g The Graphics context object for drawing.
     *          DetectAnomalies.
     *
     */
    {

        Graphics2D g2d = (Graphics2D) g; // Create a Java2D version of g.

        // Draw the image.
        g2d.drawImage(image, 0, 0, dimension.width, dimension.height, this);

    }

    // Gets the image mime type. Supported formats are image/jpeg and image/png.

    private String getImageType(byte[] image) throws IOException
    /**
     * Gets the file type of a supplied image. Raises an exception if the image
     * isn't compatible with with Amazon Lookout for Vision.
     *
     * @param image The image that you want to check.
     *
     * @return String The type of the image.
     */

    {
        InputStream is = new BufferedInputStream(new
ByteArrayInputStream(image));
        String mimeType = URLConnection.guessContentTypeFromStream(is);

        logger.log(Level.INFO, "Image type: {0}", mimeType);
```

```
        if (mimeType.equals("image/jpeg") || mimeType.equals("image/png")) {
            return mimeType;
        }
        // Not a supported file type.
        logger.log(Level.SEVERE, "Unsupported image type: {0}", mimeType);
        throw new IOException(String.format("Wrong image type. %s format isn't
supported.", mimeType));
    }

    public static void main(String[] args) throws Exception {

        String photo = null;
        String projectName = null;
        String modelVersion = null;

        final String USAGE = "\n" +
                "Usage:\n" +
                "    DetectAnomalies <project> <version> <image> \n\n" +
                "Where:\n" +
                "    project - The Lookout for Vision project.\n\n" +
                "    version - The version of the model within the project.\n\n"
+
                "    image - The path and filename of a local image. \n\n";

        try {

            if (args.length != 3) {
                System.out.println(USAGE);
                System.exit(1);
            }

            projectName = args[0];
            modelVersion = args[1];
            photo = args[2];
            ShowAnomalies panel = null;

            // Get the Lookout for Vision client.
            LookoutVisionClient lfvClient = LookoutVisionClient.builder()

.credentialsProvider(ProfileCredentialsProvider.create("lookoutvision-access"))
                .build();

            // Create frame and panel.
            JFrame frame = new JFrame(photo);
```

```
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            panel = new ShowAnomalies(lfvClient, projectName, modelVersion,
 photo);

            frame.setContentPane(panel);
            frame.pack();
            frame.setVisible(true);


        } catch (LookoutVisionException lfvError) {
            logger.log(Level.SEVERE, "Lookout for Vision client error: {0}:
 {1}",
                    new Object[] { lfvError.awsErrorDetails().errorCode(),
                            lfvError.awsErrorDetails().errorMessage() });
            System.out.println(String.format("lookout for vision client error:
 %s", lfvError.getMessage()));
            System.exit(1);

        } catch (FileNotFoundException fileError) {
            logger.log(Level.SEVERE, "Could not find file: {0}",
 fileError.getMessage());
            System.out.println(String.format("Could not find file: %s",
 fileError.getMessage()));
            System.exit(1);

        } catch (IOException ioError) {
            logger.log(Level.SEVERE, "IO error {0}", ioError.getMessage());
            System.out.println(String.format("IO error: %s",
 ioError.getMessage()));
            System.exit(1);
        }

    }
}
```

4. Jika Anda tidak berencana untuk terus menggunakan model Anda,hentikan model Anda.

Menemukan anomali denganAWS Lambdafungsi

AWS Lambda adalah layanan komputasi yang memungkinkan Anda menjalankan kode tanpa perlu menyediakan atau mengelola server. Misalnya, Anda dapat menganalisis gambar yang dikirimkan dari aplikasi seluler tanpa harus membuat server untuk meng-host kode aplikasi. Petunjuk berikut menunjukkan cara membuat fungsi Lambda dengan Python yang memanggilDetectAnomalies. Fungsi ini menganalisis gambar yang disediakan dan mengembalikan klasifikasi untuk keberadaan anomali dalam gambar itu. Instruksi termasuk contoh kode Python yang menunjukkan cara memanggil fungsi Lambda dengan gambar dalam bucket Amazon S3, atau gambar yang disediakan dari komputer lokal.

Topik

Langkah 1: BuatAWS Lambdafungsi (konsol)

Pada langkah ini, Anda membuat kosongAWSfungsi dan peran eksekusi IAM yang memungkinkan fungsi Anda memanggil`DetectAnomalies`operasi. Ini juga memberikan akses ke bucket Amazon S3 yang menyimpan gambar untuk dianalisis. Anda juga menentukan variabel lingkungan untuk berikut ini:

- Proyek Amazon Lookout for Vision dan versi model yang Anda inginkan untuk digunakan fungsi Lambda Anda.
- Batas kepercayaan diri yang Anda inginkan untuk digunakan model.

Kemudian Anda menambahkan kode sumber dan opsional lapisan ke fungsi Lambda.

Untuk membuatAWS Lambdafungsi (konsol)

1. Masuk ke AWS Management Console dan buka konsol AWS Lambda di https://console.aws.amazon.com/lambda/.
2. Pilih Buat fungsi. Untuk informasi lebih lanjut, lihatMembuat Fungsi Lambda dengan Konsol.
3. Pilih opsi berikut.

- Pilih Penulis dari scratch.
- Masukkan nilai untukNama fungsi.
- UntukRuntimepilihPython 3.10.

4. PilihBuat fungsiuntuk membuatAWS Lambdafungsi.
5. Pada halaman fungsi, PilihKonfigurasitab.
6. PadaVariabel lingkunganpanel, pilihMengedit.
7. Tambahkan variabel lingkungan berikut. Untuk setiap variabel pilihTambahkan variabel lingkungandan kemudian masukkan kunci variabel dan nilai.

Kunci	Nilai
PROJECT_NAME	Proyek Lookout for Vision yang berisi model yang ingin Anda gunakan.
MODEL_VERSION	Versi model yang ingin Anda gunakan.
KEPERCAYAAN	Nilai minumum (0-100) untuk keyakinan model bahwa prediksi itu anomali. Jika kepercayaan diri lebih rendah, klasifikasi dianggap normal.

8. PilihSimpanuntuk menyimpan variabel lingkungan.
9. PadaIzinpanel, Di bawahNama peran, pilih peran eksekusi untuk membuka peran di konsol IAM.
10. DalamIzintab, pilihTambahkan izindan kemudianBuat kebijakan inline.
11. PilihJSONdan mengganti kebijakan yang ada dengan kebijakan berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "lookoutvision:DetectAnomalies",
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DetectAnomaliesAccess"
        }
    ]
```

```
}
```

12. Pilih Selanjutnya.
13. DalamRincian kebijakan, masukkan nama untuk kebijakan, sepertiDetectAnomalies-akses.
14. Pilih Buat kebijakan.
15. Jika Anda menyimpan gambar untuk analisis dalam bucket Amazon S3, ulangi langkah 10—14.

    a. Untuk langkah 11, gunakan kebijakan berikut. Ganti*jalur ember/folder*dengan bucket Amazon S3 dan jalur folder ke gambar yang ingin Anda analisis.

    ```
    {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "S3Access",
                "Effect": "Allow",
                "Action": "s3:GetObject",
                "Resource": "arn:aws:s3:::bucket/folder path/*"
            }
        ]
    }
    ```

    b. Untuk langkah 13, pilih nama kebijakan yang berbeda, sepertiS3Bucket-akses.

Langkah 2: (Opsional) Buat layer (konsol)

Untuk menjalankan contoh ini, Anda tidak perlu melakukan langkah ini. Yang`DetectAnomalies`operasi disertakan dalam lingkungan default Lambda Python sebagai bagian dariAWSSDK untuk Python (Boto3). Jika bagian lain dari fungsi Lambda Anda perlu baru-baru iniAWSpembaruan layanan yang tidak ada di lingkungan Lambda Python default, lakukan langkah ini untuk menambahkan rilis Boto3 SDK terbaru sebagai lapisan ke fungsi Anda.

Pertama, Anda membuat arsip file.zip yang berisi Boto3 SDK. Anda kemudian membuat layer dan menambahkan arsip file.zip ke layer. Untuk informasi lebih lanjut, lihatMenggunakan layer dengan fungsi Lambda.

Untuk membuat dan menambahkan layer (konsol)

1. Buka prompt perintah dan masukkan perintah berikut.

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. Perhatikan nama file zip (boto3-layer.zip). Anda membutuhkannya pada langkah 6 dari prosedur ini.
3. Buka konsol AWS Lambda tersebut di https://console.aws.amazon.com/lambda/.
4. Di panel navigasi, pilih Layers (Lapisan).
5. Pilih Buat lapisan.
6. Masukkan nilai untukNamadanDeskripsi.
7. PilihUnggah file.zipdan pilihlahUnggah.
8. Di kotak dialog, pilih arsip file.zip (boto3-layer.zip) yang Anda buat pada langkah 1 dari prosedur ini.
9. Untuk runtime yang kompatibel, pilihPython 3.9.
10. PilihBuatuntuk membuat layer.
11. Pilih ikon menu panel navigasi.
12. Di panel navigasi, pilih Fungsi.
13. Dalam daftar sumber daya, pilih fungsi yang Anda buatLangkah 1: BuatAWS Lambdafungsi (konsol).
14. PilihKodetab.
15. DalamLapisanbagian, pilihTambahkan layer.
16. PilihLapisan kustom.
17. DalamLapisan kustom, pilih nama layer yang Anda masukkan pada langkah 6.
18. DalamVersipilih versi layer, yang seharusnya 1.
19. Pilih Tambahkan.

Langkah 3: Tambahkan kode Python (konsol)

Pada langkah ini, Anda menambahkan kode Python ke fungsi Lambda Anda dengan menggunakan editor kode konsol Lambda. Kode menganalisis gambar yang disediakan dengan`DetectAnomalies`dan mengembalikan klasifikasi (true jika gambar anomali, false jika gambar normal). Gambar yang disediakan dapat ditemukan di bucket Amazon S3 atau disediakan sebagai byte gambar yang dikodekan byte64.

Untuk menambahkan kode Python (konsol)

1. Jika Anda tidak berada di konsol Lambda, lakukan hal berikut:

   a. Buka konsol AWS Lambda tersebut di https://console.aws.amazon.com/lambda/.

   b. Buka fungsi Lambda yang Anda buatLangkah 1: BuatAWS Lambdafungsi (konsol).

2. PilihKodetab.

3. DalamSumber kode, ganti kode dilambda_function.pydengan yang berikut:

```
# 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 Amazon Lookout for Vision
 model.
"""
import base64
import imghdr
from os import environ
from io import BytesIO
import logging
import boto3

from botocore.exceptions import ClientError


logger = logging.getLogger(__name__)


# Get the model and confidence.
project_name = environ['PROJECT_NAME']
model_version = environ['MODEL_VERSION']
min_confidence = int(environ.get('CONFIDENCE', 50))

lookoutvision_client = boto3.client('lookoutvision')


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:

        file_name = ""

        # Determine image source.
        if 'image' in event:
            # Decode the encoded image
            image_bytes = event['image'].encode('utf-8')
            img_b64decoded = base64.b64decode(image_bytes)
            image_type = get_image_type(img_b64decoded)
            image = BytesIO(img_b64decoded)
            file_name = event['filename']

        elif 'S3Object' in event:
            bucket = boto3.resource('s3').Bucket(event['S3Object']['Bucket'])
            image_object = bucket.Object(event['S3Object']['Name'])
            image = image_object.get().get('Body').read()
            image_type = get_image_type(image)
            file_name = f"s3://{event['S3Object']['Bucket']}/{event['S3Object']
['Name']}"

        else:
            raise ValueError(
                'Invalid image source. Only base 64 encoded image bytes or images
 in S3 buckets are supported.')

        # Analyze the image.
        response = lookoutvision_client.detect_anomalies(
            ProjectName=project_name,
            ContentType=image_type,
            Body=image,
            ModelVersion=model_version)

        reject = reject_on_classification(
            response['DetectAnomalyResult'],
 confidence_limit=float(environ['CONFIDENCE'])/100)

        status = "anomalous" if reject else "normal"
```

```
        lambda_response = {
            "statusCode": 200,
            "body": {
                "Reject": reject,
                "RejectMessage": f"Image {file_name} is {status}."
            }
        }

    except ClientError as err:
        error_message = f"Couldn't analyze {file_name}. " + \
            err.response['Error']['Message']

        lambda_response = {
            'statusCode': 400,
            'body': {
                "Error": err.response['Error']['Code'],
                "ErrorMessage": error_message,
                "Image": file_name
            }
        }
        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),
                "Image": event['filename']
            }
        }
        logger.error("Error function %s: %s",
                     context.invoked_function_arn, format(val_error))

    return lambda_response

def get_image_type(image):
    """
    Gets the format of the image. Raises an error
    if the type is not PNG or JPEG.
    :param image: The image that you want to check.
    :return The type of the image.
```

```
    """
    image_type = imghdr.what(None, image)

    if image_type == "jpeg":
        content_type = "image/jpeg"
    elif image_type == "png":
        content_type = "image/png"
    else:
        logger.info("Invalid image type")
        raise ValueError(
            "Invalid file format. Supply a jpeg or png format file.")
    return content_type


def reject_on_classification(prediction, confidence_limit):
    """
    Returns True if the anomaly confidence is greater than or equal to
    the supplied confidence limit.
    :param image: The name of the image file that was analyzed.
    :param prediction: The DetectAnomalyResult object returned from DetectAnomalies
    :param confidence_limit: The minimum acceptable confidence. Float value between
 0 and 1.
    :return: True if the error condition indicates an anomaly, otherwise False.
    """

    reject = False

    if prediction['IsAnomalous'] and prediction['Confidence'] >= confidence_limit:
        reject = True
        reject_info = (f"Rejected: Anomaly confidence
 ({prediction['Confidence']:.2%}) is greater"
                       f" than limit ({confidence_limit:.2%})")
        logger.info("%s", reject_info)

    if not reject:
        logger.info("No anomalies found.")
    return reject
```

4. PilihMenyebarkanuntuk menyebarkan fungsi Lambda Anda.

Langkah 4: Coba fungsi Lambda

Pada langkah ini Anda menggunakan kode Python di komputer Anda untuk meneruskan gambar lokal, atau gambar dalam bucket Amazon S3, ke fungsi Lambda Anda. Gambar yang dilewatkan dari komputer lokal harus lebih kecil dari 6291456 byte. Jika gambar Anda lebih besar, unggah gambar ke bucket Amazon S3 dan panggil skrip dengan jalur Amazon S3 ke gambar. Untuk informasi tentang mengunggah file gambar ke bucket Amazon S3, lihatMengunggah objek.

Pastikan Anda menjalankan kode dalam hal yang samaAWSWilayah di mana Anda membuat fungsi Lambda. Anda dapat melihatAWSWilayah untuk fungsi Lambda Anda di bilah navigasi halaman detail fungsi diKonsol Lambda.

JikaAWS Lambdafungsi mengembalikan kesalahan batas waktu, memperpanjang periode batas waktu untuk fungsi fungsi Lambda, Untuk informasi lebih lanjut, lihatMengkonfigurasi batas waktu fungsi (konsol).

Untuk informasi selengkapnya tentang menjalankan fungsi Lambda dari kode Anda, lihatMemohonAWS LambdaFungsi.

Untuk mencoba fungsi Lambda

1. Jika Anda belum melakukannya, lakukan hal berikut:

   a. Pastikan pengguna menggunakan kode klien memiliki`lambda:InvokeFunction`izin. Anda dapat menggunakan izin berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LambdaPermission",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "ARN for lambda function"
        }
    ]
}
```

   Anda bisa mendapatkan ARN untuk fungsi fungsi Lambda Anda dari ikhtisar fungsi diKonsol Lambda.

Untuk menyediakan akses, tambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup diAWS IAM Identity Center:

  Buat set izin. Ikuti instruksi diBuat set izindi dalamAWS IAM Identity CenterPanduan Pengguna.

- Pengguna yang dikelola dalam IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti instruksi diMembuat peran untuk penyedia identitas pihak ketiga (federasi)di dalamPanduan Pengguna IAM.

- Pengguna IAM:
  - Buat peran yang dapat diasumsikan pengguna Anda. Ikuti instruksi diMembuat peran untuk pengguna IAMdi dalamPanduan Pengguna IAM.
  - (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti instruksi diMenambahkan izin ke pengguna (konsol)di dalamPanduan Pengguna IAM.

b. Instal dan konfigurasikanAWSSDK untuk Python. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.

c. Mulai modelnyayang Anda tentukan pada langkah 7Langkah 1: BuatAWS Lambdafungsi (konsol).

2. Simpan kode berikut ke file bernama`client.py`.

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

"""
Purpose: Shows how to call the anomaly detection
AWS Lambda function.
"""
from botocore.exceptions import ClientError

import argparse
import logging
import base64
import json
import boto3
from os import environ
```

```
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, "filename": image}

    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()

        # Analyze image and display results.

        result = analyze_image(args.function, args.image)

        status = result['statusCode']

        if status == 200:
            classification = result['body']
            print(f"classification: {classification['Reject']}")
            print(f"Message: {classification['RejectMessage']}")
        else:
            print(f"Error: {result['statusCode']}")
            print(f"Message: {result['body']}")

    except ClientError as error:
        logging.error(error)
        print(error)


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

3. Jalankan kode tersebut. Untuk argumen baris perintah, berikan nama fungsi Lambda dan jalur ke gambar lokal yang ingin Anda analisis. Misalnya:

```
python client.py function_name /bucket/path/image.jpg
```

Jika berhasil, output adalah klasifikasi untuk anomali yang ditemukan dalam gambar. Jika klasifikasi tidak dikembalikan, pertimbangkan untuk menurunkan nilai kepercayaan yang Anda tetapkan pada langkah 7Langkah 1: BuatAWS Lambdafungsi (konsol).

4. Jika Anda telah selesai dengan fungsi Lambda dan model tidak digunakan oleh aplikasi lain,hentikan modelnya. Ingatlah untukmulai modellain kali Anda ingin menggunakan fungsi Lambda.

Menggunakan model Amazon Lookout for Vision pada perangkat edge

Anda dapat menggunakan model Amazon Lookout for Vision pada perangkat edge yang dikelola oleh. AWS IoT Greengrass Version 2 AWS IoT Greengrass adalah layanan runtime dan cloud edge Internet of Things (IoT) open source. Anda dapat menggunakannya untuk membangun, menyebarkan, dan mengelola aplikasi IoT di perangkat Anda. Untuk informasi selengkapnya, lihat AWS IoT Greengrass.

Anda menerapkan model Amazon Lookout for Vision yang sama yang telah Anda latih di cloud ke perangkat edge yang kompatibel. AWS IoT Greengrass V2 Anda kemudian dapat menggunakan model yang digunakan untuk melakukan deteksi anomali di tempat, seperti lantai pabrik, tanpa terus-menerus mengalirkan data ke cloud. Dengan begitu Anda dapat meminimalkan biaya bandwidth dan mendeteksi anomali secara lokal dengan analisis gambar waktu nyata.

> **Tip**
>
> Sebelum menerapkan model Lookout for VisionAWS IoT Greengrass, kami sarankan Anda membaca panduan pengembang. AWS IoT Greengrass Version 2 Untuk informasi selengkapnya, lihat Apa itu AWS IoT Greengrass? .

Untuk menggunakan model Lookout for Vision pada AWS IoT Greengrass V2 perangkat inti, Anda menerapkan model dan perangkat lunak pendukung sebagai komponen ke perangkat inti. Komponen adalah modul perangkat lunak, seperti model Lookout for Vision, yang berjalan pada perangkat inti Greengrass. Ada dua bentuk komponen. Komponen kustom adalah komponen yang Anda buat dan hanya dapat diakses oleh Anda. Hal ini juga dikenal sebagai komponen pribadi. Komponen yang AWS disediakan adalah komponen pra-bangun yang AWS menyediakan. Hal ini juga dikenal sebagai komponen publik. Untuk informasi selengkapnya, lihat https://docs.aws.amazon.com/greengrass/v2/developerguide/public-components.html.

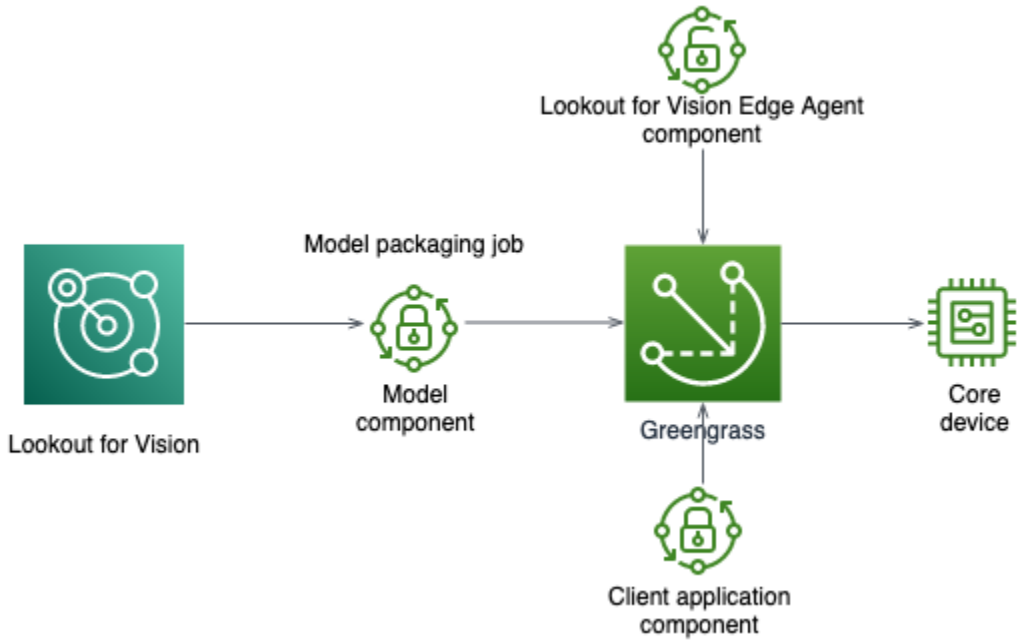
Komponen yang Anda terapkan ke perangkat inti untuk model Lookout for Vision dan perangkat lunak pendukung adalah:

- Komponen model. Komponen kustom yang berisi model Lookout for Vision Anda. Untuk membuat komponen model, Anda menggunakan Lookout for Vision untuk membuat pekerjaan pengemasan model. Pekerjaan pengemasan model menciptakan komponen untuk model dan membuatnya

tersedia sebagai komponen khusus di dalamnyaAWS IoT Greengrass V2. Untuk informasi selengkapnya, lihat Mengemas model Amazon Lookout for Vision Anda.

- Komponen aplikasi klien. Komponen kustom yang Anda buat yang mengimplementasikan kode untuk kebutuhan bisnis Anda. Misalnya, menemukan papan sirkuit anomali dari gambar yang diambil setelah perakitan. Untuk informasi selengkapnya, lihat Menulis komponen aplikasi klien Anda.
- Komponen Amazon Lookout for Vision Edge Agent. Komponen yang AWS disediakan yang menyediakan API untuk menggunakan dan mengelola model Anda. Misalnya, kode dalam komponen aplikasi klien Anda dapat menggunakan `DetectAnomalies` API untuk mendeteksi anomali pada gambar. Komponen Lookout for Vision Edge Agent adalah ketergantungan komponen model. Ini secara otomatis diinstal pada perangkat inti ketika Anda menggunakan komponen model. Untuk informasi selengkapnya, lihat Referensi API Amazon Lookout for Vision Edge Agent.

Setelah Anda membuat komponen model dan komponen aplikasi klien, Anda dapat menggunakan AWS IoT Greengrass V2 untuk menyebarkan komponen dan dependensi ke perangkat inti. Untuk informasi selengkapnya, lihat Menerapkan komponen Anda ke perangkat.



Important

Prediksi yang dibuat model Anda `DetectAnomalies` pada perangkat inti mungkin berbeda dari prediksi yang dibuat menggunakan model yang sama yang dihosting di cloud. Kami

menyarankan Anda menguji model Anda pada perangkat inti sebelum menggunakannya di lingkungan produksi.
Untuk mengurangi ketidakcocokan prediksi antara model yang dihosting perangkat dan model yang dihosting cloud, kami sarankan untuk meningkatkan jumlah gambar normal dan anomali dalam kumpulan data pelatihan Anda. Kami tidak menyarankan untuk menggunakan kembali gambar yang ada untuk meningkatkan ukuran kumpulan data pelatihan.

Menyebarkan model dan komponen aplikasi klien ke perangkat AWS IoT Greengrass Version 2 inti

Prosedur untuk menerapkan model Amazon Lookout for Vision dan komponen aplikasi klien pada perangkat AWS IoT Greengrass Version 2 inti adalah sebagai berikut:

1. Siapkan perangkat inti Anda denganAWS IoT Greengrass Version 2.
2. Buat pekerjaan pengemasan model dengan menggunakan Lookout for Vision. Pekerjaan menciptakan komponen model Anda.
3. Tulis komponen aplikasi klien. Komponen ini mengimplementasikan logika bisnis Anda.
4. Menyebarkan komponen model dan komponen aplikasi klien ke perangkat inti dengan menggunakanAWS IoT Greengrass V2.

Setelah komponen dan dependensi dikerahkan ke perangkat inti, Anda dapat menggunakan model pada perangkat inti.

> **Note**
>
> Anda harus menggunakan AWS Wilayah dan AWS akun yang sama untuk membuat dan menerapkan model Lookout for Vision dan komponen aplikasi klien Anda.

AWS IoT Greengrass Version 2persyaratan perangkat inti

Untuk menggunakan model Amazon Lookout for Vision pada AWS IoT Greengrass Version 2 perangkat inti, model Anda memiliki berbagai persyaratan perangkat inti.

Topik

Perangkat yang diuji, arsitektur chip, dan sistem operasi

Kami berharap Amazon Lookout for Vision bekerja pada perangkat keras berikut:

- Arsitektur CPU
  - X86_64 (versi 64-bit dari set instruksi x86)
  - Aarch64 (CPU ARMv8 64-bit)
- (Inferensi dipercepat GPU saja) Akselerator GPU NVIDIA dengan kapasitas memori yang cukup (Setidaknya 6,0 GB untuk model yang sedang berjalan).

Tim Amazon Lookout for Vision telah menguji model Lookout for Vision pada perangkat, arsitektur chip, dan sistem operasi berikut.

Perangkat

Perangkat	Sistem operasi	Arsitektur	Akselerator	Opsi kompiler

jetson_xavier (NVIDIA® Jetson AGX Xavier)	Linux	Aarch64	NVIDIA	`{"gpu-code": "sm_72", "trt-ver": "7.1.3", "cuda-ver": "10.2"}`
				
				`{"gpu-code": "sm_72", "trt-ver": "8.2.1", "cuda-ver": "10.2"}`

Perangkat	Sistem operasi	Arsitektur	Akselerator	Opsi kompiler
g4dn.xlarge (Instans EC2 (G4) dengan GPU Inti Tensor NVIDIA T4)	Linux	X86_64/X86-64	NVIDIA	`{"gpu-code": "sm_75", "trt-ver": "7.1.3", "cuda-ver": "10.2"}`
g5.xlarge (Instans EC2 (G5) dengan GPU Tensor Core NVIDIA A10G)	Linux	X86_64/X86-64	NVIDIA	`{"gpu-code": "sm_80", "trt-ver": "8.2.0", "cuda-ver": "11.2"}`
c5.2xlarge (Instans Amazon EC2 C5)	Linux	X86_64/X86-64	CPU	`{"mcpu": "core-avx2"}`

Memori dan penyimpanan perangkat inti

Untuk menjalankan satu model dan Amazon Lookout for Vision Edge Agent, perangkat inti Anda memiliki persyaratan memori dan penyimpanan berikut. Anda mungkin memerlukan lebih banyak memori dan penyimpanan untuk komponen aplikasi klien Anda.

- Penyimpanan - Setidaknya 1,5 GB.
- Memori — Setidaknya 6.0 GB untuk model yang sedang berjalan.

Perangkat lunak yang dibutuhkan

Perangkat inti membutuhkan perangkat lunak berikut.

Perangkat Jetson

Jika perangkat inti Anda adalah perangkat Jetson, Anda memerlukan perangkat lunak berikut yang diinstal pada perangkat inti.

Perangkat lunak	Versi yang didukung
SDK Jetpack	4.4 hingga 4.6.1
Lingkungan virtual Python dan Python untuk Lookout for Vision Edge Agent versi 1.x	3.8 atau 3.9

Perangkat keras X86

Jika perangkat inti Anda menggunakan perangkat keras x86, Anda memerlukan perangkat lunak berikut yang diinstal pada perangkat inti.

Inferensi CPU

Perangkat lunak	Versi yang didukung
Lingkungan virtual Python dan Python untuk Lookout for Vision Edge Agent versi 1.x	3.8 atau 3.9

Inferensi dipercepat GPU

Versi perangkat lunak bervariasi tergantung pada mikroarsitektur GPU NVIDIA yang Anda gunakan.

GPU NVIDIA dengan mikroarsitektur sebelum Ampere (kemampuan komputasi kurang dari 8.0)

Perangkat lunak yang diperlukan untuk GPU NVIDIA dengan mikroarsitektur sebelum Ampere (kemampuan komputasi yang kurang dari 8.0). `gpu-code`Harus kurang dari`sm_80`.

Perangkat lunak	Versi yang didukung
NVIDIA CUDA	10.2

Perangkat lunak	Versi yang didukung
TensorRT NVIDIA	Setidaknya 7.1.3 dan kurang dari 8.0.0
Lingkungan virtual Python dan Python untuk Lookout for Vision Edge Agent versi 1.x	3.8 atau 3.9

GPU NVIDIA dengan mikroarsitektur Ampere (kemampuan komputasi 8.0)

Perangkat lunak yang diperlukan untuk GPU NVIDIA dengan mikroarsitektur Ampere (kemampuan komputasi adalah 8.0). `gpu-code`Harus`sm_80`).

Perangkat lunak	Versi yang didukung
NVIDIA CUDA	11.2
TensorRT NVIDIA	8.2.0
Lingkungan virtual Python dan Python untuk Lookout for Vision Edge Agent versi 1.x	3.8 atau 3.9

Menyiapkan perangkat AWS IoT Greengrass Version 2 inti Anda

Amazon Lookout for Vision AWS IoT Greengrass Version 2 digunakan untuk menyederhanakan penerapan komponen model, komponen Amazon Lookout for Vision Edge Agent, dan komponen aplikasi klien ke perangkat inti Anda. AWS IoT Greengrass V2 Untuk informasi tentang perangkat dan perangkat keras yang dapat Anda gunakan, lihatAWS IoT Greengrass Version 2persyaratan perangkat inti.

Menyiapkan perangkat inti Anda

Gunakan informasi berikut untuk menyiapkan perangkat inti Anda.

Untuk mengatur perangkat inti

1. Siapkan pustaka GPU Anda. Jangan lakukan langkah ini jika Anda tidak menggunakan inferensi yang dipercepat GPU.

a. Verifikasi bahwa Anda memiliki GPU yang mendukung CUDA. Untuk informasi selengkapnya, lihat Memverifikasi Anda Memiliki GPU berkemampuan CUDA.

b. Siapkan CUDA, cuDNN, dan TensorRT di perangkat Anda dengan melakukan salah satu hal berikut:

- Jika Anda menggunakan perangkat Jetson, instal JetPack versi 4.4 - 4.6.1. Untuk informasi selengkapnya, lihat JetPack Arsip.
- Jika Anda menggunakan perangkat keras berbasis x86, dan mikroarsitektur GPU NVIDIA Anda sebelum Ampere (kemampuan komputasi kurang dari 8.0), lakukan hal berikut:
  1. Siapkan CUDA versi 10.2 dengan mengikuti petunjuk di Panduan Instalasi NVIDIA CUDA untuk Linux.
  2. Instal cuDNN, dengan mengikuti instruksi di Dokumentasi NVIDIA cuDNN.
  3. Siapkan TensorRT (versi 7.1.3 atau yang lebih baru, tetapi lebih awal dari 8.0.0) dengan mengikuti instruksi di NVIDIA TENSORRT DOCUMENTATION.
- Jika Anda menggunakan perangkat keras berbasis x86, dan mikroarsitektur GPU NVIDIA Anda adalah Ampere (kemampuan komputasi adalah 8.0), lakukan hal berikut:
  1. Siapkan CUDA (versi 11.2) dengan mengikuti petunjuk di Panduan Instalasi NVIDIA CUDA untuk Linux.
  2. Instal cuDNN, dengan mengikuti instruksi di Dokumentasi NVIDIA cuDNN.
  3. Siapkan TensorRT (versi 8.2.0) dengan mengikuti petunjuk di NVIDIA TENSORRT DOCUMENTATION.

2. Instal perangkat lunak AWS IoT Greengrass Version 2 inti pada perangkat inti Anda. Untuk informasi selengkapnya, lihat Menginstal perangkat lunak AWS IoT Greengrass Core di Panduan Pengembang. AWS IoT Greengrass Version 2

3. Untuk membaca dari bucket Amazon S3 yang menyimpan model, lampirkan izin ke peran IAM (peran pertukaran token) yang Anda buat selama penyiapan. AWS IoT Greengrass Version 2 Untuk informasi selengkapnya, lihat Mengizinkan akses ke bucket S3 untuk artefak komponen.

4. Pada prompt perintah, masukkan perintah berikut untuk menginstal Python dan lingkungan virtual Python ke perangkat inti.

```
sudo apt install python3.8 python3-venv python3.8-venv
```

5. Gunakan perintah berikut untuk menambahkan pengguna Greengrass ke grup video. Ini memungkinkan komponen yang digunakan Greengrass mengakses GPU:

```
sudo usermod -a -G video ggc_user
```

6. (Opsional) Jika Anda ingin memanggil Lookout for Vision Edge Agent API dari pengguna lain, tambahkan pengguna yang diperlukan ke. `ggc_group` Hal ini memungkinkan pengguna berkomunikasi dengan Lookout for Vision Edge Agent melalui soket Unix Domain:

```
sudo usermod -a -G ggc_group $(whoami)
```

Mengemas model Amazon Lookout for Vision Anda

Pekerjaan pengemasan model mengemas model Amazon Lookout for Vision sebagai komponen model.

Untuk membuat pekerjaan pengemasan model, Anda memilih model yang ingin Anda paket dan memberikan pengaturan untuk komponen model yang dibuat oleh pekerjaan. Anda hanya dapat mengemas model yang telah berhasil dilatih.

Anda dapat menggunakan konsol Lookout for Vision AWS atau SDK untuk membuat pekerjaan pengemasan model. Anda juga bisa mendapatkan informasi tentang pekerjaan pengemasan model yang Anda buat. Untuk informasi selengkapnya, lihat Mendapatkan informasi tentang pekerjaan pengemasan model. Anda dapat menggunakan AWS IoT Greengrass V2 konsol atau AWS SDK untuk menyebarkan komponen ke perangkat AWS IoT Greengrass Version 2 inti.

Topik

- Pengaturan Package
- Mengemas model Anda (Konsol)
- Kemasan model Anda (SDK)
- Mendapatkan informasi tentang pekerjaan pengemasan model

Pengaturan Package

Gunakan informasi berikut untuk memutuskan pengaturan paket untuk pekerjaan pengemasan model Anda.

Untuk membuat pekerjaan pengemasan model, lihat Mengemas model Anda (Konsol) atauKemasan model Anda (SDK).

Topik

Perangkat keras target

Anda dapat memilih perangkat target atau platform target untuk model Anda, tetapi tidak keduanya. Untuk informasi selengkapnya, lihat Perangkat yang diuji, arsitektur chip, dan sistem operasi.

Perangkat target

Perangkat target untuk model, seperti NVIDIA® Jetson AGX Xavier. Anda tidak perlu menentukan opsi kompiler.

Platform target

Amazon Lookout for Vision mendukung konfigurasi platform berikut:

- X86_64 (versi 64-bit dari set instruksi x86) dan arsitektur Aarch64 (ARMv8 64-bit CPU).
- Sistem operasi Linux.
- Inferensi menggunakan akselerator NVIDIA atau CPU.

Anda perlu menentukan opsi kompiler yang benar untuk platform target Anda.

Opsi kompiler

Opsi kompiler memungkinkan Anda menentukan platform target untuk perangkat AWS IoT Greengrass Version 2 inti Anda. Saat ini Anda dapat menentukan opsi kompiler berikut.

Akselerator NVIDIA

- `gpu-code`— Menentukan kode gpu perangkat inti yang menjalankan komponen model.
- `trt-ver`- Menentukan versi TensorRT dalam format xy.z.
- `cuda-ver`- Menentukan versi CUDA dalam format xy.

Akselerator CPU

- (Opsional) `mcpu` - menentukan set instruksi. Misalnya, `core-avx2`. Jika Anda tidak memberikan nilai, Lookout for Vision menggunakan nilainya. `core-avx2`

Anda menentukan opsi dalam format JSON. Misalnya:

```
{"gpu-code": "sm_75", "trt-ver": "7.1.3", "cuda-ver": "10.2"}
```

Untuk contoh lainnya, lihat Perangkat yang diuji, arsitektur chip, dan sistem operasi.

Pengaturan komponen

Pekerjaan pengemasan model menciptakan komponen model yang berisi model Anda. Pekerjaan menciptakan artefak yang AWS IoT Greengrass V2 digunakan untuk menyebarkan komponen model ke perangkat inti.

Anda tidak dapat membuat komponen model dengan nama komponen dan versi komponen yang sama dengan komponen yang ada.

Nama komponen

Nama untuk komponen model yang dibuat Lookout for Vision selama pengemasan model. Nama komponen yang Anda tentukan ditampilkan di AWS IoT Greengrass V2 konsol. Anda menggunakan nama komponen dalam resep yang Anda buat untuk komponen aplikasi klien. Untuk informasi selengkapnya, lihat Membuat komponen aplikasi klien.

Deskripsi komponen

(Opsional) Deskripsi untuk komponen model.

Versi komponen

Nomor versi untuk komponen model. Anda dapat menerima nomor versi default atau memilih sendiri. Nomor versi harus mengikuti sistem nomor versi semantik — major.minor.patch. Misalnya, versi 1.0.0 mewakili rilis utama pertama untuk sebuah komponen. Untuk informasi selengkapnya, lihat Versioning Semantik 2.0.0. Jika Anda tidak memberikan nilai, Lookout for Vision menggunakan nomor versi model Anda untuk menghasilkan versi untuk Anda.

Lokasi komponen

Lokasi Amazon S3 tempat Anda menginginkan pekerjaan pengemasan model untuk menyimpan artefak komponen model. Bucket Amazon S3 harus berada di Wilayah AWS dan akun AWS yang sama dengan yang Anda gunakan. AWS IoT Greengrass Version 2 Untuk membuat bucket Amazon S3, lihat Membuat bucket.

Tanda

Anda dapat mengidentifikasi, mengatur, mencari, dan memfilter komponen Anda dengan menggunakan tag. Setiap tag adalah label yang terdiri dari kunci dan nilai yang ditentukan pengguna. Tag dilampirkan ke komponen model saat pekerjaan pengemasan model membuat komponen model di Greengrass. Komponen adalah sumber daya AWS IoT Greengrass V2. Tag tidak dilampirkan ke salah satu sumber daya Lookout for Vision Anda, seperti model Anda. Untuk informasi selengkapnya, lihat Menandai sumber daya AWS.

Mengemas model Anda (Konsol)

Anda dapat membuat pekerjaan pengemasan model dengan menggunakan konsol Amazon Lookout for Vision.

Untuk informasi tentang pengaturan paket, lihatPengaturan Package.

Untuk mengemas model (konsol)

1. Buat bucket Amazon S3, atau gunakan kembali bucket yang sudah ada, yang digunakan Lookout for Vision untuk menyimpan artefak pekerjaan pengemasan (komponen model).
2. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
3. Pilih Mulai.
4. Di panel navigasi kiri, pilih Proyek.
5. Di bagian Proyek, pilih proyek yang berisi model yang ingin Anda paketkan.
6. Di panel navigasi kiri, di bawah nama proyek, pilih paket model Edge.
7. Di bagian Pekerjaan pengemasan model, pilih Buat pekerjaan pengemasan model.
8. Masukkan pengaturan untuk paket. Untuk informasi selengkapnya, lihat Pengaturan Package.
9. Pilih Buat pekerjaan pengemasan model.
10. Tunggu sampai pekerjaan pengemasan selesai. Pekerjaan selesai ketika status pekerjaan adalah Sukses.
11. Pilih pekerjaan pengemasan di bagian Pekerjaan pengemasan Model.
12. Pilih Lanjutkan penerapan di Greengrass untuk melanjutkan penerapan komponen model Anda di. AWS IoT Greengrass Version 2 Untuk informasi selengkapnya, lihat Menerapkan komponen Anda ke perangkat.

Kemasan model Anda (SDK)

Anda mengemas model sebagai komponen model dengan membuat pekerjaan pengemasan model. Untuk membuat pekerjaan pengemasan model, Anda memanggil StartModelPackagingJobAPI. Pekerjaan itu mungkin membutuhkan waktu cukup lama untuk diselesaikan. Untuk mengetahui status saat ini, panggil DescribeModelPackagingJobdan periksa `Status` bidang dalam respons.

Untuk informasi tentang pengaturan paket, lihatPengaturan Package.

Prosedur berikut menunjukkan kepada Anda cara memulai pekerjaan pengemasan dengan menggunakan AWS CLI. Anda dapat mengemas model untuk platform target atau perangkat target. Misalnya kode Java, lihat StartModelPackagingJob.

Untuk mengemas model Anda (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Pastikan Anda memiliki izin yang benar untuk memulai pekerjaan pengemasan model. Untuk informasi lebih lanjut, lihat StartModelPackagingJob.
3. Gunakan perintah CLI berikut untuk mengemas model Anda baik untuk perangkat target atau platform target.

   Target platform

   Perintah CLI berikut menunjukkan cara mengemas model untuk platform target dengan akselerator NVIDIA.

   Ubah nilai berikut:

   - `project_name`untuk nama proyek yang berisi model yang ingin Anda paket.
   - `model_version`ke versi model yang ingin Anda paket.
   - (Opsional) `description` untuk deskripsi untuk pekerjaan pengemasan model Anda.
   - `architecture`ke arsitektur (`ARM64`atau`X86_64`) perangkat AWS IoT Greengrass Version 2 inti tempat Anda menjalankan komponen model.
   - `gpu_code`ke kode gpu perangkat inti tempat Anda menjalankan komponen model.
   - `trt_ver`ke versi TensorRT yang telah Anda instal di perangkat inti Anda.
   - `cuda_ver`ke versi CUDA yang telah Anda instal di perangkat inti Anda.

- `component_name`ke nama untuk komponen model yang ingin Anda buatAWS IoT Greengrass V2.
- (Opsional) `component_version` ke versi untuk komponen model yang dibuat oleh pekerjaan pengemasan. Gunakan format `major.minor.patch`. Misalnya, 1.0.0 mewakili rilis utama pertama untuk sebuah komponen.
- `bucket`ke ember Amazon S3 tempat pekerjaan pengemasan menyimpan artefak komponen model.
- `prefix`ke lokasi di dalam ember Amazon S3 tempat pekerjaan pengemasan menyimpan artefak komponen model.
- (Opsional) `component_description` untuk deskripsi untuk komponen model.
- (Opsional) `tag_key1` dan `tag_key2` ke kunci untuk tag yang dilampirkan ke komponen model.
- (Opsional) `tag_value1` dan `tag_value2` ke nilai kunci untuk tag yang dilampirkan ke komponen model.

```
aws lookoutvision start-model-packaging-job \
  --project-name project_name \
  --model-version model_version \
  --description="description" \
  --configuration
 "Greengrass={TargetPlatform={Os='LINUX',Arch='architecture',Accelerator='NVIDIA'},Compi
code\": \"gpu_code\", \"trt-ver\": \"trt_ver\", \"cuda-ver\":
 \"cuda_ver\"}',S3OutputLocation={Bucket='bucket',Prefix='prefix'},ComponentName='Compon
 {Key='tag_key2',Value='tag_value2'}]}" \
  --profile lookoutvision-access
```

Misalnya:

```
aws lookoutvision start-model-packaging-job \
  --project-name test-project-01 \
  --model-version 1 \
  --description="Model Packaging Job for G4 Instance using TargetPlatform
 Option" \
  --configuration
 "Greengrass={TargetPlatform={Os='LINUX',Arch='X86_64',Accelerator='NVIDIA'},CompilerOpt
code\": \"sm_75\", \"trt-ver\": \"7.1.3\", \"cuda-ver\":
 \"10.2\"}',S3OutputLocation={Bucket='bucket',Prefix='test-project-01/
```

```
folder'},ComponentName='SampleComponentNameX86TargetPlatform',ComponentVersion='0.1.0',C
 is my component',Tags=[{Key='modelKey0',Value='modelValue'},
 {Key='modelKey1',Value='modelValue'}]}" \
  --profile lookoutvision-access
```

Target Device

Gunakan perintah CLI berikut untuk mengemas model untuk perangkat target.

Ubah nilai berikut:

- `project_name`untuk nama proyek yang berisi model yang ingin Anda paket.
- `model_version`ke versi model yang ingin Anda paket.
- (Opsional) `description` untuk deskripsi untuk pekerjaan pengemasan model Anda.
- `component_name`eke nama untuk komponen model yang ingin Anda buatAWS IoT Greengrass V2.
- (Opsional) `component_version` ke versi untuk komponen model yang dibuat oleh pekerjaan pengemasan. Gunakan format `major.minor.patch`. Misalnya, 1.0.0 mewakili rilis utama pertama untuk sebuah komponen.
- `bucket`ke ember Amazon S3 tempat pekerjaan pengemasan menyimpan artefak komponen model.
- `prefix`ke lokasi di dalam ember Amazon S3 tempat pekerjaan pengemasan menyimpan artefak komponen model.
- (Opsional) `component_description` untuk deskripsi untuk komponen model.
- (Opsional) `tag_key1` dan `tag_key2` ke kunci untuk tag yang dilampirkan ke komponen model.
- (Opsional) `tag_value1` dan `tag_value2` ke nilai kunci untuk tag yang dilampirkan ke komponen model.

```
aws lookoutvision start-model-packaging-job \
  --project-name project_name \
  --model-version model_version \
  --description="description" \
  --configuration
 "Greengrass={TargetDevice='jetson_xavier',S3OutputLocation={Bucket='bucket',Prefix='pre
 {Key='tag_key2',Value='tag_value2'}]}" \
  --profile lookoutvision-access
```

Misalnya:

```
aws lookoutvision start-model-packaging-job \
  --project-name project_01 \
  --model-version 1 \
  --description="description" \
  --configuration
 "Greengrass={TargetDevice='jetson_xavier',S3OutputLocation={Bucket='bucket',Prefix='com
 model component',Tags=[{Key='tag_key1',Value='tag_value1'},
 {Key='tag_key2',Value='tag_value2'}]}" \
  --profile lookoutvision-access
```

4. Perhatikan nilai JobName dalam respon. Anda membutuhkannya di langkah berikutnya. Misalnya:

```
{
    "JobName": "6bcfd0ff-90c3-4463-9a89-6b4be3daf972"
}
```

5. Gunakan `DescribeModelPackagingJob` untuk mendapatkan status pekerjaan saat ini. Ubah yang berikut ini:

   - `project_name`dengan nama proyek yang Anda gunakan.
   - `job_name`dengan nama pekerjaan yang Anda catat di langkah sebelumnya.

```
aws lookoutvision describe-model-packaging-job \
    --project-name project_name \
    --job-name job_name \
    --profile lookoutvision-access
```

   Pekerjaan pengemasan model selesai jika `Status` nilainya`SUCCEEDED`. Jika nilainya berbeda, tunggu sebentar dan coba lagi.

6. Lanjutkan penerapan menggunakanAWS IoT Greengrass V2. Untuk informasi selengkapnya, lihat Menerapkan komponen Anda ke perangkat.

Mendapatkan informasi tentang pekerjaan pengemasan model

Anda dapat menggunakan konsol Amazon Lookout for Vision AWS dan SDK untuk mendapatkan informasi tentang pekerjaan pengemasan model yang Anda buat.

Topik

Mendapatkan informasi pekerjaan pengemasan model (Konsol)

Untuk mendapatkan informasi pekerjaan pengemasan model (konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi kiri, pilih Proyek.
4. Di bagian Proyek, pilih proyek yang berisi pekerjaan pengemasan model yang ingin Anda lihat.
5. Di panel navigasi kiri, di bawah nama proyek, pilih paket model Edge.
6. Di bagian Pekerjaan pengemasan model, pilih pekerjaan pengemasan model yang ingin Anda lihat. Halaman detail untuk pekerjaan pengemasan model ditampilkan.

Mendapatkan informasi pekerjaan pengemasan model (SDK)

Anda dapat menggunakan AWS SDK untuk membuat daftar pekerjaan pengemasan model dalam proyek dan mendapatkan informasi tentang pekerjaan pengemasan model tertentu.

Daftar pekerjaan pengemasan model

Anda dapat membuat daftar pekerjaan pengemasan model dalam proyek dengan memanggil ListModelPackagingJobsAPI. Responsnya mencakup daftar ModelPackagingJobMetadataobjek yang memberikan informasi tentang setiap pekerjaan pengemasan model. Juga disertakan adalah token pagination yang dapat Anda gunakan untuk mendapatkan set hasil berikutnya, jika daftarnya tidak lengkap.

Untuk membuat daftar pekerjaan pengemasan model Anda

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan perintah CLI berikut. `project_name`Ubah nama proyek yang ingin Anda gunakan.

```
aws lookoutvision list-model-packaging-jobs \
  --project-name project_name \
  --profile lookoutvision-access
```

Jelaskan pekerjaan pengemasan model

Gunakan DescribeModelPackagingJobAPI untuk mendapatkan informasi tentang pekerjaan pengemasan model. Respons adalah ModelPackagingDescriptionobjek yang mencakup status pekerjaan saat ini dan informasi lainnya.

Untuk mendeskripsikan sebuah paket

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan perintah CLI berikut. Ubah yang berikut ini:
   - `project_name`dengan nama proyek yang Anda gunakan.
   - `job_name`dengan nama pekerjaan. Anda mendapatkan nama pekerjaan (`JobName`) saat Anda menelepon StartModelPackagingJob.

```
aws lookoutvision describe-model-packaging-job \
    --project-name project_name \
    --job-name job_name \
    --profile lookoutvision-access
```

Menulis komponen aplikasi klien Anda

Komponen aplikasi klien adalah komponen kustom AWS IoT Greengrass Version 2 yang Anda tulis. Ini mengimplementasikan logika bisnis yang Anda perlukan untuk menggunakan model Amazon Lookout for Vision pada perangkat inti. AWS IoT Greengrass Version 2

Untuk mengakses model, komponen aplikasi klien Anda menggunakan komponen Lookout for Vision Edge Agent. Komponen Lookout for Vision Edge Agent menyediakan API yang Anda gunakan untuk menganalisis gambar dengan model dan mengelola model pada perangkat inti.

Lookout for Vision Edge Agent API diimplementasikan menggunakan gRPC, yang merupakan protokol untuk membuat panggilan prosedur jarak jauh. Untuk informasi lebih lanjut, lihat gRPC. Untuk menulis kode Anda, Anda dapat menggunakan bahasa apa pun yang didukung oleh gRPC. Kami memberikan contoh kode Python. Untuk informasi selengkapnya, lihat Menggunakan model dalam komponen aplikasi klien Anda.

> **Note**
>
> Komponen Lookout for Vision Edge Agent adalah ketergantungan komponen model yang Anda gunakan. Ini secara otomatis digunakan ke perangkat inti saat Anda menyebarkan komponen model ke perangkat inti.

Untuk menulis komponen aplikasi klien, Anda melakukan hal berikut.

1. Siapkan lingkungan Anda untuk menggunakan gRPC dan menginstal pustaka pihak ketiga.
2. Tulis kode untuk menggunakan model.
3. Terapkan kode sebagai komponen khusus ke perangkat inti.

Untuk contoh komponen aplikasi klien yang menunjukkan cara melakukan deteksi anomali dalam pipeline GStreamer kustom, lihat https://github.com/awslabs/ -gstreamer. aws-greengrass-labs-lookoutvision

Menyiapkan lingkungan Anda

Untuk menulis kode klien, lingkungan pengembangan Anda terhubung dari jarak jauh ke perangkat AWS IoT Greengrass Version 2 inti tempat Anda telah menerapkan komponen dan dependensi model Amazon Lookout for Vision. Atau, Anda dapat menulis kode pada perangkat inti. Untuk informasi selengkapnya, lihat alat pengembangan AWS IoT Greengrass dan Mengembangkan komponen AWS IoT Greengrass.

Kode klien Anda harus menggunakan klien gRPC untuk mengakses Amazon Lookout for Vision Edge Agent. Bagian ini menunjukkan cara mengatur lingkungan pengembangan Anda dengan gRPC dan menginstal dependensi pihak ketiga yang diperlukan untuk kode contoh. `DetectAnomalies`

Setelah Anda selesai menulis kode klien Anda, Anda membuat komponen kustom dan menerapkan komponen kustom ke perangkat edge Anda. Untuk informasi selengkapnya, lihat Membuat komponen aplikasi klien.

Topik

- Menyiapkan gRPC
- Menambahkan dependensi pihak ketiga

Menyiapkan gRPC

Di lingkungan pengembangan Anda, Anda memerlukan klien gRPC yang Anda gunakan dalam kode Anda untuk memanggil Lookout for Vision Edge Agent API. Untuk melakukan ini, Anda membuat rintisan gRPC dengan menggunakan file definisi `.proto` layanan untuk Lookout for Vision Edge Agent.

> **Note**
>
> Anda juga bisa mendapatkan file definisi layanan dari bundel aplikasi Lookout for Vision Edge Agent. Bundel aplikasi diinstal ketika komponen Lookout for Vision Edge Agent diinstal sebagai dependensi komponen model. Bundel aplikasi terletak di`/greengrass/v2/packages/artifacts-unarchived/aws.iot.lookoutvision.EdgeAgent/`*`edge_agent_version`*`/lookoutvision_edge_agent`. Ganti `edge_agent_version` dengan versi Lookout for Vision Edge Agent yang Anda gunakan. Untuk mendapatkan bundel aplikasi, Anda perlu menyebarkan Lookout for Vision Edge Agent ke perangkat inti.

Untuk mengatur gRPC

1. Unduh file zip, proto.zip. File zip berisi file definisi layanan.proto ()`edge-agent.proto`.
2. Buka zip konten.
3. Buka prompt perintah dan arahkan ke folder yang berisi`edge-agent.proto`.
4. Gunakan perintah berikut untuk menghasilkan antarmuka klien Python.

```
%%bash
python3 -m pip install grpcio
python3 -m pip install grpcio-tools
```

```
python3 -m grpc_tools.protoc --proto_path=. --python_out=. --grpc_python_out=.
 edge-agent.proto
```

Jika perintah berhasil, rintisan `edge_agent_pb2_grpc.py` dan `edge_agent_pb2.py` dibuat di direktori kerja.

5. Tulis kode klien yang menggunakan model Anda. Untuk informasi selengkapnya, lihat Menggunakan model dalam komponen aplikasi klien Anda.

Menambahkan dependensi pihak ketiga

Kode `DetectAnomalies` contoh menggunakan pustaka Pillow untuk bekerja dengan gambar. Untuk informasi selengkapnya, lihat Mendeteksi Anomali dengan menggunakan byte gambar.

Gunakan perintah berikut untuk menginstal perpustakaan Pillow.

```
python3 -m pip install Pillow
```

Menggunakan model dalam komponen aplikasi klien Anda

Langkah-langkah untuk menggunakan model dari komponen aplikasi klien mirip dengan menggunakan model yang dihosting di cloud.

1. Mulai menjalankan model.
2. Mendeteksi anomali dalam gambar.
3. Hentikan model, jika tidak lagi diperlukan.

Amazon Lookout for Vision Edge Agent menyediakan API untuk memulai model, mendeteksi anomali dalam gambar, dan menghentikan model. Anda juga dapat menggunakan API untuk membuat daftar model pada perangkat dan mendapatkan informasi tentang model yang diterapkan. Untuk informasi selengkapnya, lihat Referensi API Amazon Lookout for Vision Edge Agent.

Anda bisa mendapatkan informasi kesalahan dengan memeriksa kode status gRPC. Untuk informasi selengkapnya, lihat Mendapatkan informasi kesalahan.

Untuk menulis kode Anda, Anda dapat menggunakan bahasa apa pun yang didukung oleh gRPC. Kami memberikan contoh kode Python.

Topik

- Menggunakan rintisan di komponen aplikasi klien Anda
- Memulai model
- Mendeteksi anomali
- Menghentikan model
- Daftar model pada perangkat
- Menggambarkan model
- Mendapatkan informasi kesalahan

Menggunakan rintisan di komponen aplikasi klien Anda

Gunakan kode berikut untuk mengatur akses ke model Anda melalui Lookout for Vision Edge Agent.

```
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2

# Creating stub.
 with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    # Add additional code that works with Edge Agent in this block to prevent resources
 leakage
```

Memulai model

Anda memulai model dengan memanggil StartModel API. Model mungkin membutuhkan waktu beberapa saat untuk memulai. Anda dapat memeriksa status saat ini dengan meneleponDescribeModel. Model berjalan jika nilai `status` bidang sedang berjalan.

Kode contoh

Ganti *`component_name`* dengan nama komponen model Anda.

```
import time

import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2
```

```
model_component_name = "component_name"


def start_model_if_needed(stub, model_name):
    # Starting model if needed.
    while True:
        model_description_response =
 stub.DescribeModel(pb2.DescribeModelRequest(model_component=model_name))
        print(f"DescribeModel() returned {model_description_response}")
        if model_description_response.model_description.status == pb2.RUNNING:
            print("Model is already running.")
            break
        elif model_description_response.model_description.status == pb2.STOPPED:
            print("Starting the model.")
            stub.StartModel(pb2.StartModelRequest(model_component=model_name))
            continue
        elif model_description_response.model_description.status == pb2.FAILED:
            raise Exception(f"model {model_name} failed to start")
        print(f"Waiting for model to start.")
        if model_description_response.model_description.status != pb2.STARTING:
            break
        time.sleep(1.0)


# Creating stub.
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    start_model_if_needed(stub, model_component_name)
```

Mendeteksi anomali

Anda menggunakan DetectAnomalies API untuk mendeteksi anomali dalam gambar.

`DetectAnomalies`Operasi mengharapkan bitmap gambar diteruskan dalam format RGB888 yang dikemas. Byte pertama mewakili saluran merah, byte kedua mewakili saluran hijau, dan byte ketiga mewakili saluran biru. Jika Anda memberikan gambar dalam format yang berbeda, seperti BGR, prediksi dari salah DetectAnomalies .

Secara default, OpenCV menggunakan format BGR untuk bitmap gambar. Jika Anda menggunakan OpenCV untuk mengambil gambar untuk dianalisis`DetectAnomalies`, Anda harus mengonversi gambar ke format RGB888 sebelum Anda meneruskan gambar ke. `DetectAnomalies`

Gambar yang Anda berikan `DetectAnomalies` harus memiliki dimensi lebar dan tinggi yang sama dengan gambar yang Anda gunakan untuk melatih model.

Mendeteksi Anomali dengan menggunakan byte gambar

Anda dapat mendeteksi anomali dalam gambar dengan memasok gambar sebagai byte gambar. Dalam contoh berikut, byte gambar diambil dari gambar yang disimpan dalam sistem file lokal.

Ganti *`sample.jpg`* dengan nama file gambar yang ingin Anda analisis. Ganti *`component_name`* dengan nama komponen model Anda.

```
import time

from PIL import Image
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2


model_component_name = "component_name"

....
# Detecting anomalies.
def detect_anomalies(stub, model_name, image_path):
    image = Image.open(image_path)
    image = image.convert("RGB")
    detect_anomalies_response = stub.DetectAnomalies(
        pb2.DetectAnomaliesRequest(
            model_component=model_name,
            bitmap=pb2.Bitmap(
                width=image.size[0],
                height=image.size[1],
                byte_data=bytes(image.tobytes())
            )
        )
    )
    print(f"Image is anomalous -
 {detect_anomalies_response.detect_anomaly_result.is_anomalous}")
    return detect_anomalies_response.detect_anomaly_result


# Creating stub.
```

```
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
    start_model_if_needed(stub, model_component_name)
    detect_anomalies(stub, model_component_name, "sample.jpg")
```

Mendeteksi Anomali dengan menggunakan segmen memori bersama

Anda dapat mendeteksi anomali pada gambar dengan memasok gambar sebagai byte gambar di segmen memori bersama POSIX. Untuk kinerja terbaik, sebaiknya gunakan memori bersama untuk DetectAnomalies permintaan. Untuk informasi selengkapnya, lihat DetectAnomalies.

Menghentikan model

Jika Anda tidak lagi menggunakan model, StopModel API akan menghentikan model berjalan.

```
stop_model_response = stub.StopModel(
    pb2.StopModelRequest(
         model_component=model_component_name
    )
)
print(f"New status of the model is {stop_model_response.status}")
```

Daftar model pada perangkat

Anda dapat menggunakan the section called “ListModels” API untuk membuat daftar model yang diterapkan ke perangkat.

```
models_list_response = stub.ListModels(
    pb2.ListModelsRequest()
)
for model in models_list_response.models:
    print(f"Model Details {model}")
```

Menggambarkan model

Anda bisa mendapatkan informasi tentang model yang diterapkan ke perangkat dengan memanggil DescribeModel API. Menggunakan `DescribeModel` berguna untuk mendapatkan status model saat ini. Misalnya, Anda perlu tahu apakah model berjalan sebelum Anda dapat meneleponDetectAnomalies. Untuk kode sampel, lihat Memulai model.

Mendapatkan informasi kesalahan

Kode status gRPC digunakan untuk melaporkan hasil API.

Anda bisa mendapatkan informasi kesalahan dengan menangkap `RpcError` pengecualian, seperti yang ditunjukkan pada contoh berikut. Untuk informasi tentang kode status kesalahan, lihat topik referensi untuk API.

```
# Error handling.
try:
    stub.DetectAnomalies(detect_anomalies_request)
except grpc.RpcError as e:
    print(f"Error code: {e.code()}, Status: {e.details()}")
```

Membuat komponen aplikasi klien

Anda dapat membuat komponen aplikasi klien setelah Anda membuat rintisan gRPC Anda dan Anda memiliki kode aplikasi klien Anda siap. Komponen yang Anda buat adalah komponen khusus yang Anda gunakan untuk perangkat AWS IoT Greengrass Version 2 inti. AWS IoT Greengrass V2 Resep yang Anda buat menjelaskan komponen kustom Anda. Resepnya mencakup dependensi apa pun yang juga perlu diterapkan. Dalam hal ini, Anda menentukan komponen model yang Anda buatMengemas model Amazon Lookout for Vision Anda. Untuk informasi selengkapnya tentang resep komponen, lihat referensi resep AWS IoT Greengrass Version 2 komponen.

Prosedur pada topik ini menunjukkan cara membuat komponen aplikasi klien dari file resep dan mempublikasikannya sebagai komponen AWS IoT Greengrass V2 khusus. Anda dapat menggunakan AWS IoT Greengrass V2 konsol atau AWS SDK untuk mempublikasikan komponen.

Untuk informasi rinci tentang membuat komponen kustom, lihat berikut ini dalam AWS IoT Greengrass V2dokumentasi.

- Mengembangkan dan menguji komponen pada perangkat Anda
- Buat komponen AWS IoT Greengrass
- Publikasikan komponen untuk diterapkan ke perangkat inti Anda

Topik

- Izin IAM untuk menerbitkan komponen aplikasi klien
- Membuat resep

- Menerbitkan komponen aplikasi klien (Konsol)
- Menerbitkan komponen aplikasi klien (SDK)

Izin IAM untuk menerbitkan komponen aplikasi klien

Untuk membuat dan mempublikasikan komponen aplikasi klien Anda, Anda memerlukan izin IAM berikut:

- `greengrass:CreateComponentVersion`
- `greengrass:DescribeComponent`
- `s3:PutObject`

Membuat resep

Dalam prosedur ini, Anda membuat resep untuk komponen aplikasi klien sederhana. Kode dalam `lookoutvision_edge_agent_example.py` daftar model yang diterapkan ke perangkat dan secara otomatis dijalankan setelah Anda menyebarkan komponen ke perangkat inti. Untuk melihat output, periksa log komponen setelah Anda menerapkan komponen. Untuk informasi selengkapnya, lihat Menerapkan komponen Anda ke perangkat. Ketika Anda siap, gunakan prosedur ini untuk membuat resep untuk kode yang mengimplementasikan logika bisnis Anda.

Anda membuat resep sebagai file format JSON atau YAMB. Anda juga mengunggah kode aplikasi klien ke bucket Amazon S3.

Untuk membuat resep komponen aplikasi klien

1. Jika Anda belum melakukannya, buat file rintisan gRPC. Untuk informasi selengkapnya, lihat Menyiapkan gRPC.
2. Simpan kode berikut ke file bernama `lookoutvision_edge_agent_example.py`

```
import grpc
from edge_agent_pb2_grpc import EdgeAgentStub
import edge_agent_pb2 as pb2

# Creating stub.
with grpc.insecure_channel("unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock") as
 channel:
    stub = EdgeAgentStub(channel)
```

```
    # Add additional code that works with Edge Agent in this block to prevent
resources leakage

    models_list_response = stub.ListModels(
        pb2.ListModelsRequest()
    )
    for model in models_list_response.models:
        print(f"Model Details {model}")
```

3. Buat bucket Amazon S3 (atau gunakan bucket yang sudah ada) untuk menyimpan file sumber komponen aplikasi klien Anda. Bucket harus ada di AWS akun Anda dan di AWS Wilayah yang sama dengan tempat Anda menggunakan AWS IoT Greengrass Version 2 dan Amazon Lookout for Vision.
4. Unggah`lookoutvision_edge_agent_example.py`, `edge_agent_pb2_grpc.py` and `edge_agent_pb2.py` ke bucket Amazon S3 yang Anda buat di langkah sebelumnya. Perhatikan jalur Amazon S3 dari setiap file. Anda menciptakan `edge_agent_pb2_grpc.py` dan `edge_agent_pb2.py` masukMenyiapkan gRPC.
5. Dalam editor membuat file resep JSON atau YAMAL berikut.

   - `model_component`untuk nama komponen model Anda. Untuk informasi selengkapnya, lihat Pengaturan komponen.
   - Ubah entri URI ke jalur S3 dari`lookoutvision_edge_agent_example.py`,`edge_agent_pb2_grpc.py`, and. `edge_agent_pb2.py`

   JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.lookoutvision.EdgeAgentPythonExample",
  "ComponentVersion": "1.0.0",
  "ComponentType": "aws.greengrass.generic",
  "ComponentDescription": "Lookout for Vision Edge Agent Sample Application",
  "ComponentPublisher": "Sample App Publisher",
  "ComponentDependencies": {
    "model_component": {
      "VersionRequirement": ">=1.0.0",
      "DependencyType": "HARD"
    }
  },
```

```
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "pip3 install grpcio grpcio-tools protobuf Pillow",
        "run": {
          "script": "python3 {artifacts:path}/
lookoutvision_edge_agent_example.py"
        }
      },
      "Artifacts": [
        {
          "Uri": "S3 path to lookoutvision_edge_agent_example.py"
        },
        {
          "Uri": "S3 path to edge_agent_pb2_grpc.py"
        },
        {
          "Uri": "S3 path to edge_agent_pb2.py"
        }
      ]
    }
  ],
  "Lifecycle": {}
}
```

YAML

```
---
RecipeFormatVersion: 2020-01-25
ComponentName: com.lookoutvison.EdgeAgentPythonExample
ComponentVersion: 1.0.0
ComponentDescription: Lookout for Vision Edge Agent Sample Application
ComponentPublisher: Sample App Publisher
ComponentDependencies:
  model_component:
    VersionRequirement: '>=1.0.0'
    DependencyType: HARD
Manifests:
  - Platform:
      os: linux
```

```
    Lifecycle:
      install: |-
        pip3 install grpcio
        pip3 install grpcio-tools
        pip3 install protobuf
        pip3 install Pillow
      run:
        script: |-
          python3 {artifacts:path}/lookout_vision_agent_example.py
    Artifacts:
      - URI: S3 path to lookoutvision_edge_agent_example.py
      - URI: S3 path to edge_agent_pb2_grpc.py
      - URI: S3 path to edge_agent_pb2.py
```

6. Simpan file JSON atau YAMB ke komputer Anda.
7. Buat komponen aplikasi klien dengan melakukan salah satu hal berikut:
   - Jika Anda ingin menggunakan AWS IoT Greengrass konsol, lakukanMenerbitkan komponen aplikasi klien (Konsol).
   - Jika Anda ingin menggunakan AWS SDK, lakukanMenerbitkan komponen aplikasi klien (SDK).

Menerbitkan komponen aplikasi klien (Konsol)

Anda dapat menggunakan AWS IoT Greengrass V2 konsol untuk mempublikasikan komponen aplikasi klien.

Untuk mempublikasikan komponen aplikasi klien

1. Jika Anda belum melakukannya, buat resep untuk komponen aplikasi klien Anda dengan melakukan. Membuat resep
2. Buka AWS IoT Greengrass konsol di https://console.aws.amazon.com/iot/
3. Di panel navigasi kiri, di bawah Greengrass pilih Komponen.
4. Di bawah Komponen saya pilih Buat komponen.
5. Pada halaman Create component pilih Enter recipe as JSON jika Anda ingin menggunakan resep format JSON. Pilih Masukkan resep sebagai YAMAL jika Anda ingin menggunakan resep format YAMAL.
6. Di bawah Resep ganti resep yang ada dengan resep JSON atau YAMAL yang Anda buat. Membuat resep

7. Pilih Buat komponen.
8. Selanjutnya, terapkan komponen aplikasi klien Anda.

Menerbitkan komponen aplikasi klien (SDK)

Anda dapat mempublikasikan komponen aplikasi klien dengan menggunakan CreateComponentVersionAPI.

Untuk mempublikasikan komponen aplikasi klien (SDK)

1. Jika Anda belum melakukannya, buat resep untuk komponen aplikasi klien Anda dengan melakukan. Membuat resep
2. Pada prompt perintah, masukkan perintah berikut untuk membuat komponen aplikasi klien. Ganti `recipe-file` dengan nama file resep yang Anda buatMembuat resep.

   ```
   aws greengrassv2 create-component-version --inline-recipe fileb://recipe-file
   ```

   Perhatikan ARN komponen dalam respons. Anda membutuhkannya di langkah berikutnya.
3. Gunakan perintah berikut untuk mendapatkan status komponen aplikasi klien. Ganti `component-arn` dengan ARN yang Anda catat di langkah sebelumnya. Komponen aplikasi klien siap jika nilainya `componentState` adalah`DEPLOYABLE`.

   ```
   aws greengrassv2 describe-component --arn component-arn
   ```
4. Selanjutnya, terapkan komponen aplikasi klien Anda.

Menerapkan komponen Anda ke perangkat

Untuk menerapkan komponen model dan komponen aplikasi klien ke perangkat AWS IoT Greengrass Version 2 inti, Anda menggunakan AWS IoT Greengrass V2 konsol atau menggunakan CreateDeploymentAPI. Untuk informasi selengkapnya, lihat Membuat penerapan atau di Panduan AWS IoT Greengrass Version 2Pengembang. Untuk informasi tentang memperbarui komponen yang diterapkan ke perangkat inti, lihat Merevisi penerapan.

Topik

- Izin IAM untuk menyebarkan komponen
- Menerapkan komponen Anda (konsol)

- Menerapkan komponen (SDK)

Izin IAM untuk menyebarkan komponen

Untuk menerapkan komponen dengan AWS IoT Greengrass V2 Anda memerlukan izin berikut:

- `greengrass:ListComponents`
- `greengrass:ListComponentVersions`
- `greengrass:ListCoreDevices`
- `greengrass:CreateDeployment`
- `greengrass:GetDeployment`
- `greengrass:ListDeployments`

`CreateDeployment`dan `GetDeployment` memiliki tindakan yang bergantung. Untuk informasi selengkapnya, lihat Tindakan yang ditentukan oleh AWS IoT Greengrass V2.

Untuk informasi tentang mengubah izin IAM, lihat Mengubah izin untuk pengguna.

Menerapkan komponen Anda (konsol)

Gunakan prosedur berikut untuk menyebarkan komponen aplikasi klien ke perangkat inti. Aplikasi klien bergantung pada komponen model (yang pada gilirannya tergantung pada Lookout for Vision Edge Agent). Menyebarkan komponen aplikasi klien juga memulai penyebaran komponen model dan Lookout for Vision Edge Agent.

> **Note**
>
> Anda dapat menambahkan komponen Anda ke penerapan yang ada. Anda juga dapat menerapkan komponen ke grup sesuatu.

Untuk menjalankan prosedur ini, Anda harus memiliki perangkat AWS IoT Greengrass V2 inti yang dikonfigurasi. Untuk informasi selengkapnya, lihat Menyiapkan perangkat AWS IoT Greengrass Version 2 inti Anda.

Untuk menyebarkan komponen Anda ke perangkat

1. Buka AWS IoT Greengrass konsol di https://console.aws.amazon.com/iot/.

2. Di panel navigasi kiri, di bawah Greengrass pilih Deployments.
3. Di bawah Deployment pilih Buat.
4. Di halaman Tentukan target, lakukan hal berikut:
    1. Di bawah informasi Deployment, masukkan atau ubah nama yang ramah untuk deployment Anda.
    2. Di bawah Target Deployment, pilih Perangkat inti dan masukkan nama target.
    3. Pilih Selanjutnya.
5. Pada halaman Pilih komponen, lakukan hal berikut:
    1. Di bawah Komponen saya, pilih nama komponen aplikasi klien Anda (`com.lookoutvison.EdgeAgentPythonExample`).
    2. Pilih Selanjutnya
6. Pada halaman Konfigurasi komponen, pertahankan konfigurasi saat ini dan pilih Berikutnya.
7. Pada halaman Konfigurasi pengaturan lanjutan, pertahankan pengaturan saat ini dan pilih Berikutnya.
8. Pada halaman Tinjauan, pilih Deploy untuk mulai menerapkan komponen Anda.

Memeriksa status penerapan (Konsol)

Anda dapat memeriksa status penerapan Anda dari AWS IoT Greengrass V2 konsol. Jika komponen aplikasi klien Anda menggunakan contoh resep dan kode darithe section called “Membuat komponen aplikasi klien”, lihat log komponen aplikasi klien setelah penerapan selesai. Jika berhasil, log menyertakan daftar model Lookout for Vision yang disebarkan ke komponen.

Untuk informasi tentang menggunakan AWS SDK untuk memeriksa status penerapan, lihat Memeriksa status penerapan.

Untuk memeriksa status penerapan

1. Buka AWS IoT Greengrass konsol di https://console.aws.amazon.com/iot/
2. Di panel navigasi kiri, pilih Perangkat inti.
3. Di bawah perangkat inti Greengrass pilih perangkat inti Anda.
4. Pilih tab Deployment untuk melihat status penerapan saat ini.
5. Setelah penerapan berhasil (status Selesai), buka jendela terminal pada perangkat inti dan lihat log komponen aplikasi klien di. `/greengrass/`

`v2/logs/com.lookoutvison.EdgeAgentPythonExample.log` Jika penerapan Anda menggunakan contoh resep dan kode, log menyertakan output dari`lookoutvision_edge_agent_example.py`. Misalnya:

```
Model Details model_component:"ModelComponent"
```

Menerapkan komponen (SDK)

Gunakan prosedur berikut untuk menerapkan komponen aplikasi klien, komponen model, dan Amazon Lookout for Vision Edge Agent ke perangkat inti Anda.

1. Buat `deployment.json` file untuk menentukan konfigurasi deployment untuk komponen Anda. File ini akan terlihat seperti contoh berikut.

```
{
  "targetArn":"targetArn",
  "components": {
    "com.lookoutvison.EdgeAgentPythonExample": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
      }
    }
  }
}
```

   - Di kolom `targetArn`, ganti *`targetArn`* dengan Amazon Resource Name (ARN) dari grup objek atau objek yang ditargetkan untuk deployment tersebut, dalam format berikut:
     - Objek: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Grup objek: `arn:aws:iot:`*`region`*`:`*`account-id:`*`thinggroup/`*`thingGroupName`*
2. Periksa apakah target penerapan memiliki penerapan yang sudah ada yang ingin Anda revisi. Lakukan hal berikut:

   a. Jalankan perintah berikut untuk membuat daftar penerapan untuk target penyebaran. Ganti `targetArn` dengan Amazon Resource Name (ARN) dari grup hal atau benda AWS IoT target. Untuk mendapatkan ARN dari hal-hal di Wilayah AWS saat ini, gunakan perintah`aws iot list-things`.

```
aws greengrassv2 list-deployments --target-arn targetArn
```

Tanggapan berisi daftar dengan deployment terbaru untuk target. Jika responsnya kosong, target tidak memiliki penerapan yang ada, dan Anda dapat melompat ke Langkah 3. Jika tidak, salin `deploymentId` dari respons untuk digunakan pada langkah berikutnya.

b. Jalankan perintah berikut untuk mendapatkan detail penerapan. Detail ini mencakup metadata, komponen, dan konfigurasi pekerjaan. Ganti `deploymentId` dengan ID dari langkah sebelumnya.

```
aws greengrassv2 get-deployment --deployment-id deploymentId
```

c. Salin salah satu pasangan nilai kunci berikut dari respons perintah sebelumnya ke deployment.json. Anda dapat mengubah nilai-nilai ini untuk penerapan baru.

- `deploymentName`— Nama penyebaran.
- `components`— Komponen penyebaran. Untuk menghapus komponen, hapus dari objek ini.
- `deploymentPolicies`— Kebijakan penyebaran.
- `tags`— Tag penyebaran.

3. Jalankan perintah berikut untuk menyebarkan komponen pada perangkat. Perhatikan nilai `deploymentId` dalam respon.

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

4. Jalankan perintah berikut untuk mendapatkan status penyebaran. Ubah `deployment-id` ke nilai yang Anda catat di langkah sebelumnya. Penerapan telah selesai dengan sukses jika nilainya adalah. `deploymentStatus` `COMPLETED`

```
aws greengrassv2 get-deployment --deployment-id deployment-id
```

5. Setelah penerapan berhasil, buka jendela terminal pada perangkat inti dan lihat log komponen aplikasi klien di. `/greengrass/v2/logs/com.lookoutvison.EdgeAgentPythonExample.log` Jika penerapan Anda menggunakan contoh resep dan kode, log menyertakan output dari`lookoutvision_edge_agent_example.py`. Misalnya:

```
Model Details model_component:"ModelComponent"
```

Referensi API Amazon Lookout for Vision Edge Agent

Bagian ini adalah referensi API untuk Amazon Lookout for Vision Edge Agent.

Mendeteksi anomali dengan model

Anda menggunakan DetectAnomalies API untuk mendeteksi anomali pada gambar dengan menggunakan model yang sedang berjalan pada perangkat AWS IoT Greengrass Version 2 inti.

Mendapatkan informasi model

API yang mendapatkan informasi tentang model yang diterapkan ke perangkat inti.

- ListModels
- DescribeModel

Menjalankan model

API untuk memulai dan menghentikan model Amazon Lookout for Vision yang diterapkan ke perangkat inti.

- StartModel
- StopModel

DetectAnomalies

Mendeteksi anomali pada gambar yang disediakan.

Respons dari `DetectAnomalies` mencakup prediksi Boolean bahwa gambar berisi satu atau lebih anomali dan nilai kepercayaan untuk prediksi. Jika model adalah model segmentasi, responsnya meliputi yang berikut:

- Gambar topeng yang menutupi setiap jenis anomali dalam warna yang unik. Anda dapat `DetectAnomalies` menyimpan gambar topeng dalam memori bersama, atau mengembalikan topeng sebagai byte gambar.

- Persentase area gambar yang dicakup oleh tipe anomali.
- Warna hex untuk jenis anomali pada gambar topeng.

> **Note**
>
> Model yang Anda gunakan `DetectAnomalies` harus berjalan. Anda bisa mendapatkan status terkini dengan memanggil DescribeModel. Untuk mulai menjalankan model, lihatStartModel.

`DetectAnomalies`mendukung bitmap yang dikemas (gambar) dalam format RGB888 yang disisipkan. Byte pertama mewakili saluran merah, byte kedua mewakili saluran hijau, dan byte ketiga mewakili saluran biru. Jika Anda memberikan gambar dalam format yang berbeda, seperti BGR, prediksi dari salah DetectAnomalies .

Secara default, OpenCV menggunakan format BGR untuk bitmap gambar. Jika Anda menggunakan OpenCV untuk mengambil gambar untuk dianalisis`DetectAnomalies`, Anda harus mengonversi gambar ke format RGB888 sebelum Anda meneruskan gambar ke. `DetectAnomalies`

Dimensi gambar minimum yang didukung adalah 64x64 piksel. Dimensi gambar maksimum yang didukung adalah 4096x4096 piksel.

Anda dapat mengirim gambar dalam pesan protobuf atau melalui segmen memori bersama. Serialisasi gambar besar ke dalam pesan protobuf dapat secara signifikan meningkatkan latensi panggilan ke. `DetectAnomalies` Untuk latensi paling sedikit, kami menyarankan Anda menggunakan memori bersama.

```
rpc DetectAnomalies(DetectAnomaliesRequest) returns (DetectAnomaliesResponse);
```

DetectAnomaliesRequest

Parameter input untuk`DetectAnomalies`.

```
message Bitmap {
  int32 width = 1;
  int32 height = 2;
  oneof data {
```

```
    bytes byte_data = 3;
    SharedMemoryHandle shared_memory_handle = 4;
  }
}
```

```
message SharedMemoryHandle {
  string name = 1;
  uint64 size = 2;
  uint64 offset = 3;
}
```

```
message AnomalyMaskParams {
SharedMemoryHandle shared_memory_handle = 2;
}
```

```
message DetectAnomaliesRequest {
string model_component = 1;
Bitmap bitmap = 2;
AnomalyMaskParams anomaly_mask_params = 3;
}
```

Bitmap

Gambar yang ingin Anda analisis`DetectAnomalies`.

lebar

Lebar gambar dalam piksel.

tingginya

Ketinggian gambar dalam piksel.

byte_data

Byte gambar diteruskan dalam pesan protobuf.

shared_memory_handle

Byte gambar diteruskan di segmen memori bersama.

SharedMemoryHandle

Merupakan segmen memori bersama POSIX.

nama

Nama segmen memori POSIX. Untuk informasi tentang membuat memori bersama, lihat shm_open.

size

Ukuran buffer gambar dalam byte mulai dari offset.

mengimbangi

Offset, dalam byte, ke awal buffer gambar dari awal segmen memori bersama.

AnomalyMaskParams

Parameter untuk mengeluarkan topeng anomali. (Model segmentasi).

shared_memory_handle

Berisi byte gambar untuk topeng, jika `shared_memory_handle` tidak disediakan.

DetectAnomaliesRequest

model_component

Nama AWS IoT Greengrass V2 komponen yang berisi model yang ingin Anda gunakan.

bitmap

Gambar yang ingin Anda analisis dengan`DetectAnomalies`.

anomaly_mask_params

Parameter opsional untuk mengeluarkan topeng. (Model segmentasi).

DetectAnomaliesResponse

Tanggapan dari`DetectAnomalies`.

```
message DetectAnomalyResult {
bool is_anomalous = 1;
```

```
float confidence = 2;
Bitmap anomaly_mask = 3;
repeated Anomaly anomalies = 4;
float anomaly_score = 5;
float anomaly_threshold = 6;
}
```

```
message Anomaly {
string name = 1;
PixelAnomaly pixel_anomaly = 2;
```

```
message PixelAnomaly {
float total_percentage_area = 1;
string hex_color = 2;
}
```

```
message DetectAnomaliesResponse {
  DetectAnomalyResult detect_anomaly_result = 1;
}
```

Anomali

Merupakan anomali yang ditemukan pada gambar. (Model segmentasi).

nama

Nama jenis anomali yang ditemukan dalam gambar. `name`mememetakan ke jenis anomali dalam dataset pelatihan. Layanan secara otomatis memasukkan jenis anomali latar belakang ke dalam respons dari. DetectAnomalies

pixel_anomali

Informasi tentang topeng piksel yang mencakup jenis anomali.

PixelAnomaly

Informasi tentang topeng piksel yang mencakup jenis anomali. (Model segmentasi).

total_percentage_area

Persentase area gambar yang dicakup oleh tipe anomali.

hex_color

Nilai warna hex yang mewakili jenis anomali pada gambar. Warna memetakan warna jenis anomali yang digunakan dalam dataset pelatihan.

DetectAnomalyResult

is_anomali

Menunjukkan apakah gambar mengandung anomali. `true`jika gambar mengandung anomali. `false`jika gambarnya normal.

kepercayaan

Keyakinan yang `DetectAnomalies` ada pada keakuratan prediksi. `confidence`adalah nilai floating point antara 0 dan 1.

anomaly_mask

jika shared_memory_handle tidak disediakan, berisi byte gambar untuk topeng. (Model segmentasi).

anomali

Daftar 0 atau lebih anomali yang ditemukan dalam gambar input. (Model segmentasi).

anomaly_score

Angka yang mengukur berapa banyak anomali yang diprediksi untuk gambar menyimpang dari gambar tanpa anomali. `anomaly_score`adalah nilai float mulai dari `0.0` ke (deviasi terendah dari gambar normal) hingga 1,0 (deviasi tertinggi dari gambar normal). Amazon Lookout for Vision mengembalikan nilai `anomaly_score` untuk, meskipun prediksi untuk gambar normal.

anomaly_threshold

Angka (float) yang menentukan kapan klasifikasi yang diprediksi untuk suatu gambar normal atau anomali. Gambar dengan nilai `anomaly_score` yang sama dengan atau di atas nilai `anomaly_threshold` dianggap anomali. `anomaly_score`Nilai yang ada di bawah `anomaly_threshold` menunjukkan gambar normal. Nilai `anomaly_threshold` yang digunakan model dihitung oleh Amazon Lookout for Vision saat Anda melatih model. Anda tidak dapat mengatur atau mengubah nilai `anomaly_threshold`

Kode status

Code	Jumlah	Deskripsi
OKE	0	`DetectAnomalies` Berhasil Membuat Prediksi
TIDAK DIKETAHUI	2	Terjadi kesalahan yang tidak diketahui.
ARGUMEN INVALID_	3	Satu atau lebih parameter input tidak valid. Periksa pesan kesalahan untuk lebih jelasnya.
TIDAK_DITEMUKAN	5	Model dengan nama yang ditentukan tidak ditemukan.
RESOURCE_EXHAUSTED	8	Tidak ada sumber daya yang cukup untuk melakukan operasi ini. Misalnya, Agen Lookout for Vision Edge tidak dapat mengikuti tingkat panggilan ke. `DetectAnomalies` Periksa pesan kesalahan untuk lebih jelasnya.
FAILED_PRECONDITION	9	`DetectAnomalies` dipanggil untuk model yang tidak dalam status RUNNING.
BATIN	13	Terjadi kesalahan internal.

DescribeModel

Menjelaskan model Amazon Lookout for Vision yang diterapkan ke perangkat inti. AWS IoT Greengrass Version 2

```
rpc DescribeModel(DescribeModelRequest) returns (DescribeModelResponse);
```

DescribeModelRequest

```
message DescribeModelRequest {
  string model_component = 1;
}
```

model_component

Nama AWS IoT Greengrass V2 komponen yang berisi model yang ingin Anda gambarkan.

DescribeModelResponse

```
message ModelDescription {
  string model_component = 1;
  string lookout_vision_model_arn = 2;
  ModelStatus status = 3;
  string status_message = 4;
}
```

```
message DescribeModelResponse {
  ModelDescription model_description = 1;
}
```

ModelDescription

model_component

Nama AWS IoT Greengrass Version 2 komponen yang berisi model Amazon Lookout for Vision.

lookout_vision_model_arn

Nama Sumber Daya Amazon ARN dari model Amazon Lookout for Vision yang digunakan untuk menghasilkan komponen. AWS IoT Greengrass V2

status

Status model saat ini. Untuk informasi selengkapnya, lihat ModelStatus.

status_message

Pesan status untuk model.

Kode status

Code	Jumlah	Deskripsi
OKE	0	Panggilan itu berhasil.
TIDAK DIKETAHUI	2	Terjadi kesalahan yang tidak diketahui.
ARGUMEN INVALID_	3	Satu atau lebih parameter input tidak valid. Periksa pesan kesalahan untuk lebih jelasnya.
TIDAK_DITEMUKAN	5	Model dengan nama yang disediakan tidak ditemukan.
BATIN	13	Terjadi kesalahan internal.

ListModels

Daftar model yang digunakan ke perangkat AWS IoT Greengrass Version 2 inti.

```
rpc ListModels(ListModelsRequest) returns (ListModelsResponse);
```

ListModelsRequest

```
message ListModelsRequest {}
```

ListModelsResponse

```
message ModelMetadata {
  string model_component = 1;
  string lookout_vision_model_arn = 2;
  ModelStatus status = 3;
  string status_message = 4;
}
```

```
message ListModelsResponse {
  repeated ModelMetadata models = 1;
}
```

ModelMetadata

model_component

Nama AWS IoT Greengrass Version 2 komponen yang berisi model Amazon Lookout for Vision.

lookout_vision_model_arn

Nama Sumber Daya Amazon (ARN) dari model Amazon Lookout for Vision yang digunakan untuk menghasilkan komponen. AWS IoT Greengrass V2

status

Status model saat ini. Untuk informasi selengkapnya, lihat ModelStatus.

status_message

Pesan status untuk model.

Kode status

Code	Jumlah	Deskripsi
OKE	0	Panggilan itu berhasil.
TIDAK DIKETAHUI	2	Terjadi kesalahan yang tidak diketahui.
BATIN	13	Terjadi kesalahan internal.

StartModel

Memulai model yang berjalan pada perangkat AWS IoT Greengrass Version 2 inti. Mungkin perlu beberapa saat bagi model untuk mulai berjalan. Untuk memeriksa panggilan status saat iniDescribeModel. Model berjalan jika `Status` bidangnya`RUNNING`.

Jumlah model yang dapat Anda jalankan secara bersamaan tergantung pada spesifikasi perangkat keras perangkat inti Anda.

```
rpc StartModel(StartModelRequest) returns (StartModelResponse);
```

StartModelRequest

```
message StartModelRequest {
  string model_component = 1;
}
```

model_component

Nama AWS IoT Greengrass Version 2 komponen yang berisi model yang ingin Anda mulai.

StartModelResponse

```
message StartModelResponse {
  ModelStatus status = 1;
}
```

status

Status model saat ini. Jawabannya adalah `STARTING` jika panggilan berhasil. Untuk informasi selengkapnya, lihat ModelStatus.

Kode status

Code	Jumlah	Deskripsi
OKE	0	Modelnya dimulai

Code	Jumlah	Deskripsi
TIDAK DIKETAHUI	2	Terjadi kesalahan yang tidak diketahui.
ARGUMEN INVALID_	3	Satu atau lebih parameter input tidak valid. Periksa pesan kesalahan untuk lebih jelasnya.
TIDAK_DITEMUKAN	5	Model dengan nama yang disediakan tidak ditemukan.
RESOURCE_EXHAUSTED	8	Tidak ada sumber daya yang cukup untuk melakukan operasi ini. Misalnya, tidak ada cukup memori untuk memuat model. Periksa pesan kesalahan untuk lebih jelasnya.
FAILED_PRECONDITION	9	Metode ini dipanggil untuk model yang tidak dalam status STOPTED atau FAILED.
BATIN	13	Terjadi kesalahan internal.

StopModel

Menghentikan model yang berjalan pada perangkat AWS IoT Greengrass Version 2 inti. `StopModel`kembali setelah model berhenti. Model telah berhenti dengan sukses jika `Status` bidang dalam respons adalah`STOPPED`.

```
  rpc StopModel(StopModelRequest) returns (StopModelResponse);
```

StopModelRequest

```
message StopModelRequest {
  string model_component = 1;
}
```

model_component

Nama AWS IoT Greengrass Version 2 komponen yang berisi model yang ingin Anda hentikan.

StopModelResponse

```
message StopModelResponse {
  ModelStatus status = 1;
}
```

status

Status model saat ini. Jawabannya adalah `STOPPED` jika panggilan berhasil. Untuk informasi selengkapnya, lihat ModelStatus.

Kode status

Code	Jumlah	Deskripsi
OKE	0	Modelnya berhenti.
TIDAK DIKETAHUI	2	Terjadi kesalahan yang tidak diketahui.
ARGUMEN INVALID_	3	Satu atau lebih parameter input tidak valid. Periksa pesan kesalahan untuk lebih jelasnya.
TIDAK_DITEMUKAN	5	Model dengan nama yang disediakan tidak ditemukan.
FAILED_PRECONDITION	9	Metode ini dipanggil untuk model yang tidak dalam status RUNNING.

Code	Jumlah	Deskripsi
BATIN	13	Terjadi kesalahan internal.

ModelStatus

Status model yang dikerahkan ke perangkat AWS IoT Greengrass Version 2 inti. Untuk mendapatkan status saat ini, hubungiDescribeModel.

```
enum ModelStatus {
  STOPPED = 0;
  STARTING = 1;
  RUNNING = 2;
  FAILED = 3;
  STOPPING = 4;
}
```

Menggunakan dasbor Amazon Lookout for Vision

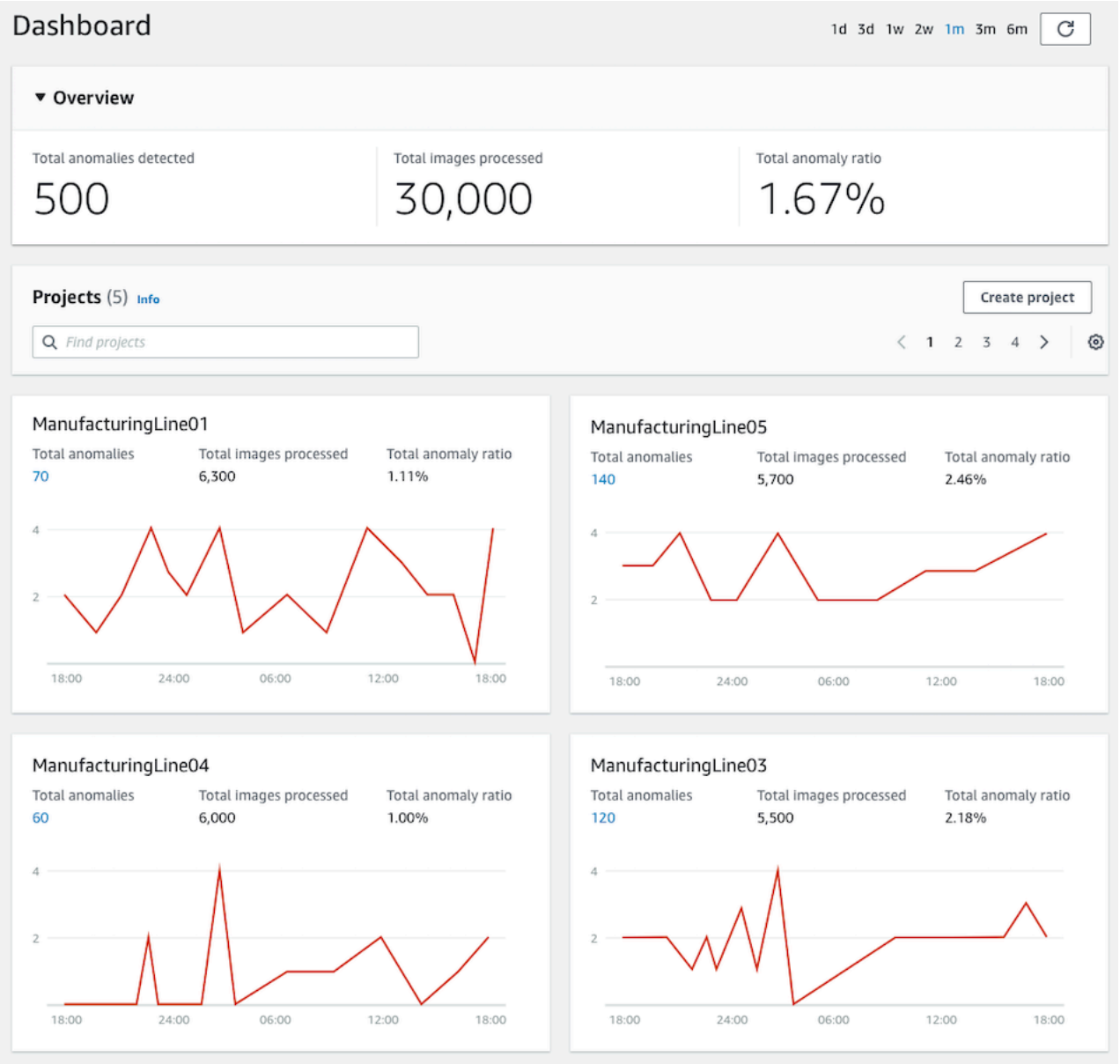
Dasbor menyediakan ikhtisar metrik untuk proyek Amazon Lookout for Vision Anda, seperti jumlah total anomali yang terdeteksi selama seminggu terakhir. Dengan dasbor Anda mendapatkan ikhtisar untuk semua proyek Anda dan ikhtisar untuk setiap proyek individu. Anda dapat memilih timeline di mana metrik ditampilkan. Anda juga dapat menggunakan dasbor untuk membuat proyek baru.

Bagian Ikhtisar menunjukkan jumlah total proyek, jumlah total gambar, dan jumlah total gambar yang terdeteksi oleh semua proyek Anda.

Bagian Proyek menunjukkan informasi ikhtisar berikut untuk masing-masing proyek:

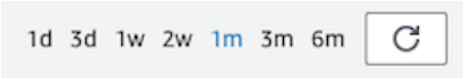
- Jumlah total atau anomali terdeteksi.
- Jumlah total gambar yang diproses.
- Rasio anomali total (yaitu persentase gambar yang terdeteksi dengan anomali).
- Grafik menunjukkan deteksi anomali selama kerangka waktu yang dipilih.

Anda juga bisa mendapatkan informasi lebih lanjut tentang proyek.

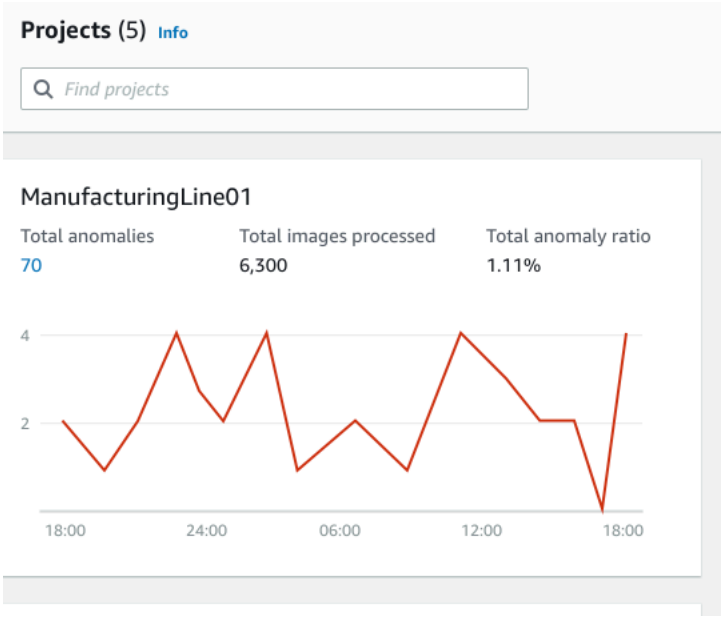


Menggunakan dasbor Anda

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi sebelah kiri, pilih Dasbor.
4. Untuk melihat metrik dalam jangka waktu tertentu, lakukan hal berikut:
    a. Pilih kerangka waktu di sisi kanan atas dasbor.
    b. Pilih tombol refresh untuk menampilkan dasbor dengan garis waktu baru.



5. Untuk mendapatkan rincian lebih lanjut tentang proyek, pilih nama proyek di bagian Proyek (misalnya,`ManufacturingLine01`).



6. Untuk membuat proyek, pilih Create project di bagian Projects.

Mengelola sumber daya Amazon Lookout for Vision

Anda dapat mengelola sumber daya Amazon Lookout for Vision menggunakan konsol atau SDK. AWS Amazon Lookout for Vision memiliki sumber daya berikut:

- Proyek
- Set Data
- Model
- Deteksi percobaan

> **Note**
>
> Anda tidak dapat menghapus tugas deteksi uji coba. Selain itu, Anda tidak dapat mengelola deteksi uji coba dengan menggunakan AWS SDK.

Topik

Melihat proyek Anda

Anda bisa mendapatkan daftar proyek Amazon Lookout for Vision dan informasi tentang proyek individual dari konsol atau dengan menggunakan SDK. AWS

> **Note**
>
> Daftar proyek pada akhirnya konsisten. Jika Anda membuat atau menghapus proyek, Anda mungkin harus menunggu beberapa saat sebelum daftar proyek diperbarui.

Melihat proyek Anda (konsol)

Lakukan langkah-langkah dalam prosedur berikut untuk melihat proyek Anda di konsol.

Untuk melihat proyek Anda

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi kiri, pilih Proyek. Tampilan proyek ditampilkan.
4. Pilih nama proyek untuk melihat detail proyek.

Melihat proyek Anda (SDK)

Sebuah proyek mengelola kumpulan data dan model untuk satu kasus penggunaan. Misalnya, mendeteksi anomali pada bagian-bagian mesin. Contoh berikut memanggil `ListProjects` untuk mendapatkan daftar proyek Anda.

Untuk melihat proyek Anda (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode contoh berikut untuk melihat proyek Anda.

   CLI

   Gunakan `list-projects` perintah untuk membuat daftar proyek di akun Anda.

   ```
   aws lookoutvision list-projects \
     --profile lookoutvision-access
   ```

   Gunakan `describe-project` perintah untuk mendapatkan informasi tentang proyek.

Ubah nilai `project-name` menjadi nama proyek yang ingin Anda gambarkan.

```
aws lookoutvision describe-project --project-name project_name \
  --profile lookoutvision-access
```

Python

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
    @staticmethod
    def list_projects(lookoutvision_client):
        """
        Lists information about the projects that are in in your AWS account
        and in the current AWS Region.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        """
        try:
            response = lookoutvision_client.list_projects()
            for project in response["Projects"]:
                print("Project: " + project["ProjectName"])
                print("\tARN: " + project["ProjectArn"])
                print("\tCreated: " + str(["CreationTimestamp"]))
                print("Datasets")
                project_description = lookoutvision_client.describe_project(
                    ProjectName=project["ProjectName"]
                )
                if not project_description["ProjectDescription"]["Datasets"]:
                    print("\tNo datasets")
                else:
                    for dataset in project_description["ProjectDescription"][
                        "Datasets"
                    ]:
                        print(f"\ttype: {dataset['DatasetType']}")
                        print(f"\tStatus: {dataset['StatusMessage']}")

                print("Models")
                response_models = lookoutvision_client.list_models(
                    ProjectName=project["ProjectName"]
                )
                if not response_models["Models"]:
```

```
                    print("\tNo models")
                else:
                    for model in response_models["Models"]:
                        Models.describe_model(
                            lookoutvision_client,
                            project["ProjectName"],
                            model["ModelVersion"],
                        )


 print("-----------------------------------------------------------\n")
            print("Done!")
        except ClientError:
            logger.exception("Problem listing projects.")
            raise
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
/**
 * Lists the Amazon Lookout for Vision projects in the current AWS account and
 AWS
 * Region.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return List<ProjectMetadata> Metadata for each project.
 */
public static List<ProjectMetadata> listProjects(LookoutVisionClient lfvClient)
                throws LookoutVisionException {

        logger.log(Level.INFO, "Getting projects:");
        ListProjectsRequest listProjectsRequest = ListProjectsRequest.builder()
                        .maxResults(100)
                        .build();

        List<ProjectMetadata> projectMetadata = new ArrayList<>();
```

```
        ListProjectsIterable projects =
 lfvClient.listProjectsPaginator(listProjectsRequest);

        projects.stream().flatMap(r -> r.projects().stream())
                        .forEach(project -> {
                                projectMetadata.add(project);
                                logger.log(Level.INFO, project.projectName());
                        });

        logger.log(Level.INFO, "Finished getting projects.");

        return projectMetadata;

}
```

Menghapus proyek

Anda dapat menghapus proyek dari halaman tampilan proyek di konsol atau dengan menggunakan `DeleteProject` operasi.

Gambar yang direferensikan oleh kumpulan data proyek tidak dihapus.

Menghapus proyek (konsol)

Gunakan prosedur berikut untuk menghapus proyek. Jika Anda menggunakan prosedur konsol, versi model terkait dan kumpulan data akan dihapus untuk Anda.

Untuk menghapus proyek

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi kiri, pilih Proyek.
4. Pada halaman Proyek, pilih proyek yang ingin Anda hapus.
5. Pilih Hapus di bagian atas halaman.
6. Di kotak dialog Hapus, masukkan hapus untuk mengonfirmasi bahwa Anda ingin menghapus proyek.
7. Jika perlu, pilih untuk menghapus kumpulan data dan model terkait.
8. Pilih Hapus proyek.

Menghapus proyek (SDK)

Anda menghapus proyek Amazon Lookout for Vision dengan DeleteProjectmemanggil dan memberikan nama proyek yang ingin Anda hapus.

Sebelum Anda dapat menghapus proyek, Anda harus terlebih dahulu menghapus semua model dalam proyek. Untuk informasi selengkapnya, lihat Menghapus model (SDK). Anda juga harus menghapus kumpulan data yang terkait dengan model. Untuk informasi selengkapnya, lihat Menghapus dataset.

Proyek ini mungkin membutuhkan beberapa saat untuk dihapus. Selama waktu itu, status proyek adalah`DELETING`. Proyek akan dihapus jika panggilan berikutnya `DeleteProject` tidak menyertakan proyek yang Anda hapus.

Untuk menghapus proyek (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode berikut untuk menghapus proyek.

   AWS CLI

   Ubah nilai `project-name` menjadi nama proyek yang ingin Anda hapus.

   ```
   aws lookoutvision delete-project --project-name project_name \
     --profile lookoutvision-access
   ```

   Python

   Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

   ```
       @staticmethod
       def delete_project(lookoutvision_client, project_name):
           """
           Deletes a Lookout for Vision Model

           :param lookoutvision_client: A Boto3 Lookout for Vision client.
           :param project_name: The name of the project that you want to delete.
           """
   ```

```
        try:
            logger.info("Deleting project: %s", project_name)
            response =
 lookoutvision_client.delete_project(ProjectName=project_name)
            logger.info("Deleted project ARN: %s ", response["ProjectArn"])
        except ClientError as err:
            logger.exception("Couldn't delete project %s.", project_name)
            raise
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
/**
 * Deletes an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that you want to create.
 * @return String The ARN of the deleted project.
 */
public static String deleteProject(LookoutVisionClient lfvClient, String
 projectName)
                throws LookoutVisionException {

        logger.log(Level.INFO, "Deleting project: {0}", projectName);

        DeleteProjectRequest deleteProjectRequest =
 DeleteProjectRequest.builder()
                        .projectName(projectName)
                        .build();

        DeleteProjectResponse response =
 lfvClient.deleteProject(deleteProjectRequest);

        logger.log(Level.INFO, "Deleted project: {0} ARN: {1}",
                        new Object[] { projectName, response.projectArn() });

        return response.projectArn();

}
```

Melihat kumpulan data Anda

Sebuah proyek dapat memiliki satu set data yang digunakan untuk melatih dan menguji model Anda. Atau, Anda dapat memiliki kumpulan data pelatihan dan pengujian terpisah. Anda dapat menggunakan konsol untuk melihat kumpulan data Anda. Anda juga dapat menggunakan `DescribeDataset` operasi untuk mendapatkan informasi tentang dataset (pelatihan atau tes).

Melihat kumpulan data dalam proyek (konsol)

Lakukan langkah-langkah dalam prosedur berikut untuk melihat kumpulan data proyek Anda di konsol.

Untuk melihat kumpulan data Anda (konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi kiri, pilih Proyek.
4. Pada halaman Proyek, pilih proyek yang berisi kumpulan data yang ingin Anda lihat.
5. Di panel navigasi kiri, pilih Dataset untuk melihat detail kumpulan data. Jika Anda memiliki pelatihan dan kumpulan data pengujian, tab untuk setiap kumpulan data ditampilkan.

Melihat kumpulan data dalam proyek (SDK)

Anda dapat menggunakan `DescribeDataset` operasi untuk mendapatkan informasi tentang kumpulan data pelatihan atau pengujian yang terkait dengan proyek.

Untuk melihat kumpulan data (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode contoh berikut untuk melihat dataset.

   CLI

   Ubah nilai berikut:

   - `project-name`dengan nama proyek yang berisi model yang ingin Anda lihat.
   - `dataset-type`ke jenis kumpulan data yang ingin Anda lihat (`train`atau`test`).

```
aws lookoutvision describe-dataset --project-name project name\
  --dataset-type train or test \
  --profile lookoutvision-access
```

Python

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
    @staticmethod
    def describe_dataset(lookoutvision_client, project_name, dataset_type):
        """
        Gets information about a Lookout for Vision dataset.

        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the dataset
 that
                             you want to describe.
        :param dataset_type: The type (train or test) of the dataset that you
 want
                             to describe.
        """
        try:
            response = lookoutvision_client.describe_dataset(
                ProjectName=project_name, DatasetType=dataset_type
            )
            print(f"Name: {response['DatasetDescription']['ProjectName']}")
            print(f"Type: {response['DatasetDescription']['DatasetType']}")
            print(f"Status: {response['DatasetDescription']['Status']}")
            print(f"Message: {response['DatasetDescription']['StatusMessage']}")
            print(f"Images: {response['DatasetDescription']['ImageStats']
['Total']}")
            print(f"Labeled: {response['DatasetDescription']['ImageStats']
['Labeled']}")
            print(f"Normal: {response['DatasetDescription']['ImageStats']
['Normal']}")
            print(f"Anomaly: {response['DatasetDescription']['ImageStats']
['Anomaly']}")
        except ClientError:
            logger.exception("Service error: problem listing datasets.")
            raise
```

```
        print("Done.")
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
/**
 * Gets the description for a Amazon Lookout for Vision dataset.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to describe a
 *                    dataset.
 * @param datasetType The type of the dataset that you want to describe (train
 *                    or test).
 * @return DatasetDescription A description of the dataset.
 */
public static DatasetDescription describeDataset(LookoutVisionClient lfvClient,
                String projectName,
                String datasetType) throws LookoutVisionException {

        logger.log(Level.INFO, "Describing {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

        DescribeDatasetRequest describeDatasetRequest =
 DescribeDatasetRequest.builder()
                        .projectName(projectName)
                        .datasetType(datasetType)
                        .build();

        DescribeDatasetResponse describeDatasetResponse =
 lfvClient.describeDataset(describeDatasetRequest);
        DatasetDescription datasetDescription =
 describeDatasetResponse.datasetDescription();

        logger.log(Level.INFO, "Project: {0}\n"
                        + "Created: {1}\n"
                        + "Type: {2}\n"
                        + "Total: {3}\n"
                        + "Labeled: {4}\n"
                        + "Normal: {5}\n"
                        + "Anomalous: {6}\n",
```

```
                        new Object[] {
                                        datasetDescription.projectName(),
                                        datasetDescription.creationTimestamp(),
                                        datasetDescription.datasetType(),

 datasetDescription.imageStats().total().toString(),

 datasetDescription.imageStats().labeled().toString(),

 datasetDescription.imageStats().normal().toString(),

 datasetDescription.imageStats().anomaly().toString(),
                        });

        return datasetDescription;

}
```

Menambahkan gambar ke kumpulan data Anda

Setelah membuat kumpulan data, Anda mungkin ingin menambahkan lebih banyak gambar ke kumpulan data. Misalnya, jika evaluasi model menunjukkan model yang buruk, Anda dapat meningkatkan kualitas model Anda dengan menambahkan lebih banyak gambar. Jika Anda telah membuat kumpulan data pengujian, menambahkan lebih banyak gambar dapat meningkatkan akurasi metrik kinerja model Anda.

Latih ulang model Anda setelah memperbarui kumpulan data Anda.

Topik

- Menambahkan lebih banyak gambar
- Menambahkan lebih banyak gambar (SDK)

Menambahkan lebih banyak gambar

Anda dapat menambahkan lebih banyak gambar ke kumpulan data Anda dengan mengunggah gambar dari komputer lokal Anda. Untuk menambahkan lebih banyak gambar berlabel dengan SDK, gunakan operasi. UpdateDatasetEntries

Untuk menambahkan lebih banyak gambar ke kumpulan data Anda (konsol)

1. Pilih Tindakan dan pilih kumpulan data yang ingin Anda tambahkan gambar.
2. Pilih gambar yang ingin Anda unggah ke kumpulan data. Anda dapat menyeret gambar atau memilih gambar yang ingin Anda unggah dari komputer lokal Anda. Anda dapat mengunggah hingga 30 gambar sekaligus.
3. Pilih Unggah gambar.
4. Pilih Save changes (Simpan perubahan).

Setelah selesai menambahkan lebih banyak gambar, Anda perlu memberi label sehingga dapat digunakan untuk melatih model. Untuk informasi selengkapnya, lihat Mengklasifikasikan gambar (konsol).

Menambahkan lebih banyak gambar (SDK)

Untuk menambahkan lebih banyak gambar berlabel dengan SDK, gunakan operasi. UpdateDatasetEntries Anda menyediakan file manifes yang berisi gambar yang ingin Anda tambahkan. Anda juga dapat memperbarui gambar yang ada dengan menentukan gambar di `source-ref` bidang baris JSON dalam file manifes. Untuk informasi selengkapnya, lihat Membuat file manifes.

Untuk menambahkan lebih banyak gambar ke kumpulan data (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode contoh berikut untuk menambahkan lebih banyak gambar ke kumpulan data.

   CLI

   Ubah nilai berikut:

   - `project-name`dengan nama proyek yang berisi kumpulan data yang ingin Anda perbarui.
   - `dataset-type`eke jenis kumpulan data yang ingin Anda perbarui (`train`atau`test`).
   - `changes`ke lokasi file manifes yang berisi pembaruan kumpulan data.

```
aws lookoutvision update-dataset-entries\
    --project-name project\
```

```
    --dataset-type train or test\
    --changes fileb://manifest file \
    --profile lookoutvision-access
```

Python

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
    @staticmethod
    def update_dataset_entries(lookoutvision_client, project_name, dataset_type,
 updates_file):
        """
        Adds dataset entries to an Amazon Lookout for Vision dataset.
        :param lookoutvision_client: The Amazon Rekognition Custom Labels Boto3
 client.
        :param project_name: The project that contains the dataset that you want
 to update.
        :param dataset_type: The type of the dataset that you want to update
 (train or test).
        :param updates_file: The manifest file of JSON Lines that contains the
 updates.
        """

        try:
            status = ""
            status_message = ""
            manifest_file = ""

            # Update dataset entries
            logger.info(f"""Updating {dataset_type} dataset for project
 {project_name}
with entries from {updates_file}.""")

            with open(updates_file) as f:
                manifest_file = f.read()

            lookoutvision_client.update_dataset_entries(
                ProjectName=project_name,
                DatasetType=dataset_type,
                Changes=manifest_file,
            )
```

```
            finished = False
            while finished == False:

                dataset =
lookoutvision_client.describe_dataset(ProjectName=project_name,

DatasetType=dataset_type)

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

                if status == "UPDATE_IN_PROGRESS":
                    logger.info(
                        (f"Updating {dataset_type} dataset for project
{project_name}."))
                    time.sleep(5)
                    continue

                if status == "UPDATE_FAILED_ROLLBACK_IN_PROGRESS":
                    logger.info(
                        (f"Update failed, rolling back {dataset_type} dataset
for project {project_name}."))
                    time.sleep(5)
                    continue

                if status == "UPDATE_COMPLETE":
                    logger.info(
                        f"Dataset updated: {status} : {status_message} :
{dataset_type} dataset for project {project_name}.")
                    finished = True
                    continue

                if status == "UPDATE_FAILED_ROLLBACK_COMPLETE":
                    logger.info(
                        f"Rollback complated after update failure: {status} :
{status_message} : {dataset_type} dataset for project {project_name}.")
                    finished = True
                    continue

                logger.exception(
                    f"Failed. Unexpected state for dataset update: {status} :
{status_message} : {dataset_type} dataset for project {project_name}.")
                raise Exception(
```

```
                    f"Failed. Unexpected state for dataset update: {status} :
 {status_message} :{dataset_type} dataset for project {project_name}.")

            logger.info(f"Added entries to dataset.")

            return status, status_message

        except ClientError as err:
            logger.exception(
                f"Couldn't update dataset: {err.response['Error']['Message']}")
            raise
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
/**
 * Updates an Amazon Lookout for Vision dataset from a manifest file.
 * Returns after Lookout for Vision updates the dataset.
 *
 * @param lfvClient    An Amazon Lookout for Vision client.
 * @param projectName  The name of the project in which you want to update a
 *                     dataset.
 * @param datasetType  The type of the dataset that you want to update (train or
 *                     test).
 * @param manifestFile The name and location of a local manifest file that you
 want to
 * use to update the dataset.
 * @return DatasetStatus The status of the updated dataset.
 */

public static DatasetStatus updateDatasetEntries(LookoutVisionClient lfvClient,
 String projectName,
                String datasetType, String updateFile) throws
 FileNotFoundException, LookoutVisionException,
                InterruptedException {

        logger.log(Level.INFO, "Updating {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

        InputStream sourceStream = new FileInputStream(updateFile);
        SdkBytes sourceBytes = SdkBytes.fromInputStream(sourceStream);
```

```
       UpdateDatasetEntriesRequest updateDatasetEntriesRequest =
UpdateDatasetEntriesRequest.builder()
                        .projectName(projectName)
                        .datasetType(datasetType)
                        .changes(sourceBytes)
                        .build();

       lfvClient.updateDatasetEntries(updateDatasetEntriesRequest);

       boolean finished = false;
       DatasetStatus status = null;

       // Wait until update completes.

       do {

               DescribeDatasetRequest describeDatasetRequest =
DescribeDatasetRequest.builder()
                                .projectName(projectName)
                                .datasetType(datasetType)
                                .build();
               DescribeDatasetResponse describeDatasetResponse = lfvClient
                                .describeDataset(describeDatasetRequest);

               DatasetDescription datasetDescription =
describeDatasetResponse.datasetDescription();

               status = datasetDescription.status();

               switch (status) {

                       case UPDATE_COMPLETE:
                               logger.log(Level.INFO, "{0} Dataset updated for
project {1}.",
                                                new Object[] { datasetType,
projectName });
                               finished = true;
                               break;

                       case UPDATE_IN_PROGRESS:

                               logger.log(Level.INFO, "{0} Dataset update for
project {1} in progress.",
```

```
                                                new Object[] { datasetType,
 projectName });
                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case UPDATE_FAILED_ROLLBACK_IN_PROGRESS:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
 project {1}. Rolling back",
                                                new Object[] { datasetType,
 projectName });

                                TimeUnit.SECONDS.sleep(5);

                                break;

                        case UPDATE_FAILED_ROLLBACK_COMPLETE:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
 project {1}. Rollback completed.",
                                                new Object[] { datasetType,
 projectName });
                                finished = true;
                                break;

                        default:
                                logger.log(Level.SEVERE,
                                                "{0} Dataset update failed for
 project {1}. Unexpected error returned.",
                                                new Object[] { datasetType,
 projectName });
                                finished = true;

                }

        } while (!finished);

        return status;

}
```

3. Ulangi langkah sebelumnya dan berikan nilai untuk jenis dataset lainnya.

Menghapus gambar dari kumpulan data Anda

Anda tidak dapat menghapus gambar secara langsung dari kumpulan data. Sebagai gantinya, Anda harus menghapus kumpulan data yang ada dan membuat kumpulan data baru tanpa gambar yang ingin Anda hapus. Cara Anda menghapus gambar tergantung cara Anda mengimpor gambar ke kumpulan data yang ada (file manifes, bucket Amazon S3, atau komputer lokal).

Anda juga dapat menggunakan AWS SDK untuk menghapus gambar. Ini berguna saat membuat model segmentasi gambar tanpa file manifes segmentasi gambar, sehingga tidak perlu menggambar ulang topeng gambar dengan konsol Amazon Lookout for Vision.

Topik

- Menghapus gambar dari kumpulan data (Konsol)
- Menghapus gambar dari kumpulan data (SDK)

Menghapus gambar dari kumpulan data (Konsol)

Gunakan prosedur berikut untuk menghapus gambar dari kumpulan data dengan konsol Amazon Lookout for Vision.

Untuk menghapus gambar dari kumpulan data (konsol)

1. Buka galeri dataset proyek.
2. Perhatikan nama setiap gambar yang ingin Anda hapus.
3. Hapus kumpulan data yang ada.
4. Lakukan salah satu dari berikut:
   - Jika Anda membuat kumpulan data dengan file manifes, lakukan:
     a. Di editor teks, buka file manifes yang Anda gunakan untuk membuat kumpulan data.
     b. Hapus garis JSON untuk setiap gambar yang Anda catat di langkah 2. Anda dapat mengidentifikasi garis JSON untuk gambar dengan memeriksa `source-ref` bidang.
     c. Simpan file manifes.
     d. Buat kumpulan data baru dengan file manifes yang diperbarui.

- Jika Anda membuat kumpulan data dari gambar yang diimpor dari bucket Amazon S3, lakukan:

  a. Hapus gambar yang Anda catat di langkah 2 dari bucket Amazon S3.

  b. Buat kumpulan data baru dengan gambar yang tersisa di bucket Amazon S3. Jika Anda mengklasifikasikan gambar berdasarkan nama folder, Anda tidak perlu mengklasifikasikan gambar di langkah berikutnya.

  c. Lakukan salah satu dari berikut:

     - Jika Anda membuat model klasifikasi gambar, klasifikasikan setiap gambar yang tidak berlabel.
     - Jika Anda membuat model segmentasi gambar, klasifikasikan dan segmentasikan setiap gambar yang tidak berlabel.

- Jika Anda membuat kumpulan data dari gambar yang diimpor dari komputer lokal, lakukan:

  a. Di komputer Anda, buat folder dengan gambar yang ingin Anda gunakan. Jangan sertakan gambar yang ingin Anda hapus dari kumpulan data. Untuk informasi selengkapnya, lihat Membuat kumpulan data menggunakan gambar yang disimpan di komputer lokal Anda.

  b. Buat kumpulan data dengan gambar di folder yang Anda buat di langkah 4.a.

  c. Lakukan salah satu dari berikut:

     - Jika Anda membuat model klasifikasi gambar, klasifikasikan setiap gambar yang tidak berlabel.
     - Jika Anda membuat model segmentasi gambar, klasifikasikan dan segmentasikan setiap gambar yang tidak berlabel.

5. Latih modelnya.

Menghapus gambar dari kumpulan data (SDK)

Anda dapat menggunakan AWS SDK untuk menghapus gambar dari kumpulan data.

Untuk menghapus gambar dari kumpulan data (SDK)

1. Buka galeri dataset proyek.
2. Perhatikan nama setiap gambar yang ingin Anda hapus.
3. Ekspor garis JSON untuk kumpulan data dengan menggunakan operasi. ListDatasetEntries

4. Buat file manifes dengan baris JSON yang diekspor.
5. Dalam editor teks, buka file manifes.
6. Hapus garis JSON untuk setiap gambar yang Anda catat di langkah 2. Anda dapat mengidentifikasi garis JSON untuk gambar dengan memeriksa `source-ref` bidang.
7. Simpan file manifes.
8. Hapus kumpulan data yang ada.
9. Buat kumpulan data baru dengan file manifes yang diperbarui.
10. Latih modelnya.

Menghapus dataset

Anda dapat menghapus kumpulan data dari proyek dengan menggunakan konsol atau `DeleteDataset` operasi. Gambar yang direferensikan oleh kumpulan data tidak dihapus. Jika Anda menghapus kumpulan data pengujian dari proyek yang memiliki pelatihan dan kumpulan data pengujian, proyek akan kembali ke proyek kumpulan data tunggal—kumpulan data yang tersisa dibagi selama pelatihan untuk membuat kumpulan data pelatihan dan pengujian. Jika Anda menghapus kumpulan data pelatihan, Anda tidak dapat melatih model dalam proyek hingga Anda membuat kumpulan data pelatihan baru.

Menghapus dataset (konsol)

Lakukan langkah-langkah dalam prosedur berikut untuk menghapus kumpulan data. Jika Anda menghapus semua kumpulan data dalam proyek, halaman Buat kumpulan data akan ditampilkan.

Untuk menghapus dataset (konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi kiri, pilih Proyek.
4. Pada halaman Proyek, pilih proyek yang berisi kumpulan data yang ingin Anda hapus.
5. Di panel navigasi kiri, pilih Dataset.
6. Pilih Tindakan dan kemudian pilih kumpulan data yang ingin Anda hapus.
7. Di kotak dialog Hapus, masukkan hapus untuk mengonfirmasi bahwa Anda ingin menghapus kumpulan data.

8. Pilih Hapus kumpulan data pelatihan atau Hapus kumpulan data pengujian untuk menghapus kumpulan data.

Menghapus dataset (SDK)

Gunakan `DeleteDataset` operasi untuk menghapus kumpulan data.

Untuk menghapus dataset (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode contoh berikut untuk menghapus model.

   CLI

   Ubah nilai berikut

   - `project-name`eke nama proyek yang berisi model yang ingin Anda hapus.
   - `dataset-type`eke salah satu `train` atau`test`, tergantung pada kumpulan data mana yang ingin Anda hapus. Jika Anda memiliki proyek kumpulan data tunggal, tentukan `train` untuk menghapus kumpulan data.

   ```
   aws lookoutvision delete-dataset --project-name project name\
     --dataset-type dataset type \
     --profile lookoutvision-access
   ```

   Python

   Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

   ```
       @staticmethod
       def delete_dataset(lookoutvision_client, project_name, dataset_type):
           """
           Deletes a Lookout for Vision dataset

           :param lookoutvision_client: A Boto3 Lookout for Vision client.
           :param project_name: The name of the project that contains the dataset
   that
   ```

```
                            you want to delete.
        :param dataset_type: The type (train or test) of the dataset that you
                            want to delete.
        """
        try:
            logger.info(
                "Deleting the %s dataset for project %s.", dataset_type,
 project_name
            )
            lookoutvision_client.delete_dataset(
                ProjectName=project_name, DatasetType=dataset_type
            )
            logger.info("Dataset deleted.")
        except ClientError:
            logger.exception("Service error: Couldn't delete dataset.")
            raise
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
/**
 * Deletes the train or test dataset in an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project in which you want to delete a
 *                    dataset.
 * @param datasetType The type of the dataset that you want to delete (train or
 *                    test).
 * @return Nothing.
 */
public static void deleteDataset(LookoutVisionClient lfvClient, String
 projectName, String datasetType)
                throws LookoutVisionException {

        logger.log(Level.INFO, "Deleting {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

        DeleteDatasetRequest deleteDatasetRequest =
 DeleteDatasetRequest.builder()
                        .projectName(projectName)
```

```
                        .datasetType(datasetType)
                        .build();

        lfvClient.deleteDataset(deleteDatasetRequest);

        logger.log(Level.INFO, "Deleted {0} dataset for project {1}",
                        new Object[] { datasetType, projectName });

}
```

Mengekspor kumpulan data dari proyek (SDK)

Anda dapat menggunakan AWS SDK untuk mengekspor kumpulan data dari project Amazon Lookout for Vision ke lokasi bucket Amazon S3.

Dengan mengekspor kumpulan data, Anda dapat melakukan tugas-tugas seperti membuat proyek Lookout for Vision dengan salinan kumpulan data proyek sumber. Anda juga dapat membuat snapshot dari kumpulan data yang digunakan untuk versi model tertentu.

Kode Python dalam prosedur ini mengekspor kumpulan data pelatihan (gambar manifes dan dataset) untuk proyek ke lokasi Amazon S3 tujuan yang Anda tentukan. Jika ada dalam proyek, kode juga mengekspor manifes kumpulan data pengujian dan gambar kumpulan data. Tujuannya bisa berada di bucket Amazon S3 yang sama dengan proyek sumber, atau bucket Amazon S3 yang berbeda. Kode menggunakan ListDatasetEntriesoperasi untuk mendapatkan file manifes kumpulan data. Operasi Amazon S3 menyalin gambar kumpulan data dan file manifes yang diperbarui ke lokasi Amazon S3 tujuan.

Prosedur ini menunjukkan cara mengekspor kumpulan data proyek. Ini juga menunjukkan cara membuat proyek baru dengan dataset yang diekspor.

Untuk mengekspor kumpulan data dari proyek (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Tentukan jalur Amazon S3 tujuan untuk ekspor kumpulan data. Pastikan bahwa tujuan berada di AWSWilayah yang didukung Amazon Lookout for Vision. Untuk membuat bucket Amazon S3 baru, lihat Membuat bucket.
3. Pastikan pengguna memiliki izin akses ke jalur Amazon S3 tujuan untuk ekspor set data dan lokasi S3 untuk file gambar dalam kumpulan data proyek sumber. Anda dapat menggunakan

kebijakan berikut yang mengasumsikan file gambar dapat berada di lokasi mana pun. Ganti *ember/jalur* dengan bucket tujuan dan jalur untuk ekspor dataset.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PutExports",
            "Effect": "Allow",
            "Action": [
                "S3:PutObjectTagging",
                "S3:PutObject"
            ],
            "Resource": "arn:aws:s3:::bucket/path/*"
        },
        {
            "Sid": "GetSourceRefs",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion"
            ],
            "Resource": "*"
        }
    ]
}
```

Untuk memberikan akses, tambahkan izin ke pengguna, grup, atau peran Anda:

- Pengguna dan grup diAWS IAM Identity Center:

  Buat set izin. Ikuti petunjuk di Buat set izin di Panduan AWS IAM Identity Center Pengguna.

- Pengguna yang dikelola di IAM melalui penyedia identitas:

  Buat peran untuk federasi identitas. Ikuti petunjuk dalam Membuat peran untuk penyedia identitas pihak ketiga (federasi) di Panduan Pengguna IAM.

- Pengguna IAM:
  - Buat peran yang dapat diasumsikan oleh pengguna Anda. Ikuti petunjuk dalam Membuat peran untuk pengguna IAM di Panduan Pengguna IAM.

- (Tidak disarankan) Lampirkan kebijakan langsung ke pengguna atau tambahkan pengguna ke grup pengguna. Ikuti petunjuk di Menambahkan izin ke pengguna (konsol) di Panduan Pengguna IAM.

4. Simpan kode berikut ke file bernama`dataset_export.py`.

```
"""
Purpose

Shows how to export the datasets (manifest files and images)
from an Amazon Lookout for Vision project to a new Amazon
S3 location.
"""

import argparse
import json
import logging

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)


def copy_file(s3_resource, source_file, destination_file):
    """
    Copies a file from a source Amazon S3 folder to a destination
    Amazon S3 folder.
    The destination can be in a different S3 bucket.
    :param s3: An Amazon S3 Boto3 resource.
    :param source_file: The Amazon S3 path to the source file.
    :param destination_file: The destination Amazon S3 path for
    the copy operation.
    """

    source_bucket, source_key = source_file.replace("s3://", "").split("/", 1)
    destination_bucket, destination_key = destination_file.replace("s3://",
 "").split(
        "/", 1
    )

    try:
```

```
        bucket = s3_resource.Bucket(destination_bucket)
        dest_object = bucket.Object(destination_key)
        dest_object.copy_from(CopySource={"Bucket": source_bucket, "Key":
 source_key})
        dest_object.wait_until_exists()
        logger.info("Copied %s to %s", source_file, destination_file)
    except ClientError as error:
        if error.response["Error"]["Code"] == "404":
            error_message = (
                f"Failed to copy {source_file} to "
                f"{destination_file}. : {error.response['Error']['Message']}"
            )
            logger.warning(error_message)
            error.response["Error"]["Message"] = error_message
        raise


def upload_manifest_file(s3_resource, manifest_file, destination):
    """
    Uploads a manifest file to a destination Amazon S3 folder.
    :param s3: An Amazon S3 Boto3 resource.
    :param manifest_file: The manifest file that you want to upload.
    :destination: The Amazon S3 folder location to upload the manifest
    file to.
    """

    destination_bucket, destination_key = destination.replace("s3://",
 "").split("/", 1)

    bucket = s3_resource.Bucket(destination_bucket)

    put_data = open(manifest_file, "rb")
    obj = bucket.Object(destination_key + manifest_file)

    try:
        obj.put(Body=put_data)
        obj.wait_until_exists()
        logger.info("Put manifest file '%s' to bucket '%s'.", obj.key,
 obj.bucket_name)
    except ClientError:
        logger.exception(
            "Couldn't put manifest file '%s' to bucket '%s'.", obj.key,
 obj.bucket_name
        )
```

```
        raise
    finally:
        if getattr(put_data, "close", None):
            put_data.close()


def get_dataset_types(lookoutvision_client, project):
    """
    Determines the types of the datasets (train or test) in an
    Amazon Lookout for Vision project.
    :param lookoutvision_client: A Lookout for Vision Boto3 client.
    :param project: The Lookout for Vision project that you want to check.
    :return: The dataset types in the project.
    """

    try:
        response = lookoutvision_client.describe_project(ProjectName=project)

        datasets = []

        for dataset in response["ProjectDescription"]["Datasets"]:
            if dataset["Status"] in ("CREATE_COMPLETE", "UPDATE_COMPLETE"):
                datasets.append(dataset["DatasetType"])
        return datasets

    except lookoutvision_client.exceptions.ResourceNotFoundException:
        logger.exception("Project %s not found.", project)
        raise


def process_json_line(s3_resource, entry, dataset_type, destination):
    """
    Creates a JSON line for a new manifest file, copies image and mask to
    destination.
    :param s3_resource: An Amazon S3 Boto3 resource.
    :param entry: A JSON line from the manifest file.
    :param dataset_type: The type (train or test) of the dataset that
    you want to create the manifest file for.
    :param destination: The destination Amazon S3 folder for the manifest
    file and dataset images.
    :return: A JSON line with details for the destination location.
    """
    entry_json = json.loads(entry)
```

```
    print(f"source: {entry_json['source-ref']}")

    # Use existing folder paths to ensure console added image names don't clash.
    bucket, key = entry_json["source-ref"].replace("s3://", "").split("/", 1)
    logger.info("Source location: %s/%s", bucket, key)

    destination_image_location = destination + dataset_type + "/images/" + key

    copy_file(s3_resource, entry_json["source-ref"], destination_image_location)

    # Update JSON for writing.
    entry_json["source-ref"] = destination_image_location

    if "anomaly-mask-ref" in entry_json:
        source_anomaly_ref = entry_json["anomaly-mask-ref"]
        mask_bucket, mask_key = source_anomaly_ref.replace("s3://", "").split("/",
 1)

        destination_mask_location = destination + dataset_type + "/masks/" +
 mask_key
        entry_json["anomaly-mask-ref"] = destination_mask_location

        copy_file(s3_resource, source_anomaly_ref, entry_json["anomaly-mask-ref"])

    return entry_json


def write_manifest_file(
    lookoutvision_client, s3_resource, project, dataset_type, destination
):
    """
    Creates a manifest file for a dataset. Copies the manifest file and
    dataset images (and masks, if present) to the specified Amazon S3 destination.
    :param lookoutvision_client: A Lookout for Vision Boto3 client.
    :param project: The Lookout for Vision project that you want to use.
    :param dataset_type: The type (train or test) of the dataset that
    you want to create the manifest file for.
    :param destination: The destination Amazon S3 folder for the manifest file
    and dataset images.
    """

    try:
        # Create a reusable Paginator
        paginator = lookoutvision_client.get_paginator("list_dataset_entries")
```

```
        # Create a PageIterator from the Paginator
        page_iterator = paginator.paginate(
            ProjectName=project,
            DatasetType=dataset_type,
            PaginationConfig={"PageSize": 100},
        )

        output_manifest_file = dataset_type + ".manifest"

        # Create manifest file then upload to Amazon S3 with images.
        with open(output_manifest_file, "w", encoding="utf-8") as manifest_file:
            for page in page_iterator:
                for entry in page["DatasetEntries"]:
                    try:
                        entry_json = process_json_line(
                            s3_resource, entry, dataset_type, destination
                        )

                        manifest_file.write(json.dumps(entry_json) + "\n")

                    except ClientError as error:
                        if error.response["Error"]["Code"] == "404":
                            print(error.response["Error"]["Message"])
                            print(f"Excluded JSON line: {entry}")
                        else:
                            raise
        upload_manifest_file(
            s3_resource, output_manifest_file, destination + "datasets/"
        )

    except ClientError:
        logger.exception("Problem getting dataset_entries")
        raise


def export_datasets(lookoutvision_client, s3_resource, project, destination):
    """
    Exports the datasets from an Amazon Lookout for Vision project to a specified
    Amazon S3 destination.
    :param project: The Lookout for Vision project that you want to use.
    :param destination: The destination Amazon S3 folder for the exported datasets.
    """
    # Add trailing backslash, if missing.
```

```
    destination = destination if destination[-1] == "/" else destination + "/"

    print(f"Exporting project {project} datasets to {destination}.")

    # Get each dataset and export to destination.

    dataset_types = get_dataset_types(lookoutvision_client, project)
    for dataset in dataset_types:
        logger.info("Copying %s dataset to %s.", dataset, destination)

        write_manifest_file(
            lookoutvision_client, s3_resource, project, dataset, destination
        )

    print("Exported dataset locations")
    for dataset in dataset_types:
        print(f"   {dataset}: {destination}datasets/{dataset}.manifest")

    print("Done.")


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

    parser.add_argument("project", help="The project that contains the dataset.")
    parser.add_argument("destination", help="The destination Amazon S3 folder.")


def main():
    """
    Exports the datasets from an Amazon Lookout for Vision project to a
    destination Amazon S3 location.
    """
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    parser = argparse.ArgumentParser(usage=argparse.SUPPRESS)
    add_arguments(parser)

    args = parser.parse_args()

    try:
        session = boto3.Session(profile_name="lookoutvision-access")
```

```
        lookoutvision_client = session.client("lookoutvision")
        s3_resource = session.resource("s3")

        export_datasets(
            lookoutvision_client, s3_resource, args.project, args.destination
        )
    except ClientError as err:
        logger.exception(err)
        print(f"Failed: {format(err)}")


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

5. Jalankan kode tersebut. Berikan argumen baris perintah berikut:

   - project — Nama proyek sumber yang berisi dataset yang ingin Anda ekspor.
   - tujuan - Jalur Amazon S3 tujuan untuk kumpulan data.

   Misalnya, `python dataset_export.py` *`myproject`* `s3://`*`bucket/path/`*

6. Perhatikan lokasi file manifes yang ditampilkan kode. Anda membutuhkannya di langkah 8.
7. Buat proyek Lookout for Vision baru dengan dataset yang diekspor dengan mengikuti petunjuk di. Membuat proyek Anda
8. Lakukan salah satu dari berikut:

   - Gunakan konsol Lookout for Vision untuk membuat kumpulan data untuk proyek baru Anda dengan mengikuti petunjuk di. Membuat kumpulan data dengan file manifes (konsol) Anda tidak perlu melakukan langkah 1-6.

     Untuk langkah 12, lakukan hal berikut:

     a. Jika proyek sumber memiliki kumpulan data pengujian, pilih Daftar data pelatihan dan pengujian terpisah, jika tidak pilih kumpulan data tunggal.

     b. Untuk lokasi file.manifest, masukkan lokasi file manifes yang sesuai (latih atau uji) yang Anda catat di langkah 6.

   - Gunakan CreateDatasetoperasi untuk membuat kumpulan data untuk proyek baru Anda dengan menggunakan kode di. Membuat kumpulan data dengan file manifes (SDK) Untuk `manifest_file` parameter, gunakan lokasi file manifes yang Anda catat di langkah 6.

Jika proyek sumber memiliki kumpulan data pengujian, gunakan kode lagi untuk membuat kumpulan data pengujian.

9. Jika Anda siap, latih model dengan mengikuti instruksi diMelatih model Anda.

Melihat model Anda

Sebuah proyek dapat memiliki beberapa versi model. Anda dapat menggunakan konsol untuk melihat model dalam proyek. Anda juga dapat menggunakan `ListModels` operasi ini.

> **Note**
>
> Daftar model pada akhirnya konsisten. Jika Anda membuat model, Anda mungkin harus menunggu beberapa saat sebelum daftar model diperbarui.

Melihat model Anda (konsol)

Lakukan langkah-langkah dalam prosedur berikut untuk melihat model proyek Anda di konsol.

Untuk melihat model Anda (konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi kiri, pilih Proyek.
4. Pada halaman Proyek, pilih proyek yang berisi model yang ingin Anda lihat.
5. Di panel navigasi kiri, pilih Model dan kemudian lihat detail model.

Melihat model Anda (SDK)

Untuk mendapatkan tampilan versi model Anda menggunakan `ListModels` operasi. Untuk mendapatkan informasi tentang versi model tertentu, gunakan `DescribeModel` operasi. Contoh berikut mencantumkan semua versi model dalam sebuah proyek dan kemudian menampilkan informasi konfigurasi kinerja dan output untuk versi model individual.

Untuk melihat model Anda (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode contoh berikut untuk membuat daftar model Anda dan mendapatkan informasi tentang model.

   CLI

   Gunakan `list-models` perintah untuk membuat daftar model dalam proyek.

   Ubah nilai berikut:

   - `project-name`dengan nama proyek yang berisi model yang ingin Anda lihat.

   ```
   aws lookoutvision list-models --project-name project name \
     --profile lookoutvision-access
   ```

   Gunakan `describe-model` perintah untuk mendapatkan informasi tentang model. Ubah nilai berikut:

   - `project-name`dengan nama proyek yang berisi model yang ingin Anda lihat.
   - `model-version`ke versi model yang ingin Anda gambarkan.

   ```
   aws lookoutvision describe-model --project-name project name\
       --model-version model version \
      --profile lookoutvision-access
   ```

   Python

   Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

   ```
       @staticmethod
       def describe_models(lookoutvision_client, project_name):
           """
           Gets information about all models in a Lookout for Vision project.
   ```

```
        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that you want to use.
        """
        try:
            response =
 lookoutvision_client.list_models(ProjectName=project_name)
            print("Project: " + project_name)
            for model in response["Models"]:
                Models.describe_model(
                    lookoutvision_client, project_name, model["ModelVersion"]
                )
                print()
            print("Done...")
        except ClientError:
            logger.exception("Couldn't list models.")
            raise
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
/**
 * Lists the models in an Amazon Lookout for Vision project.
 *
 * @param lfvClient   An Amazon Lookout for Vision client.
 * @param projectName The name of the project that contains the models that
 *                    you want to list.
 * @return List <Metadata> A list of models in the project.
 */
public static List<ModelMetadata> listModels(LookoutVisionClient lfvClient,
 String projectName)
                throws LookoutVisionException {

        ListModelsRequest listModelsRequest = ListModelsRequest.builder()
                        .projectName(projectName)
                        .build();

        // Get a list of models in the supplied project.
        ListModelsResponse response = lfvClient.listModels(listModelsRequest);

        for (ModelMetadata model : response.models()) {
```

```
                logger.log(Level.INFO, "Model ARN: {0}\nVersion: {1}\nStatus:
 {2}\nMessage: {3}", new Object[] {
                                model.modelArn(),
                                model.modelVersion(),
                                model.statusMessage(),
                                model.statusAsString() });
        }

        return response.models();

}
```

Menghapus model

Anda dapat menghapus versi model dengan menggunakan konsol atau dengan menggunakan `DeleteModel` operasi. Anda tidak dapat menghapus versi model yang sedang berjalan atau sedang dilatih.

Jika model menjalankan versi, pertama-tama gunakan `StopModel` operasi untuk menghentikan versi model. Untuk informasi selengkapnya, lihat Menghentikan model Amazon Lookout for Vision Anda. Jika model sedang dilatih, tunggu hingga selesai sebelum Anda menghapus model.

Mungkin perlu beberapa detik untuk menghapus model. Untuk menentukan apakah model telah dihapus, panggil ListProjectsdan periksa apakah versi model (`ModelVersion`) ada dalam `Models` array.

Menghapus model (konsol)

Lakukan langkah-langkah berikut untuk menghapus model dari konsol.

Untuk menghapus model (konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi kiri, pilih Proyek.
4. Pada halaman Proyek, pilih proyek yang berisi model yang ingin Anda hapus.
5. Di panel navigasi kiri, pilih Model.
6. Pada tampilan model, pilih tombol radio untuk model yang ingin Anda hapus.

7. Pilih Hapus di bagian atas halaman.
8. Di kotak dialog Hapus, masukkan hapus untuk mengonfirmasi bahwa Anda ingin menghapus model.
9. Pilih Hapus model untuk menghapus model.

Menghapus model (SDK)

Gunakan prosedur berikut untuk menghapus model dengan `DeleteModel` operasi.

Untuk menghapus model (SDK)

1. Jika Anda belum melakukannya, instal dan konfigurasikan AWS CLI dan AWS SDK. Untuk informasi selengkapnya, lihat Langkah 4: Siapkan AWS CLI dan AWS SDK.
2. Gunakan kode contoh berikut untuk menghapus model.

   CLI

   Ubah nilai berikut:

   - `project-name`ke nama proyek yang berisi model yang ingin Anda hapus.
   - `model-version`ke versi model yang ingin Anda hapus.

   ```
   aws lookoutvision delete-model --project-name project name\
     --model-version model version \
     --profile lookoutvision-access
   ```

   Python

   Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

   ```
       @staticmethod
       def delete_model(lookoutvision_client, project_name, model_version):
           """
           Deletes a Lookout for Vision model. The model must first be stopped and
    can't
           be in training.
   ```

```
        :param lookoutvision_client: A Boto3 Lookout for Vision client.
        :param project_name: The name of the project that contains the desired
model.
        :param model_version: The version of the model that you want to delete.
        """
        try:
            logger.info("Deleting model: %s", model_version)
            lookoutvision_client.delete_model(
                ProjectName=project_name, ModelVersion=model_version
            )

            model_exists = True
            while model_exists:
                response =
lookoutvision_client.list_models(ProjectName=project_name)

                model_exists = False
                for model in response["Models"]:
                    if model["ModelVersion"] == model_version:
                        model_exists = True

                if model_exists is False:
                    logger.info("Model deleted")
                else:
                    logger.info("Model is being deleted...")
                    time.sleep(2)

            logger.info("Deleted Model: %s", model_version)
        except ClientError:
            logger.exception("Couldn't delete model.")
            raise
```

Java V2

Kode ini diambil dari GitHub repositori contoh SDK AWS Dokumentasi. Lihat contoh lengkapnya di sini.

```
/**
* Deletes an Amazon Lookout for Vision model.
*
```

```
* @param lfvClient    An Amazon Lookout for Vision client. Returns after the
 model is deleted.
* @param projectName  The name of the project that contains the model that you
 want to delete.
* @param modelVersion The version of the model that you want to delete.
* @return void
*/
public static void deleteModel(LookoutVisionClient lfvClient,
                String projectName,
                String modelVersion) throws LookoutVisionException,
 InterruptedException {

        DeleteModelRequest deleteModelRequest = DeleteModelRequest.builder()
                        .projectName(projectName)
                        .modelVersion(modelVersion)
                        .build();

        lfvClient.deleteModel(deleteModelRequest);

        boolean deleted = false;

        do {

                ListModelsRequest listModelsRequest =
 ListModelsRequest.builder()
                                .projectName(projectName)
                                .build();

                // Get a list of models in the supplied project.
                ListModelsResponse response =
 lfvClient.listModels(listModelsRequest);

                ModelMetadata modelMetadata = response.models().stream()
                                .filter(model ->
 model.modelVersion().equals(modelVersion)).findFirst()
                                .orElse(null);

                if (modelMetadata == null) {
                        deleted = true;
                        logger.log(Level.INFO, "Deleted: Model version {0} of
 project {1}.",
                                        new Object[] { modelVersion,
 projectName });
```

```
                } else {
                        logger.log(Level.INFO, "Not yet deleted: Model version
 {0} of project {1}.",
                                        new Object[] { modelVersion,
 projectName });
                        TimeUnit.SECONDS.sleep(60);
                }

        } while (!deleted);

}
```

Model penandaan

Anda dapat mengidentifikasi, mengatur, mencari, dan memfilter model Amazon Lookout for Vision Anda dengan menggunakan tag. Setiap tag adalah label yang terdiri dari kunci dan nilai yang ditentukan pengguna. Misalnya, untuk membantu menentukan penagihan untuk model Anda, Anda dapat menandai model Anda dengan `Cost center` kunci dan menambahkan nomor pusat biaya yang sesuai sebagai nilai. Untuk informasi selengkapnya, lihat Menandai sumber daya AWS.

Gunakan tag untuk:

- Lacak penagihan untuk model dengan menggunakan tag alokasi biaya. Untuk informasi selengkapnya, lihat Menggunakan Tag Alokasi Biaya.
- Kontrol akses ke model dengan menggunakan Identity and Access Management (IAM). Untuk informasi selengkapnya, lihat Mengontrol akses ke sumber daya AWS menggunakan tag sumber daya.
- Mengotomatiskan manajemen model. Misalnya, Anda dapat menjalankan skrip start atau stop otomatis yang mematikan model pengembangan selama jam non-bisnis untuk mengurangi biaya. Untuk informasi selengkapnya, lihat Menjalankan model Amazon Lookout for Vision Anda yang terlatih.

Anda dapat menandai model dengan menggunakan konsol Amazon Lookout for Vision atau dengan menggunakan SDK. AWS

Topik

- Model penandaan (konsol)

- Model penandaan (SDK)

Model penandaan (konsol)

Anda dapat menggunakan konsol Amazon Lookout for Vision untuk menambahkan tag ke model, melihat tag yang dilampirkan ke model, dan menghapus tag.

Menambahkan atau menghapus tag (konsol)

Prosedur ini menjelaskan cara menambahkan tag ke, atau menghapus tag dari, model yang ada. Anda juga dapat menambahkan tag ke model baru saat dilatih. Untuk informasi selengkapnya, lihat Melatih model Anda.

Untuk menambahkan tag ke, atau menghapus tag dari, model yang ada (konsol)

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi, pilih Proyek.
4. Pada halaman Sumber daya proyek, pilih proyek yang berisi model yang ingin Anda tag.
5. Di panel navigasi, di bawah proyek yang Anda pilih sebelumnya, pilih Model.
6. Di bagian Model, pilih model yang ingin Anda tambahkan tag.
7. Pada halaman detail model, pilih tab Tag.
8. Di bagian Tag, pilih Kelola tag.
9. Pada halaman Kelola tag, pilih Tambahkan tag baru.
10. Masukkan kunci dan nilai.
    a. Untuk Key, masukkan nama untuk kunci tersebut.
    b. Untuk Nilai, masukkan nilai.
11. Untuk menambahkan lebih banyak tag, ulangi langkah 9 dan 10.
12. (Opsional) Untuk menghapus tag, pilih Hapus di samping tag yang ingin Anda hapus. Jika Anda menghapus tag yang disimpan sebelumnya, tag tersebut akan dihapus saat Anda menyimpan perubahan.
13. Pilih Simpan perubahan untuk menyimpan perubahan Anda.

Melihat tag model (konsol)

Anda dapat menggunakan konsol Amazon Lookout for Vision untuk melihat tag yang dilampirkan ke model.

Untuk melihat tag yang dilampirkan ke semua model dalam proyek, Anda harus menggunakan AWS SDK. Untuk informasi selengkapnya, lihat Daftar tag model (SDK).

Untuk melihat tag yang dilampirkan ke model

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi, pilih Proyek.
4. Pada halaman Sumber daya proyek, pilih proyek yang berisi model yang tagnya ingin Anda lihat.
5. Di panel navigasi, di bawah proyek yang Anda pilih sebelumnya, pilih Model.
6. Di bagian Model, pilih model yang tagnya ingin Anda lihat.
7. Pada halaman detail model, pilih tab Tag. Tag ditampilkan di bagian Tag.

Model penandaan (SDK)

Anda dapat menggunakan AWS SDK untuk:

- Tambahkan tag ke model baru
- Tambahkan tag ke model yang ada
- Buat daftar tag yang dilampirkan pada model
- Hapus tag dari model

Bagian ini mencakup AWS CLI contoh. Jika Anda belum menginstalAWS CLI, lihatLangkah 4: Siapkan AWS CLI dan AWS SDK.

Menambahkan tag ke model baru (SDK)

Anda dapat menambahkan tag ke model saat Anda membuatnya menggunakan CreateModeloperasi. Tentukan satu tanda atau lebih dalam parameter input array `Tags`.

```
aws lookoutvision create-model --project-name "project name"\
```

```
  --output-config '{ "S3Location": { "Bucket": "output bucket", "Prefix":  "output
 folder" } }'\
  --tags '[{"Key":"Key","Value":"Value"}]' \
  --profile lookoutvision-access
```

Untuk informasi tentang membuat dan melatih model, lihatMelatih model (SDK).

Menambahkan tag ke model yang ada (SDK)

Untuk menambahkan satu atau beberapa tag ke model yang ada, gunakan TagResourceoperasi. Tentukan Amazon Resource Name (ARN) (`ResourceArn`) model dan tag (`Tags`) yang ingin Anda tambahkan.

```
aws lookoutvision tag-resource --resource-arn "resource-arn"\
  --tags '[{"Key":"Key","Value":"Value"}]' \
  --profile lookoutvision-access
```

Misalnya kode Java, lihat TagModel.

Daftar tag model (SDK)

Untuk mencantumkan tag yang dilampirkan ke model, gunakan ListTagsForResourceoperasi dan tentukan Amazon Resource Name (ARN) model, the (`ResourceArn`). Responsnya adalah peta kunci tag dan nilai yang dilampirkan ke model yang ditentukan.

```
aws lookoutvision list-tags-for-resource --resource-arn resource-arn \
  --profile lookoutvision-access
```

Untuk melihat model mana dalam proyek yang memiliki tag tertentu, hubungi `ListModels` untuk mendapatkan daftar model. Kemudian panggil `ListTagsForResource` setiap model dalam tanggapan dari`ListModels`. Periksa respons dari `ListTagsForResource` untuk melihat apakah tag yang diperlukan ada.

Misalnya kode Java, lihat ListModelTags. Misalnya kode Python yang mencari nilai tag di semua proyek, lihat find_tag.py.

Menghapus tag dari model (SDK)

Untuk menghapus satu atau lebih tag dari model, gunakan UntagResourceoperasi. Tentukan Amazon Resource Name (ARN) (`ResourceArn`) model dan kunci tag (`Tag-Keys`) yang ingin Anda hapus.

```
aws lookoutvision untag-resource --resource-arn resource-arn\
  --tag-keys '["Key"]' \
  --profile lookoutvision-access
```

Misalnya kode Java, lihat UntagModel.

Melihat tugas deteksi uji coba

Anda dapat melihat deteksi uji coba menggunakan konsol. Anda tidak dapat menggunakan AWS SDK untuk melihat tugas deteksi uji coba.

> **Note**
>
> Daftar deteksi percobaan pada akhirnya konsisten. Jika Anda membuat deteksi uji coba, Anda mungkin harus menunggu beberapa saat sebelum daftar deteksi uji coba diperbarui.

Melihat tugas deteksi uji coba Anda (konsol)

Gunakan prosedur berikut untuk melihat deteksi uji coba Anda.

Untuk melihat tugas deteksi uji coba

1. Buka konsol Amazon Lookout for Vision di https://console.aws.amazon.com/lookoutvision/.
2. Pilih Mulai.
3. Di panel navigasi kiri, pilih Deteksi percobaan.
4. Pada halaman deteksi uji coba, pilih tugas deteksi uji coba untuk melihat detailnya.

Contoh kode dan kumpulan data

Berikut ini adalah contoh kode dan kumpulan data yang dapat Anda gunakan dengan Amazon Lookout for Vision.

Topik

Kode contoh

Contoh kode berikut untuk Amazon Lookout for Vision tersedia.

Contoh	Deskripsi
GitHub	Contoh kode Python yang melatih dan menghosting model Amazon Lookout for Vision.
Amazon Lookout for Vision Lab	Notebook Python yang dapat Anda gunakan untuk membuat model dengan gambar contoh circuitboard.
Kode contoh Python	Contoh Python digunakan dalam dokumentasi Amazon Lookout for Vision.
Contoh kode Java	Contoh Java digunakan dalam dokumentasi Amazon Lookout for Vision.

set data contoh

Berikut ini adalah contoh kumpulan data yang dapat Anda gunakan dengan Amazon Lookout for Vision.

Topik

- Set data klasifikasi gambar

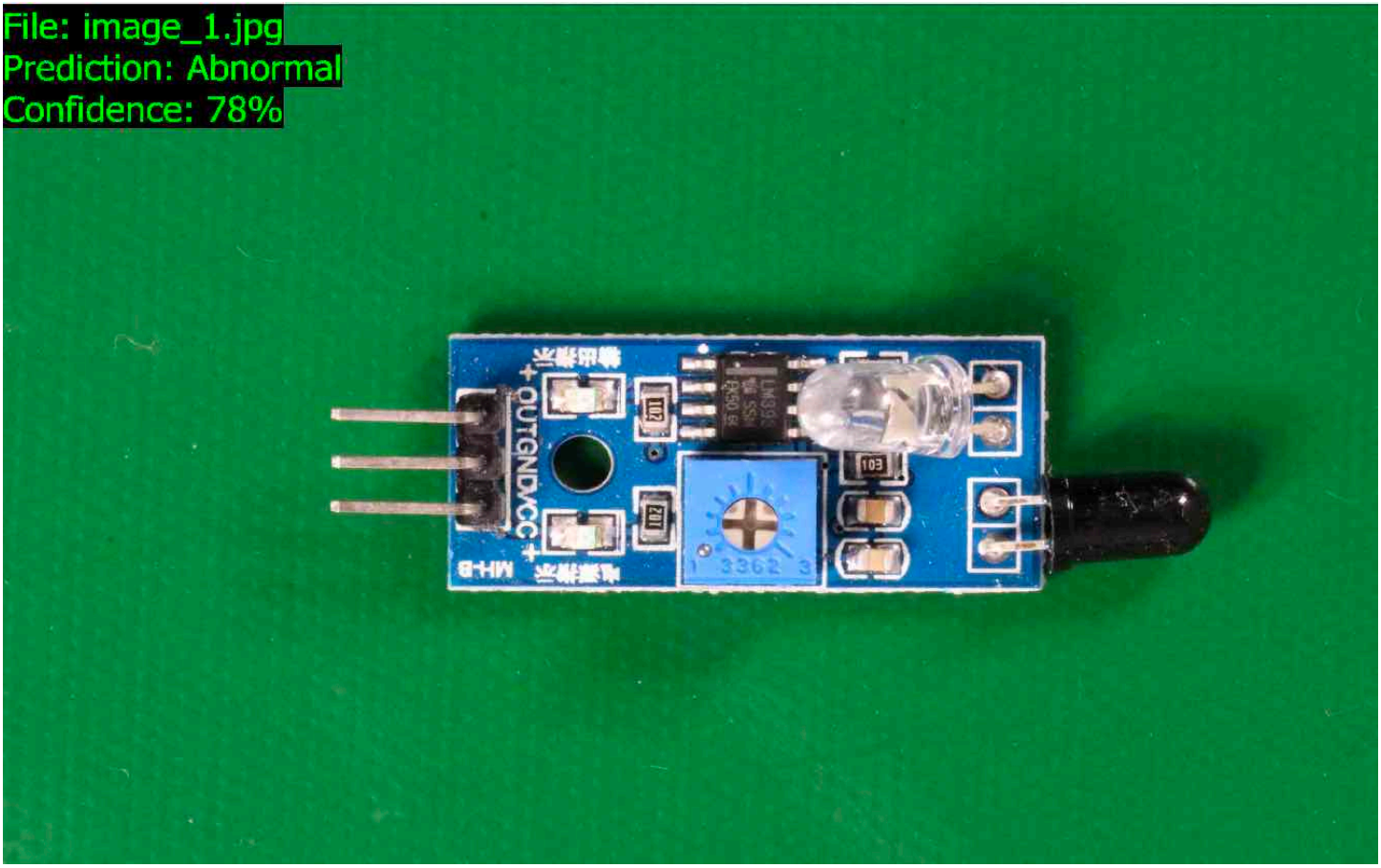
Set data segmentasi gambar

Memulai dengan Amazon Lookout for Visionmenyediakan dataset cookie rusak yang dapat Anda gunakan untuk membuat model segmentasi gambar.

Untuk kumpulan data lain yang membuat model segmentasi gambar, lihat Mengidentifikasi lokasi anomali menggunakan Amazon Lookout for Vision di tepi tanpa menggunakan GPU.

Set data klasifikasi gambar

Amazon Lookout for Vision memberikan contoh gambar papan sirkuit yang dapat Anda gunakan untuk membuat model klasifikasi gambar.



Anda dapat menyalin gambar dariamazon-lookout-for-vision GitHub repositori https://github.com/aws-samples/. Gambar ada di`circuitboard` folder.

`circuitboard`Folder memiliki folder berikut.

- `train`- Gambar yang dapat Anda gunakan dalam kumpulan data pelatihan.
- `test`- Gambar yang dapat Anda gunakan dalam kumpulan data pengujian.
- `extra_images`- Gambar yang dapat Anda gunakan untuk menjalankan deteksi uji coba atau untuk mencoba model terlatih Anda dengan DetectAnomaliesoperasi.

`testFoldertrain` dan masing-masing memiliki subfolder bernama`normal` (berisi gambar yang normal) dan subfolder bernama`anomaly` (berisi gambar dengan anomali).

> **Note**
>
> Kemudian, saat Anda membuat kumpulan data dengan konsol, Amazon Lookout for Vision dapat menggunakan nama folder (`normal`dan`anomaly`) untuk memberi label pada gambar secara otomatis. Untuk informasi selengkapnya, lihat the section called “Bucket Amazon S3”.

Untuk menyiapkan gambar set data

1. Kloningamazon-lookout-for-vision repositori https://github.com/aws-samples/ ke komputer Anda. Untuk informasi selengkapnya, lihat Mengkloning repositori.
2. Buat bucket Amazon S3. Untuk informasi selengkapnya, lihat Bagaimana cara membuat bucket S3? .
3. Pada prompt perintah, masukkan perintah berikut untuk menyalin gambar set data dari komputer ke bucket Amazon S3.

```
aws s3 cp --recursive your-repository-folder/circuitboard s3://your-bucket/
circuitboard
```

Setelah mengunggah gambar, Anda dapat membuat model. Anda dapat secara otomatis mengklasifikasikan gambar dengan menambahkan gambar dari lokasi Amazon S3 yang sebelumnya Anda unggah gambar papan sirkuit. Ingatlah bahwa Anda dikenai biaya untuk setiap pelatihan model yang berhasil dan untuk jumlah waktu model berjalan (dihosting).

Untuk membuat model klasifikasi

1. LakukanMembuat proyek (konsol).
2. LakukanMembuat kumpulan data menggunakan gambar yang disimpan di bucket Amazon S3.

- Untuk langkah 6, pilih tab Set data pelatihan dan uji terpisah.
- Untuk langkah 8a, masukkan URI S3 untuk gambar latihan yang Anda unggah Untuk menyiapkan gambar set data. Misalnya, `s3://your-bucket/circuitboard/train`. Untuk langkah 8b, masukkan URI S3 untuk kumpulan data pengujian. Sebagai contoh, `s3://your-bucket/circuitboard/test`.
- Pastikan untuk melakukan langkah 9.

3. LakukanMelatih model (konsol).
4. LakukanMemulai model Anda (konsol).
5. LakukanMendeteksi anomali dalam gambar. Anda dapat menggunakan gambar dari`test_images` folder.
6. Setelah Anda selesai dengan model, lakukanMenghentikan model Anda (konsol).

Amazon Lookout for Vision

Keamanan cloud di AWS merupakan prioritas tertinggi. Sebagai pelanggan AWS, Anda mendapatkan manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara AWS dan Anda. Model tanggung jawab bersama menggambarkan ini sebagai keamanan dari cloud dan keamanan di dalam cloud:

- Keamanan cloud – AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan layanan-layanan AWS di dalam AWS Cloud. AWS juga memberikan Anda layanan yang dapat digunakan dengan aman. Auditor pihak ketiga melakukan pengujian dan verifikasi secara berkala terhadap efektivitas keamanan kami sebagai bagian dari Program Kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon Lookout for Vision, lihat Layanan .
- Keamanan di cloud – Tanggung jawab Anda ditentukan menurut layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain termasuk sensitivitas data Anda, persyaratan perusahaan Anda, serta hukum dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Lookout for Vision. Topik berikut menunjukkan kepada Anda cara mengonfigurasi Lookout for Vision untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan layanan AWS lain yang membantu Anda memantau dan mengamankan Lookout for Vision.

Topik

Perlindungan data di Amazon Lookout for Vision

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di Amazon Lookout for Vision. Sebagaimana diuraikan dalam model ini, AWS bertanggung jawab untuk memberikan perlindungan terhadap infrastruktur global yang menjalankan semua AWS Cloud. Anda harus bertanggung jawab untuk memelihara kendali terhadap konten yang di-hosting pada infrastruktur ini. Anda juga bertanggung jawab atas tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat FAQ Privasi Data. Untuk informasi tentang perlindungan data di Eropa, silakan lihat postingan blog Model Tanggung Jawab Bersama AWS dan GDPR di Blog Keamanan AWS.

Untuk tujuan perlindungan data, sebaiknya Anda melindungi kredensial Akun AWS dan menyiapkan AWS IAM Identity Center atau AWS Identity and Access Management (IAM) untuk pengguna individu. Dengan cara seperti itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugas mereka. Kami juga merekomendasikan agar Anda mengamankan data Anda dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk melakukan komunikasi dengan sumber daya AWS. Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Siapkan API dan log aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi enkripsi AWS, bersama dengan semua kontrol keamanan default dalam Layanan AWS.
- Gunakan layanan keamanan terkelola lanjutan seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-2 ketika mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Untuk informasi selengkapnya tentang titik akhir FIPS yang tersedia, silakan lihat Standar Pemrosesan Informasi Federal (FIPS) 140-2.

Sebaiknya Anda tidak memasukkan informasi rahasia atau sensitif, seperti alamat email pelanggan, ke dalam tanda atau bidang teks bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Lookout for Vision atau Layanan AWS lainnya menggunakan konsol, APIAWS CLI, atau SDK. AWS Data apa pun yang Anda masukkan ke dalam tanda atau bidang teks bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau diagnostik. Saat Anda memberikan URL ke server eksternal, sebaiknya jangan menyertakan informasi kredensial di URL untuk memvalidasi permintaan Anda ke server tersebut.

Enkripsi data

Informasi berikut menjelaskan di mana Amazon Lookout for Vision menggunakan enkripsi data untuk melindungi data Anda.

Enkripsi saat tidak aktif

Citra

Untuk melatih model Anda, Amazon Lookout for Vision membuat salinan pelatihan sumber dan gambar uji Anda. Gambar yang disalin dienkripsi saat istirahat di Amazon Simple Storage Service (S3) Amazon Simple Storage Service (S3) menggunakan enkripsi sisi server dengan kunci atau kunci yang Anda berikan. Kunci milik AWS Kunci disimpan menggunakan AWS Key Management Service (SSE-KMS). Citra sumber Anda tidak terpengaruh. Untuk informasi selengkapnya, lihat Melatih model Anda.

Amazon Lookout for Vision model

Secara default, model terlatih dan file manifes dienkripsi di Amazon S3 menggunakan enkripsi sisi server dengan kunci KMS yang disimpan di AWS Key Management Service (SSE-KMS). Lookout for Vision menggunakan file. Kunci milik AWS Untuk informasi selengkapnya, lihat Melindungi Data Menggunakan Enkripsi Sisi Server. Hasil pelatihan ditulis ke bucket yang ditentukan dalam parameter input `output_bucket` ke `CreateModel`. Hasil pelatihan dienkripsi menggunakan pengaturan enkripsi yang dikonfigurasi untuk bucket (`output_bucket`).

Amazon Lookout for Vision konsol bucket

Konsol Amazon Lookout for Vision membuat bucket Amazon S3 (bucket konsol) yang dapat Anda gunakan untuk mengelola proyek. Bucket konsol tersebut dienkripsi menggunakan enkripsi Amazon S3 default. Untuk informasi selengkapnya, lihat Enkripsi default Layanan Penyimpanan Sederhana Amazon untuk bucket S3. Jika Anda menggunakan kunci KMS Anda sendiri, konfigurasikan bucket konsol tersebut setelah dibuat. Untuk informasi selengkapnya, lihat Melindungi Data Menggunakan Enkripsi Sisi Server. Amazon Lookout for Vision memblokir akses publik ke bucket konsol.

Enkripsi dalam transit

Titik akhir Amazon Lookout for Vision API hanya mendukung koneksi aman melalui HTTPS. Semua komunikasi dienkripsi dengan Keamanan Lapisan Pengangkutan (TLS).

Manajemen kunci

Anda dapat menggunakan AWS Key Management Service (KMS) untuk mengelola enkripsi gambar masukan yang Anda simpan di bucket Amazon S3. Untuk informasi selengkapnya, lihat Langkah 5: (Opsional) Menggunakan kunci AWS Key Management Service Anda sendiri.

Secara default gambar Anda dienkripsi dengan kunci yang dimiliki dan dikelola AWS. Anda juga dapat memilih untuk menggunakan kunci AWS Key Management Service (KMS) milik Anda sendiri. Untuk informasi selengkapnya, lihat Konsep AWS Key Management Service.

Privasi lalu lintas antarjaringan

Titik akhir Amazon Virtual Private Cloud (Amazon VPC) untuk Amazon Lookout for Vision adalah entitas logis dalam VPC yang memungkinkan konektivitas hanya ke Amazon Lookout for Vision. Amazon VPC merutekan permintaan ke Amazon Lookout for Vision dan merutekan respons kembali ke VPC. Untuk informasi selengkapnya, lihat VPC Endpoints dalam Panduan Pengguna Amazon VPC. Untuk informasi tentang menggunakan titik akhir Amazon VPC dengan Amazon Lookout for Vision, lihat. Akses Amazon Lookout for Vision menggunakan endpoint antarmuka (AWS PrivateLink)

Manajemen identitas dan akses untuk Amazon Lookout for Vision

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya Lookout for Vision. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- Audiens
- Mengautentikasi dengan identitas
- Mengelola akses menggunakan kebijakan
- Bagaimana Amazon Lookout for Vision bekerja dengan IAM
- Contoh kebijakan berbasis identitas Amazon Lookout for Vision
- AWSkebijakan terkelola untuk Amazon Lookout for Vision
- Memecahkan masalah identitas dan akses Amazon Lookout for Vision

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Lookout for Vision.

Pengguna layanan — Jika Anda menggunakan layanan Lookout for Vision untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak fitur Lookout for Vision untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di Lookout for Vision, lihat. Memecahkan masalah identitas dan akses Amazon Lookout for Vision

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya Lookout for Vision di perusahaan Anda, Anda mungkin memiliki akses penuh ke Lookout for Vision. Tugas Anda adalah menentukan fitur dan sumber daya Lookout for Vision mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep Basic IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM dengan Lookout for Vision, lihat. Bagaimana Amazon Lookout for Vision bekerja dengan IAM

Administrator IAM - Jika Anda seorang administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Lookout for Vision. Untuk melihat contoh kebijakan berbasis identitas Lookout for Vision yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas Amazon Lookout for Vision

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensyal identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensi yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (IAM Identity Center), autentikasi masuk tunggal perusahaan Anda, dan kredensi Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensil Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Menandatangani permintaan AWS API di Panduan Pengguna IAM.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat Autentikasi multi-faktor dalam Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) dalam AWS dalam Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas gabungan

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensi sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, direktori Pusat Identitas AWS Directory Service, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensil yang disediakan melalui sumber identitas. Ketika identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensi sementara.

Untuk manajemen akses terpusat, kami sarankan Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, lihat Apakah itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, kami merekomendasikan untuk mengandalkan kredensial sementara, bukan membuat pengguna IAM yang memiliki kredensial jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan tertentu yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami merekomendasikan Anda merotasi kunci akses. Untuk informasi selengkapnya, lihat Merotasi kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang dalam Panduan Pengguna IAM.

Grup IAM adalah identitas yang menentukan sekumpulan pengguna IAM. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup yang bernama IAMAdmins dan memberikan izin ke grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat Kapan harus membuat pengguna IAM (bukan peran) dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat mengambil peran IAM untuk sementara AWS Management Console dengan beralih peran. Anda dapat mengambil peran dengan memanggil operasi AWS CLI atau AWS API atau dengan menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, lihat Menggunakan peran IAM dalam Panduan Pengguna IAM.

Peran IAM dengan kredensial sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga dalam Panduan Pengguna IAM. Jika menggunakan Pusat Identitas IAM, Anda harus mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM akan mengorelasikan set izin ke peran dalam IAM. Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM guna mendapatkan berbagai izin secara sementara untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (prinsipal tepercaya) di akun lain untuk mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.
- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Sebagai contoh, ketika Anda memanggil suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses teruskan (FAS) — Saat Anda menggunakan pengguna IAM atau peran untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses maju.
  - Peran layanan – Peran layanan adalah peran IAM yang dijalankan oleh layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

  - Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan peran IAM untuk mengelola kredensi sementara untuk aplikasi yang berjalan pada instans EC2 dan membuat atau permintaan API. AWS CLI AWS Cara ini lebih dianjurkan daripada menyimpan kunci akses dalam instans EC2. Untuk menetapkan AWS peran ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 mendapatkan kredensial sementara. Untuk informasi selengkapnya, lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan dalam instans Amazon EC2 dalam Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, lihat Kapan harus membuat peran IAM (bukan pengguna) dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasinya. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan

tindakan `iam:GetRole`. Pengguna dengan kebijakan tersebut bisa mendapatkan informasi peran dari AWS Management Console, API AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan yang dikelola atau kebijakan inline, lihat Memilih antara kebijakan yang dikelola dan kebijakan inline dalam Panduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, lihat Gambaran umum daftar kontrol akses (ACL) dalam Panduan Developer Amazon Simple Storage Service.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas ke entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batasan izin, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) — SCP adalah kebijakan JSON yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di. AWS Organizations AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur di organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau semua akun Anda. SCP membatasi izin untuk entitas di akun anggota, termasuk masing-masing. Pengguna root akun AWS Untuk informasi selengkapnya tentang Organisasi dan SCP, lihat Cara kerja SCP dalam Panduan Pengguna AWS Organizations .
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana Amazon Lookout for Vision bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Lookout for Vision, pelajari fitur IAM yang tersedia untuk digunakan dengan Lookout for Vision.

Fitur IAM yang dapat Anda gunakan dengan Amazon Lookout for Vision

Fitur IAM	Dukungan Lookout for Vision
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
kunci-kunci persyaratan kebijakan (spesifik layanan)	Ya
ACL	Tidak
ABAC (tanda dalam kebijakan)	Parsial
Kredensial sementara	Ya
Sesi akses teruskan (FAS)	Ya
Peran layanan	Tidak
Peran terkait layanan	Tidak

Untuk mendapatkan tampilan tingkat tinggi tentang cara Lookout for Vision dan AWS layanan lainnya bekerja dengan sebagian besar fitur IAM, lihat AWS layanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Kebijakan berbasis identitas untuk Lookout for Vision

Mendukung kebijakan berbasis identitas	Ya

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, seperti pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan secara spesifik apakah tindakan dan sumber daya diizinkan atau ditolak, serta kondisi yang menjadi dasar dikabulkan atau ditolaknya tindakan tersebut. Anda tidak dapat menentukan secara spesifik prinsipal dalam sebuah kebijakan berbasis identitas karena prinsipal berlaku bagi pengguna atau peran yang melekat kepadanya. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk Lookout for Vision

Untuk melihat contoh kebijakan berbasis identitas Lookout for Vision, lihat. Contoh kebijakan berbasis identitas Amazon Lookout for Vision

Kebijakan berbasis sumber daya dalam Lookout for Vision

Mendukung kebijakan berbasis sumber daya	Tidak

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan secara spesifik seluruh akun atau entitas IAM di akun lain sebagai prinsipal dalam kebijakan berbasis sumber daya. Menambahkan prinsipal akun silang ke kebijakan berbasis sumber daya hanya setengah dari membangun hubungan kepercayaan. Ketika prinsipal dan sumber daya berbeda Akun AWS, administrator IAM di akun tepercaya juga harus memberikan izin entitas utama (pengguna atau peran) untuk mengakses sumber daya. Mereka memberikan izin dengan melampirkan kebijakan berbasis identitas kepada

entitas. Namun, jika kebijakan berbasis sumber daya memberikan akses ke prinsipal dalam akun yang sama, tidak diperlukan kebijakan berbasis identitas tambahan. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.

Tindakan kebijakan untuk Lookout for Vision

Mendukung tindakan kebijakan	Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan operasi AWS API terkait. Ada beberapa pengecualian, misalnya tindakan hanya izin yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Untuk melihat daftar tindakan Lookout for Vision, lihat Tindakan yang ditentukan oleh Amazon Lookout for Vision di Referensi Otorisasi Layanan.

Tindakan kebijakan di Lookout for Vision menggunakan awalan berikut sebelum tindakan:

```
lookoutvision
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [
      "lookoutvision:action1",
      "lookoutvision:action2"
         ]
```

Untuk melihat contoh kebijakan berbasis identitas Lookout for Vision, lihat. Contoh kebijakan berbasis identitas Amazon Lookout for Vision

Sumber daya kebijakan untuk Lookout for Vision

Mendukung sumber daya kebijakan	Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya Lookout for Vision dan ARNnya, lihat Sumber daya yang ditentukan oleh Amazon Lookout for Vision dalam Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh Amazon Lookout for Vision.

Untuk melihat contoh kebijakan berbasis identitas Lookout for Vision, lihat. Contoh kebijakan berbasis identitas Amazon Lookout for Vision

Kunci kondisi kebijakan untuk Lookout for Vision

Mendukung kunci kondisi kebijakan khusus layanan	Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke apa. Artinya, prinsipal manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan kondisi apa.

Elemen `Condition` (atau blok `Condition`) akan memungkinkan Anda menentukan kondisi yang menjadi dasar suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, maka AWS akan mengevaluasinya menggunakan operasi `AND` logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tag yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, lihat Elemen kebijakan IAM: variabel dan tag dalam Panduan Pengguna IAM.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan Pengguna IAM.

Untuk melihat daftar kunci kondisi Lookout for Vision, lihat Kunci kondisi untuk Amazon Lookout for Vision di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh Amazon Lookout for Vision.

Untuk melihat contoh kebijakan berbasis identitas Lookout for Vision, lihat. Contoh kebijakan berbasis identitas Amazon Lookout for Vision

ACL di Lookout for Vision

Mendukung ACL	Tidak

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

ABAC dengan Lookout for Vision

Mendukung ABAC (tanda dalam kebijakan)	Parsial

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Dalam AWS, atribut ini disebut tag. Anda dapat melampirkan tag ke entitas IAM (pengguna atau peran) dan ke banyak AWS sumber daya. Penandaan ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian rancanglah kebijakan ABAC untuk mengizinkan operasi ketika tag milik prinsipal cocok dengan tag yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna di situasi saat manajemen kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tag di elemen kondisi dari kebijakan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Apa itu ABAC? dalam Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) dalam Panduan Pengguna IAM.

Menggunakan kredensyal sementara dengan Lookout for Vision

Mendukung penggunaan kredensial sementara	Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensyal sementara. Untuk informasi tambahan, termasuk yang Layanan AWS bekerja dengan kredensi sementara, lihat Layanan AWS yang bekerja dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensi sementara jika Anda masuk AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda mengakses AWS menggunakan tautan masuk tunggal (SSO) perusahaan Anda, proses tersebut secara otomatis membuat kredensi sementara. Anda juga akan secara otomatis membuat kredensial sementara ketika Anda masuk ke konsol sebagai seorang pengguna lalu beralih peran. Untuk informasi selengkapnya tentang peralihan peran, lihat Peralihan peran (konsol) dalam Panduan Pengguna IAM.

Anda dapat membuat kredensyal sementara secara manual menggunakan API AWS CLI atau AWS . Anda kemudian dapat menggunakan kredensyal sementara tersebut untuk mengakses.

AWS AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensi sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM.

Teruskan sesi akses untuk Lookout for Vision

Mendukung sesi akses maju (FAS)	Ya

Saat Anda menggunakan pengguna atau peran IAM untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FAS menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Permintaan FAS hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses maju.

Peran layanan untuk Lookout for Vision

Mendukung peran layanan	Tidak

Peran layanan adalah sebuah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas Lookout for Vision. Edit peran layanan hanya jika Lookout for Vision memberikan panduan untuk melakukannya.

Peran terkait layanan untuk Lookout for Vision

Mendukung peran terkait layanan	Tidak

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang pembuatan atau manajemen peran terkait layanan, lihat Layanan AWS yang berfungsi dengan IAM. Cari layanan dalam tabel yang memiliki `Yes` di kolom Peran terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Contoh kebijakan berbasis identitas Amazon Lookout for Vision

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Lookout for Vision. Mereka juga tidak dapat melakukan tugas dengan menggunakan AWS Management Console, AWS Command Line Interface (AWS CLI), atau AWS API. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian akan dapat menambahkan kebijakan IAM ke peran, dan pengguna dapat mengambil peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Lookout for Vision, termasuk format ARN untuk setiap jenis sumber daya, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon Lookout for Vision dalam Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Mengakses satu proyek Amazon Lookout for Vision
- Contoh kebijakan berbasis tanda

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Lookout for Vision di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat Kebijakan yang dikelola AWS atau Kebijakan yang dikelola AWS untuk fungsi tugas dalam Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat Kebijakan dan izin dalam IAM dalam Panduan Pengguna IAM.
- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Kondisi dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan IAM Access Analyzer dalam Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi akses API yang dilindungi MFA dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan dalam IAM dalam Panduan Pengguna IAM.

Mengakses satu proyek Amazon Lookout for Vision

Dalam contoh ini, Anda ingin memberikan pengguna di AWS akun Anda akses ke salah satu proyek Amazon Lookout for Vision Anda.

```
{
    "Sid": "SpecificProjectOnly",
    "Effect": "Allow",
    "Action": [
        "lookoutvision:DetectAnomalies"
    ],
    "Resource": "arn:aws:lookoutvision:us-east-1:123456789012:model/myproject/*"
}
```

Contoh kebijakan berbasis tanda

Kebijakan berbasis tanda adalah dokumen kebijakan JSON yang menentukan tindakan yang dapat dilakukan oleh prinsip pada sumber daya yang ditandai.

Menggunakan tag untuk mengakses sumber daya

Kebijakan contoh ini memberikan izin kepada pengguna atau peran di akun AWS Anda untuk menggunakan `DetectAnomalies` operasi dengan model apa pun yang ditandai dengan kunci `stage` dan nilainya. `production`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "LookoutVision:DetectAnomalies"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "production"
                }
            }
        }
   ]
}
```

Menggunakan tag untuk menolak akses ke operasi Amazon Lookout for Vision tertentu

Kebijakan contoh ini menolak izin bagi pengguna atau peran di akun AWS Anda untuk memanggil `DeleteModel` atau `StopModel` operasi dengan model apa pun yang ditandai dengan kunci `stage` dan nilainya. `production`

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "LookoutVision:DeleteModel",
                "LookoutVision:StopModel"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "production"
                }
            }
        }
   ]
}
```

AWSkebijakan terkelola untuk Amazon Lookout for Vision

SebuahAWSkebijakan terkelola adalah kebijakan mandiri yang dibuat dan dikelola olehAWS.AWSkebijakan terkelola dirancang untuk memberikan izin untuk banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwaAWSkebijakan terkelola mungkin tidak memberikan izin paling sedikit hak istimewa untuk kasus penggunaan spesifik Anda karena tersedia untuk semuaAWSpelanggan untuk digunakan. Kami menyarankan Anda mengurangi izin lebih lanjut dengan mendefinisikankebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalamAWSkebijakan yang dikelola. JikaAWSmemperbarui izin yang didefinisikan dalamAWSkebijakan terkelola, pembaruan mempengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan

kebijakan.AWSkemungkinan besar akan memperbaruiAWSkebijakan terkelola saat baruLayanan AWSdiluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat Kebijakan terkelola AWS dalam Panduan Pengguna IAM.

Kebijakan yang dikelola AWS: AmazonLookoutVisionReadOnlyAccess

GunakanAmazonLookoutVisionReadOnlyAccesskebijakan untuk mengizinkan pengguna akses hanya-baca ke Amazon Lookout for Vision (dan dependensinya) dengan tindakan Amazon Lookout for Vision (operasi SDK) berikut. Misalnya, Anda dapat menggunakanDescribeModeluntuk mendapatkan informasi tentang model yang ada.

- DescribeDataset
- DescribeModel
- DescribeModelPackagingJob
- DescribeProject
- ListDatasetEntries
- ListModelPackagingJobs
- ListModels
- ListProjects
- ListTagsForResource

Untuk memanggil tindakan hanya-baca, pengguna tidak memerlukan izin bucket Amazon S3. Namun, respons operasi mungkin menyertakan referensi ke bucket Amazon S3. Misalnya,source-refentri dalam respon dariListDatasetEntriesadalah referensi ke gambar dalam bucket Amazon S3. Tambahkan izin bucket Amazon S3 jika pengguna Anda perlu mengakses bucket yang direferensikan. Misalnya, pengguna mungkin ingin mengunduh gambar yang direferensikan olehsource-refbidang. Untuk informasi selengkapnya, lihat Memberikan izin Bucket Amazon S3.

Anda dapat melampirkan kebijakan AmazonLookoutVisionReadOnlyAccess ke identitas-identitas IAM Anda.

Rincian perizinan

Kebijakan ini mencakup izin berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:DescribeDataset",
                "lookoutvision:DescribeModel",
                "lookoutvision:DescribeProject",
                "lookoutvision:DescribeModelPackagingJob",
                "lookoutvision:ListDatasetEntries",
                "lookoutvision:ListModels",
                "lookoutvision:ListProjects",
                "lookoutvision:ListTagsForResource",
                "lookoutvision:ListModelPackagingJobs"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan yang dikelola AWS:AmazonLookoutVisionFullAccess

Gunakan`AmazonLookoutVisionFullAccess`kebijakan untuk memungkinkan pengguna akses penuh ke Amazon Lookout for Vision (dan dependensinya) dengan tindakan Amazon Lookout for Vision (operasi SDK). Misalnya, Anda dapat melatih model tanpa harus menggunakan konsol Amazon Lookout for Vision. Untuk informasi selengkapnya, lihat Tindakan.

Untuk membuat dataset (`CreateDataset`) atau membuat model (`CreateModel`), pengguna Anda harus memiliki izin akses penuh ke bucket Amazon S3 yang menyimpan gambar set data, AmazonSageMakerGround Truth manifest file, dan output pelatihan. Untuk informasi selengkapnya, lihat Langkah 2: Siapkan izin.

Anda juga dapat memberikan izin untuk tindakan Amazon Lookout for Vision SDK dengan menggunakan`AmazonLookoutVisionConsoleFullAccess`kebijakan.

Anda dapat melampirkan kebijakan `AmazonLookoutVisionFullAccess` ke identitas-identitas IAM Anda.

Rincian perizinan

Kebijakan ini mencakup izin berikut.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionFullAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan yang dikelola AWS: AmazonLookoutVisionConsoleFullAccess

Gunakan`AmazonLookoutVisionFullAccess`kebijakan untuk memungkinkan pengguna akses penuh ke konsol Amazon Lookout for Vision, tindakan (operasi SDK), dan dependensi apa pun yang dimiliki layanan. Untuk informasi selengkapnya, lihat Memulai dengan Amazon Lookout for Vision.

Yang`LookoutVisionConsoleFullAccess`kebijakan mencakup izin ke bucket konsol Amazon Lookout for Vision Anda. Untuk informasi tentang bucket konsol, lihatLangkah 3: Buat bucket konsol. Untuk menyimpan kumpulan data, gambar, dan AmazonSageMakerFile manifes Ground Truth di bucket Amazon S3 yang berbeda, pengguna Anda memerlukan izin tambahan. Untuk informasi selengkapnya, lihat the section called “Menyetel izin bucket Amazon S3”.

Anda dapat melampirkan kebijakan `AmazonLookoutVisionConsoleFullAccess` ke identitas-identitas IAM Anda.

Pengelompokan izin

Kebijakan ini dikelompokkan ke dalam pernyataan berdasarkan kumpulan izin yang disediakan:

- `LookoutVisionFullAccess`- Memungkinkan akses untuk melakukan semua tindakan Lookout for Vision.
- `LookoutVisionConsoleS3BucketSearchAccess`— Memungkinkan daftar semua bucket Amazon S3 yang dimiliki oleh pemanggil. Lookout for Vision menggunakan tindakan ini untuk mengidentifikasi bucket konsol Lookout for Vision khusus Wilayah AWS, jika ada di akun pemanggil.
- `LookoutVisionConsoleS3BucketFirstUseSetupAccessPermissions`- Memungkinkan membuat dan mengonfigurasi bucket Amazon S3 yang cocok dengan pola nama bucket konsol Lookout for Vision. Lookout for Vision menggunakan tindakan ini untuk membuat dan mengonfigurasi bucket konsol Lookout for Vision khusus Wilayah saat tidak dapat menemukannya.
- `LookoutVisionConsoleS3BucketAccess`— Mengizinkan tindakan Amazon S3 yang bergantung pada bucket yang cocok dengan pola nama bucket konsol Lookout for Vision. Lookout untuk penggunaan Visi`s3:ListBucket`untuk mencari objek gambar saat membuat kumpulan data dari bucket Amazon S3 dan saat memulai tugas deteksi uji coba. Lookout untuk penggunaan Visi`s3:GetBucketLocation` dan`s3:GetBucketVersioning`untuk memvalidasi bucketAWSWilayah, pemilik, dan konfigurasi sebagai bagian dari berikut ini:
  - Membuat dataset
  - Melatih model
  - Memulai tugas deteksi percobaan
  - Melakukan umpan balik deteksi uji coba

  `LookoutVisionConsoleS3ObjectAccess`— Memungkinkan membaca dan menulis objek Amazon S3 di dalam bucket yang cocok dengan pola nama bucket Lookout for Vision Console. Lookout for Vision menggunakan tindakan ini untuk menampilkan gambar dalam tampilan galeri konsol dan untuk mengunggah gambar baru untuk digunakan dalam kumpulan data. Selain itu, izin ini memungkinkan Lookout for Vision untuk menuliskan metadata sambil membuat kumpulan data, melatih model, memulai tugas deteksi uji coba, dan melakukan umpan balik deteksi uji coba.
- `LookoutVisionConsoleDatasetLabelingToolsAccess`- Memungkinkan Amazon yang bergantungSageMaker GroundTruthtindakan pelabelan. Lookout for Vision menggunakan tindakan ini untuk memindai bucket S3 untuk gambar, buatGroundTruthfile manifes, dan untuk membubuhi anotasi hasil tugas deteksi percobaan dengan label validasi.

- `LookoutVisionConsoleDashboardAccess`- Memungkinkan membaca AmazonCloudWatchmetrik. Lookout for Vision menggunakan tindakan ini untuk mengisi grafik dasbor dan statistik anomali yang terdeteksi.
- `LookoutVisionConsoleTagSelectorAccess`- Memungkinkan membaca saran kunci tag dan nilai tag khusus akun. Lookout for Vision menggunakan izin ini untuk memberikan rekomendasi untuk kunci tag dan nilai tag dalamMengelola taghalaman konsol.
- `LookoutVisionConsoleKmsKeySelectorAccess`- Memungkinkan daftarAWS Key Management Service(KMS) kunci dan alias. Amazon Lookout for Vision menggunakan izin ini untuk mengisi kunci KMS dalam saranTagseleksi pada tindakan Lookout for Vision tertentu yang mendukung kunci KMS yang dikelola pelanggan untuk enkripsi.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionFullAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketSearchAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketFirstUseSetupAccess",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketVersioning",
                "s3:PutLifecycleConfiguration",
                "s3:PutEncryptionConfiguration",
                "s3:PutBucketPublicAccessBlock"
            ],
```

```
            "Resource": "arn:aws:s3:::lookoutvision-*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation",
                "s3:GetBucketAcl",
                "s3:GetBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*"
        },
        {
            "Sid": "LookoutVisionConsoleS3ObjectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*/*"
        },
        {
            "Sid": "LookoutVisionConsoleDatasetLabelingToolsAccess",
            "Effect": "Allow",
            "Action": [
                "groundtruthlabeling:RunGenerateManifestByCrawlingJob",
                "groundtruthlabeling:AssociatePatchToManifestJob",
                "groundtruthlabeling:DescribeConsoleJob"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleDashboardAccess",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        },
```

```
        {
            "Sid": "LookoutVisionConsoleTagSelectorAccess",
            "Effect": "Allow",
            "Action": [
                "tag:GetTagKeys",
                "tag:GetTagValues"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleKmsKeySelectorAccess",
            "Effect": "Allow",
            "Action": [
                "kms:ListAliases"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan yang dikelola AWS: AmazonLookoutVisionConsoleReadOnlyAkses

Gunakan`AmazonLookoutVisionConsoleReadOnlyAccess`kebijakan untuk mengizinkan pengguna akses hanya-baca ke konsol Amazon Lookout for Vision, tindakan (operasi SDK), dan dependensi apa pun yang dimiliki layanan.

Yang`AmazonLookoutVisionConsoleReadOnlyAccess`kebijakan mencakup izin Amazon S3 untuk bucket konsol Amazon Lookout for Vision. Jika gambar dataset Anda atau AmazonSageMakerFile manifes Ground Truth berada dalam bucket Amazon S3 yang berbeda, pengguna Anda memerlukan izin tambahan. Untuk informasi selengkapnya, lihat the section called “Menyetel izin bucket Amazon S3”.

Anda dapat melampirkan kebijakan `AmazonLookoutVisionConsoleReadOnlyAccess` ke identitas-identitas IAM Anda.

Pengelompokan izin

Kebijakan ini dikelompokkan ke dalam pernyataan berdasarkan kumpulan izin yang disediakan:

- `LookoutVisionReadOnlyAccess`- Memungkinkan akses untuk melakukan read-only Lookout untuk tindakan Vision.

- `LookoutVisionConsoleS3BucketSearchAccess`- Memungkinkan daftar semua bucket S3 yang dimiliki oleh pemanggil. Lookout for Vision menggunakan tindakan ini untuk mengidentifikasi bucket konsol Lookout for Vision khusus Wilayah AWS, jika ada satu di akun pemanggil.
- `LookoutVisionConsoleS3ObjectReadAccess`- Memungkinkan membaca objek Amazon S3 dan versi objek Amazon S3 di bucket konsol Lookout for Vision. Lookout for Vision menggunakan tindakan ini untuk menampilkan gambar dalam kumpulan data, model, dan deteksi uji coba.
- `LookoutVisionConsoleDashboardAccess`- Memungkinkan membaca AmazonCloudWatchmetrik. Lookout for Vision menggunakan tindakan ini untuk mengisi statistik untuk grafik dasbor dan anomali yang terdeteksi.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:DescribeDataset",
                "lookoutvision:DescribeModel",
                "lookoutvision:DescribeProject",
                "lookoutvision:DescribeTrialDetection",
                "lookoutvision:DescribeModelPackagingJob",
                "lookoutvision:ListDatasetEntries",
                "lookoutvision:ListModels",
                "lookoutvision:ListProjects",
                "lookoutvision:ListTagsForResource",
                "lookoutvision:ListTrialDetections",
                "lookoutvision:ListModelPackagingJobs"
            ],
            "Resource": "*"
        },
        {
            "Sid": "LookoutVisionConsoleS3BucketSearchAccess",
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "LookoutVisionConsoleS3ObjectReadAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::lookoutvision-*/*"
        },
        {
            "Sid": "LookoutVisionConsoleDashboardAccess",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricData",
                "cloudwatch:GetMetricStatistics"
            ],
            "Resource": "*"
        }
    ]
}
```

Lookout untuk pembaruan VisionAWSkebijakan terkelola

Lihat detail tentang pembaruanAWSkebijakan terkelola untuk Lookout for Vision sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman riwayat Lookout for Vision Document.

ditambahkan	menambahkan operasi pengemasan model berikut keAmazonLookoutVisionFullAccessdanAmazonLookoutVisionConsoleFullAccess kebijakan: • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob Amazon Lookout for Vision menambahkan operasi pengemasan model berikut keAmazonLookoutVisionReadOnlyAccessdanAmazonLookoutVisionConsoleReadOnlyAccesskebijakan: • DescribeModelPackagingJob • ListModelPackagingJobs	
Kebijakan baru ditambahkan	Amazon Lookout for Vision menambahkan kebijakan berikut. • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess • AmazonLookoutVisionConsoleReadOnlyAccess	11 Mei 2021
Lookout for Vision mulai melacak perubahan	Amazon Lookout for Vision mulai melacak perubahan	1 Maret, 2021

Perubahan	Deskripsi	Tanggal
Operasi pengemasan model ditambahkan	Amazon Lookout for Vision menambahkan operasi pengemasan model berikut keAmazonLookoutVisionFullAccessdanAmazonLookoutVisionConsoleFullAccess kebijakan: • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob Amazon Lookout for Vision menambahkan operasi pengemasan model berikut keAmazonLookoutVisionReadOnlyAccessdanAmazonLookoutVisionConsoleReadOnlyAccesskebijakan: • DescribeModelPackagingJob • ListModelPackagingJobs	7 Desember 2021
Kebijakan baru ditambahkan	Amazon Lookout for Vision menambahkan kebijakan berikut. • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess • AmazonLookoutVisionConsoleReadOnlyAccess nyaAWSkebijakan yang	11 Mei 2021

Memecahkan masalah identitas dan akses Amazon Lookout for Vision

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Lookout for Vision dan IAM.

Topik

Saya tidak berwenang untuk melakukan tindakan di Lookout for Vision

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `lookoutvision:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lookoutvision:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk pengguna `mateojackson` harus diperbarui untuk mengizinkan akses ke sumber daya *`my-example-widget`* dengan menggunakan tindakan `lookoutvision:`*`GetWidget`*.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan yang tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Lookout for Vision.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Lookout for Vision. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang-orang di luar saya Akun AWS untuk mengakses sumber daya Lookout for Vision saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mengetahui apakah Lookout for Vision mendukung fitur-fitur ini, lihat. Bagaimana Amazon Lookout for Vision bekerja dengan IAM
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.

Validasi kepatuhan untuk Amazon Lookout for Vision

Auditor pihak ketiga menilai keamanan dan kepatuhan Amazon Lookout for Vision sebagai bagian dari beberapa program kepatuhan. AWS Amazon Lookout for Vision mematuhi Peraturan Perlindungan Data Umum (GDPR).

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar AWS yang berfokus pada keamanan dan kepatuhan.
- Arsitektur untuk Keamanan dan Kepatuhan HIPAA di Amazon Web Services — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang memenuhi syarat HIPAA.

  > **Note**
  >
  > Tidak semua memenuhi Layanan AWS syarat HIPAA. Untuk informasi selengkapnya, lihat Referensi Layanan yang Memenuhi Syarat HIPAA.

- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.
- AWS Panduan Kepatuhan Pelanggan - Memahami model tanggung jawab bersama melalui lensa kepatuhan. Panduan ini merangkum praktik terbaik untuk mengamankan Layanan AWS dan memetakan panduan untuk kontrol keamanan di berbagai kerangka kerja (termasuk Institut Standar dan Teknologi Nasional (NIST), Dewan Standar Keamanan Industri Kartu Pembayaran (PCI), dan Organisasi Internasional untuk Standardisasi (ISO)).

- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— Ini Layanan AWS memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS. Security Hub menggunakan kontrol keamanan untuk sumber daya AWS Anda serta untuk memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik. Untuk daftar layanan dan kontrol yang didukung, lihat Referensi kontrol Security Hub.
- Amazon GuardDuty — Ini Layanan AWS mendeteksi potensi ancaman terhadap beban kerja Akun AWS, kontainer, dan data Anda dengan memantau lingkungan Anda untuk aktivitas mencurigakan dan berbahaya. GuardDuty dapat membantu Anda mengatasi berbagai persyaratan kepatuhan, seperti PCI DSS, dengan memenuhi persyaratan deteksi intrusi yang diamanatkan oleh kerangka kerja kepatuhan tertentu.
- AWS Audit ManagerIni Layanan AWS membantu Anda terus mengaudit AWS penggunaan Anda untuk menyederhanakan cara Anda mengelola risiko dan kepatuhan terhadap peraturan dan standar industri.

Amazon Lookout for Vision

Infrastruktur global AWS dibangun di sekitar Wilayah AWS dan Availability Zone. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan jaringan berlatensi rendah, throughput yang tinggi, dan sangat redundan. Dengan Availability Zone, Anda dapat mendesain dan mengoperasikan aplikasi dan basis data yang secara otomatis mengalami kegagalan di antara zona tanpa gangguan. Availability Zone lebih tersedia, memiliki toleransi kesalahan, dan dapat diskalakan dibandingkan dengan satu atau beberapa infrastruktur pusat data tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat Infrastruktur Global AWS.

Keamanan infrastruktur di Amazon Lookout for Vision

Sebagai layanan terkelola, Amazon Lookout for Vision dilindungi oleh keamanan jaringan AWS global. Untuk informasi tentang layanan AWS keamanan dan cara AWS melindungi infrastruktur, lihat Keamanan AWS Cloud. Untuk merancang AWS lingkungan Anda menggunakan praktik terbaik untuk

keamanan infrastruktur, lihat Perlindungan Infrastruktur dalam Kerangka Kerja Pilar Keamanan yang AWS Diarsiteksikan dengan Baik.

Anda menggunakan panggilan API yang AWS dipublikasikan untuk mengakses Lookout for Vision melalui jaringan. Klien harus mendukung hal berikut:

- Transport Layer Security (TLS). Kami membutuhkan TLS 1.2 dan merekomendasikan TLS 1.3.
- Suite cipher dengan kerahasiaan maju sempurna (PFS) seperti DHE (Ephemeral Diffie-Hellman) atau ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Sebagian besar sistem modern seperti Java 7 dan sistem yang lebih baru mendukung mode ini.

Selain itu, permintaan harus ditandatangani menggunakan access key ID dan secret access key yang terkait dengan principal IAM. Atau Anda bisa menggunakan AWS Security Token Service (AWS STS) untuk membuat kredensial keamanan sementara guna menandatangani permintaan.

Memantau Amazon Lookout for Vision

Pemantauan adalah bagian penting dari pemeliharaan keandalan, ketersediaan, dan kinerja Amazon Lookout for Vision dan solusi AWS lainnya. AWS menyediakan alat pemantauan berikut untuk mengawasi Lookout for Vision, melaporkan saat terjadi kesalahan, dan mengambil tindakan otomatis jika diperlukan:

- Amazon CloudWatch memantauAWS sumber daya Anda dan aplikasi yang Anda jalankanAWS di secara langsung. Anda dapat mengumpulkan dan melacak metrik, membuat dasbor yang disesuaikan, dan mengatur alarm yang memberi tahu Anda atau mengambil tindakan saat metrik tertentu mencapai ambang batas yang ditentukan. Misalnya, Anda dapat membuat CloudWatch melacak penggunaan CPU atau metrik lain dari instans Amazon EC2 Anda dan secara otomatis meluncurkan instans baru ketika diperlukan. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.
- Amazon CloudWatch Logs memungkinkan Anda memantau, menyimpan, dan mengakses file log Anda dari instans Amazon EC2, CloudTrail, dan sumber lainnya. CloudWatch Log dapat memantau informasi dalam file log dan memberi tahu Anda ketika ambang tertentu terpenuhi. Anda juga dapat mengarsipkan data log Anda dalam penyimpanan yang sangat tahan lama. Untuk informasi selengkapnya, lihat Panduan Pengguna Amazon CloudWatch Logs.
- Amazon EventBridge dapat digunakan untuk mengotomatiskanAWS layanan Anda dan merespons peristiwa sistem secara otomatis seperti masalah ketersediaan aplikasi atau perubahan sumber daya. Kejadian-kejadian dariAWS layanan dikirimkan EventBridge secara hampir waktu nyata. Anda dapat menuliskan aturan sederhana untuk menunjukkan peristiwa mana yang sesuai kepentingan Anda, dan tindakan otomatis mana yang diambil ketika suatu peristiwa sesuai dengan suatu aturan. Untuk informasi selengkapnya, lihat Panduan EventBridge Pengguna Amazon.
- AWS CloudTrail merekam panggilan API dan peristiwa terkait yang dilakukan oleh atau atas nama akun AWS Anda dan mengirimkan berkas log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun mana yang memanggil AWS, alamat IP sumber yang melakukan panggilan, dan kapan panggilan tersebut terjadi. Untuk mengetahui informasi selengkapnya, lihat Panduan Pengguna AWS CloudTrail.

Memantau Lookout for Vision dengan Amazon CloudWatch

Anda dapat memantau Lookout for Vision menggunakan CloudWatch, yang mengumpulkan data mentah dan memprosesnya menjadi metrik yang dapat dibaca dan hampir waktu nyata. Statistik

ini disimpan untuk jangka waktu 15 bulan, sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang performa aplikasi atau layanan web Anda. Anda juga dapat mengatur alarm yang memperhatikan ambang batas tertentu dan mengirim notifikasi atau mengambil tindakan saat ambang batas tersebut terpenuhi. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.

Layanan Lookout for Vision melaporkan metrik berikut di`AWS/LookoutVision` namespace.

Metrik	Deskripsi
`DetectedAnomalyCount`	Jumlah anomali yang terdeteksi dalam sebuah proyek Statistik yang valid:`Sum,Average` Unit: Jumlah
`ProcessedImageCount`	Jumlah total gambar yang dijalankan melalui deteksi anomali Statistik yang valid:`Sum,Average` Unit: Jumlah
`InvalidImageCount`	Jumlah gambar yang tidak valid dan tidak dapat mengembalikan hasil Statistik yang valid:`Sum,Average` Unit: Jumlah
`SuccessfulRequestCount`	Jumlah panggilan API yang berhasil Statistik yang valid:`Sum,Average` Unit: Jumlah
`ErrorCount`	Bilangan galat API Statistik yang valid:`Sum,Average`

Metrik	Deskripsi
	Unit: Jumlah
`ThrottledCount`	Jumlah error API yang disebabkan oleh throttling Statistik yang valid:`Sum,Average` Unit: Jumlah
`Time`	Waktu dalam hitungan milidetik untuk Lookout for Vision untuk mengomputasi deteksi anomali Statistik yang valid:`Data Samples,Average` Unit: Milidetik untuk`Average` statistik dan Hitung untuk`Data Samples` statistik
`MinInferenceUnits`	Jumlah unit inferensi minimum yang ditentukan selama`StartModel` permintaan. Statistik valid: `Average` Unit: Jumlah
`MaxInferenceUnits`	Jumlah maksimum unit inferensi yang ditentukan selama`StartModel` permintaan. Statistik valid: `Average` Unit: Jumlah
`DesiredInferenceUnits`	Jumlah unit inferensi yang Lookout for Vision meningkat atau turun. Statistik valid: `Average` Unit: Jumlah

Metrik	Deskripsi
`InServiceInferenceUnits`	Jumlah unit inferensi yang digunakan model. Statistik valid: `Average` Dianjurkan agar Anda menggunakan statistik Rata-rata untuk mendapatkan rata-rata 1 menit dari berapa banyak instance yang digunakan. Unit: Jumlah

Dimensi berikut didukung untuk metrik Lookout for Vision.

Dimensi	Deskripsi
`ProjectName`	Anda dapat membagi metrik berdasarkan proyek untuk melihat proyek mana yang mengalami masalah atau perlu diperbarui.
`ModelVersion`	Anda dapat membagi metrik berdasarkan versi model untuk melihat model mana yang mengalami masalah atau perlu diperbarui.

Logging Lookout untuk panggilan API Vision denganAWS CloudTrail

Amazon Lookout for Vision terintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atauAWS layanan di Lookout for Vision. CloudTrail merekam semua panggilan API untuk Lookout for Vision sebagai kejadian. Panggilan yang direkam mencakup panggilan dari konsol Lookout for Vision dan panggilan kode ke operasi Lookout for Vision API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail peristiwa berkelanjutan ke bucket Amazon S3, termasuk peristiwa untuk Lookout for Vision. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru dalam CloudTrail konsol di Riwayat kejadian. Menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan

permintaan yang dibuat ke Lookout for Vision, alamat IP asal permintaan tersebut dibuat, siapa yang membuat permintaan, kapan permintaan dibuat, dan detail lainnya.

Untuk mempelajari lebih lanjut CloudTrail, lihat PanduanAWS CloudTrail Pengguna.

Lookout for Vision CloudTrail

CloudTrail diaktifkan padaAWS akun Anda saat Anda membuat akun tersebut. Saat aktivitas terjadi di Lookout for Vision, aktivitas tersebut dicatat dalam CloudTrail kejadian bersama kejadianAWS layanan lainnya di Riwayat kejadian. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS Anda. Untuk informasi selengkapnya, lihat Melihat Kejadian dengan Riwayat CloudTrail Kejadian.

Untuk catatan berkelanjutan tentang peristiwa diAWS akun Anda, termasuk peristiwa untuk Lookout for Vision, buat jejak. Jejak CloudTrail memungkinkan pengiriman berkas log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak diterapkan ke semua Wilayah AWS. Jejak mencatat kejadian dari semua Wilayah di partisi AWS dan mengirimkan berkas log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasiAWS layanan lainnya untuk menganalisis lebih lanjut dan bertindak berdasarkan data kejadian yang dikumpulkan di CloudTrail log. Untuk informasi selengkapnya, lihat yang berikut:

- Ikhtisar untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengonfigurasi Notifikasi Amazon SNS untuk CloudTrail
- Menerima Berkas CloudTrail Log dari Beberapa Wilayah dan Menerima Berkas CloudTrail Log dari Beberapa Akun

Semua tindakan Lookout for Vision dicatat oleh CloudTrail dan didokumentasikan dalam dokumentasi referensi Lookout for Vision API. Misalnya, panggilan ke`CreateProject`,`DetectAnomalies` dan`StartModel` tindakan menghasilkan entri dalam file CloudTrail log.

Jika Anda memantau panggilan API Amazon Lookout for Vision, Anda mungkin melihat panggilan ke API berikut.

- lookoutvision:StartTriallDetection
- lookoutvision:ListTriallDetection
- lookoutvision:DescribeTrialDetection

Panggilan khusus ini digunakan oleh Amazon Lookout for Vision untuk mendukung berbagai operasi yang terkait dengan deteksi uji coba. Untuk informasi selengkapnya, lihat Memverifikasi model Anda dengan tugas deteksi uji coba.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut:

- Baik permintaan tersebut dibuat dengan kredensial pengguna root atau AWS Identity and Access Management.
- Baik permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan.
- Bahwa permintaan dibuat oleh layanan AWS lain.

Untuk informasi lain, lihat Elemen userIdentity CloudTrail .

Memahami Lookout untuk entri file log Vision

Jejak adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai berkas log ke bucket Amazon S3 yang Anda tentukan. CloudTrail berkas log berisi satu atau beberapa entri log. Sebuah peristiwa mewakili permintaan tunggal dari sumber apa pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail berkas log bukan jejak tumpukan terurut dari panggilan API publik, sehingga berkas tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan`CreateDataset` tindakan tersebut.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAYN4CJAYDEXAMPLE:user",
    "arn": "arn:aws:sts::123456789012:assumed-role/Admin/MyUser",
    "accountId": "123456789012",
    "accessKeyId": "ASIAYN4CJAYEXAMPLE",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAYN4CJAYDCGEXAMPLE",
```

```
        "arn": "arn:aws:iam::123456789012:role/Admin",
        "accountId": "123456789012",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-11-20T13:15:09Z"
      }
    }
  },
  "eventTime": "2020-11-20T13:15:43Z",
  "eventSource": "lookoutvision.amazonaws.com",
  "eventName": "CreateDataset",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "128.0.0.1",
  "userAgent": "aws-cli/3",
  "requestParameters": {
    "projectName": "P1",
    "datasetType": "train",
    "datasetSource": {
      "groundTruthManifest": {
        "s3Object": {
          "bucket": "myuser-bucketname",
          "key": "training.manifest"
        }
      }
    },
    "clientToken": "EXAMPLE-0526-47dd-a5d3-2ca975820a34"
  },
  "responseElements": {
    "status": "CREATE_IN_PROGRESS"
  },
  "requestID": "EXAMPLE-15e1-4bc9-be38-cda2537c75bf",
  "eventID": "EXAMPLE-c5e7-43e0-8449-8d9b87e15acb",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "123456789012"
}
```

Membuat sumber daya Amazon Lookout for VisionAWS CloudFormation

Amazon Lookout for Vision terintegrasi denganAWS CloudFormation, layanan yang membantu Anda memodelkan dan menyiapkanAWS sumber daya sehingga Anda dapat lebih cepat dalam membuat dan mengelola sumber daya dan infrastruktur Anda. Anda membuat templat yang menjelaskan semuaAWS sumber daya yang Anda inginkan, dan yang akanAWS CloudFormation mengurus penyediaan dan konfigurasi sumber daya tersebut untuk Anda.

Anda dapat menggunakannyaAWS CloudFormation untuk menyediakan dan mengonfigurasi proyek Amazon Lookout for Vision.

Saat menggunakanAWS CloudFormation, Anda dapat menggunakan kembali templat Anda untuk mengatur proyek Lookout for Vision secara konsisten dan berulang kali. Cukup jelaskan proyek Anda sekali dan kemudian sediakan proyek yang sama berulang-ulang dalam beberapa akun dan Wilayah AWS.

Lookout for Vision danAWS CloudFormation templat

Untuk menyediakan dan mengonfigurasi proyek untuk Lookout for Vision dan layanan terkait, Anda harus memahami AWS CloudFormationtemplat. Templat adalah file teks dengan format JSON atau YAML. Templat ini menjelaskan sumber daya yang ingin Anda sediakan di tumpukan AWS CloudFormation Anda. Jika Anda tidak terbiasa dengan JSON atau YAML, Anda dapat menggunakan AWS CloudFormation Designer untuk membantu Anda memulai dengan templat AWS CloudFormation. Untuk informasi selengkapnya, lihat Apa yang dimaksud dengan AWS CloudFormation Designer? dalam Panduan Pengguna AWS CloudFormation.

Untuk informasi referensi tentang proyek Lookout for Vision, termasuk contoh templat JSON dan YAKL, lihat referensi jenisLookoutVision sumber daya.

Pelajari selengkapnya tentang AWS CloudFormation

Untuk mempelajari selengkapnya tentang AWS CloudFormation, lihat sumber daya berikut:

- AWS CloudFormation
- AWS CloudFormationPanduan Pengguna

- AWS CloudFormationReferensi API
- AWS CloudFormationPanduan Pengguna Baris Perintah

Akses Amazon Lookout for Vision menggunakan endpoint antarmuka (AWS PrivateLink)

Anda dapat menggunakanAWS PrivateLink untuk membuat koneksi pribadi antara VPC dan Amazon Lookout for Vision. Anda dapat mengakses Lookout for Vision seolah-olah berada di VPC Anda, tanpa menggunakan gateway internet, atauAWS Direct Connect koneksi. Instans dalam VPC Anda tidak memerlukan alamat IP publik untuk mengakses Lookout for Vision.

Anda membuat koneksi pribadi ini dengan membuat endpoint antarmuka, didukung olehAWS PrivateLink. Kami membuat antarmuka jaringan endpoint di setiap subnet yang Anda aktifkan untuk endpoint antarmuka. Ini adalah antarmuka jaringan yang dikelola permintaan yang berfungsi sebagai titik masuk untuk lalu lintas yang ditujukan untuk Lookout for Vision.

Untuk informasi selengkapnya, lihat Mengakses Layanan AWS melalui AWS PrivateLink di Panduan AWS PrivateLink.

Pertimbangan untuk Lookout untuk titik akhir Visi VPC

Sebelum Anda menyiapkan titik akhir antarmuka untuk Lookout for Vision, tinjau Pertimbangan dalam AWS PrivateLinkPanduan.

Lookout for Vision mendukung pembuatan panggilan ke semua tindakan API melalui titik akhir antarmuka.

Kebijakan VPC endpoint tidak mendukung untuk Lookout for Vision. Secara default, akses penuh ke Lookout for Vision diizinkan melalui titik akhir antarmuka. Atau, Anda dapat mengaitkan grup keamanan dengan antarmuka jaringan titik akhir untuk mengontrol lalu lintas ke Lookout for Vision melalui titik akhir antarmuka.

Membuat VPC endpoint antarmuka untuk Lookout for Vision

Anda dapat membuat titik akhir antarmuka untuk Lookout for Vision menggunakan konsol Amazon VPC atauAWS Command Line Interface (AWS CLI). Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka di AWS PrivateLinkPanduan.

Buat titik akhir antarmuka untuk Lookout for Vision menggunakan nama layanan berikut:

```
com.amazonaws.region.lookoutvision
```

Jika Anda mengaktifkan DNS privat untuk titik akhir antarmuka, Anda dapat membuat permintaan API untuk Lookout for Vision menggunakan nama DNS defaultnya. Sebagai contoh, `lookoutvision.us-east-1.amazonaws.com`.

Membuat kebijakan VPC endpoint untuk Lookout for Vision

Kebijakan endpoint adalah sumber daya IAM yang dapat Anda lampirkan ke titik akhir antarmuka. Kebijakan endpoint default memungkinkan akses penuh ke Lookout for Vision melalui endpoint antarmuka. Untuk mengontrol akses yang diizinkan untuk Lookout for Vision dari VPC Anda, lampirkan kebijakan endpoint kustom ke endpoint antarmuka.

kebijakan titik akhir mencantumkan informasi berikut:

- Prinsipal yang dapat melakukan tindakan (Akun AWS, pengguna IAM, dan peran IAM).
- Tindakan-tindakan yang dapat dilakukan.
- Sumber daya untuk melakukan tindakan.

Untuk informasi selengkapnya, lihat Mengontrol akses ke layanan menggunakan kebijakan titik akhir di Panduan AWS PrivateLink.

Contoh: Kebijakan VPC endpoint untuk tindakan Lookout for Vision

Berikut ini adalah contoh kebijakan akhir kustom untuk Lookout for Vision. Ketika Anda melampirkan kebijakan ini ke titik akhir antarmuka, kebijakan menentukan bahwa semua pengguna yang memiliki akses ke titik akhir antarmuka VPC diizinkan untuk meminta operasi`DetectAnomalies` API untuk model Lookout for Vision yang`myModel` terkait dengan proyek`myProject`.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "lookoutvision:DetectAnomalies"
         ],
         "Resource": "arn:aws:lookoutvision:us-west-2:123456789012:model/myProject/
myModel"
```

```
        }
    ]
}
```

Kuota di Amazon Lookout for Vision

Tabel berikut menjelaskan kuota saat ini dalam Amazon Lookout for Vision. Untuk informasi tentang kuota yang dapat diubah, lihatKuota layanan AWS.

Kuota model

Kuota berikut berlaku untuk pengujian, pelatihan, dan fungsionalitas model.

Resource	Kuota
Format file yang didukung	Format gambar PNG dan JPEG
Dimensi gambar minimum file gambar dalam bucket Amazon S3	64 piksel x 64 piksel
Dimensi gambar maksimum file gambar dalam bucket Amazon S3	4096 piksel X 4096 piksel adalah maksimum. Dimensi yang lebih kecil dapat mengunggah lebih cepat.
Dimensi gambar yang berbeda dari file gambar yang digunakan dalam proyek	Semua gambar dalam dataset harus memiliki dimensi yang sama
Ukuran file maksimum untuk gambar dalam bucket Amazon S3	8 MB
Kurangnya label	Gambar harus diberi label normal atau anomali sebelum pelatihan. Gambar tanpa label diabaikan selama pelatihan.
Jumlah minimum gambar berlabel normal dalam kumpulan data pelatihan	10 untuk proyek dengan set data pelatihan dan pengujian terpisah. 20 untuk proyek dengan satu set data.
Jumlah minimum gambar berlabel anomali dalam kumpulan data pelatihan	0 untuk proyek dengan set data pelatihan dan pengujian terpisah. 10 untuk proyek dengan satu set data.

Resource	Kuota
Jumlah maksimum gambar dalam kumpulan data pelatihan klasifikasi	16.000
Jumlah maksimum gambar dalam kumpulan data uji klasifikasi	4.000
Jumlah minimum gambar berlabel normal dalam set data uji	10
Jumlah minimum gambar berlabel anomali dalam set data uji	10
Jumlah maksimum gambar dalam kumpulan data pelatihan pelokalan anomali	8000
Jumlah maksimum gambar dalam kumpulan data uji pelokalan anomali	800
Jumlah maksimum gambar dalam set data deteksi percobaan	2,000
Ukuran file manifes set data maksimum	1 GB
Jumlah maksimum kumpulan data pelatihan dalam model	1
Waktu latihan maksimal	24 jam
Waktu pengujian maksimum	24 jam
Jumlah maksimum label anomali dalam sebuah proyek	100
Jumlah maksimum label anomali pada gambar topeng	20

Resource	Kuota
Jumlah minimum gambar untuk label anomali. Untuk menghitung, gambar harus berisi hanya satu jenis label anomali.	20 untuk proyek kumpulan data tunggal. 10 untuk setiap kumpulan data dalam proyek dengan set data pelatihan dan pengujian terpisah.

Riwayat dokumen Amazon Lookout for Vision

Tabel berikut menjelaskan perubahan penting dalam setiap rilis Panduan Developer Amazon Lookout for Vision. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

- Update dokumentasi terbaru: 20 Februari 2023

Perubahan	Deskripsi	Tanggal
Ditambahkan contoh fungsi Lambda	Contoh yang menunjukkan bagaimana menemukan anomali dengan AWS Lambda fungsi. Untuk informasi selengkapnya, lihat Menemukan anomali dengan fungsi AWS Lambda.	20 Februari 2023 Februari 2023 Februari 2023 Februari 2023
Memperbarui panduan IAM untuk AWS WAF	Panduan yang diperbarui untuk menyelaraskan dengan praktik terbaik IAM. Untuk informasi selengkapnya, lihat Praktik terbaik keamanan di IAM.	8 Februari 2023 Februari 2023 Februari 2023 Februari 2023
Ditambahkan dataset ekspor contoh	Menambahkan contoh Python yang menunjukkan cara menggunakan `ListDatasetEntries` operasi untuk mengekspor kumpulan data dari proyek Amazon Lookout for Vision. Untuk informasi selengkapnya, lihat Mengekspor kumpulan data dari proyek (SDK).	2 Desember 2022 Desember 2022 Desember 2022 Desember 2020

Topik memulai yang diperbarui	Diperbarui memulai untuk menunjukkan membuat model segmentasi gambar dengan contoh dataset. Untuk informasi selengkapnya, lihat Memulai dengan Amazon Lookout for Vision.	20 Oktober 2022 Oktober 2022 Oktober 2022 Oktober 2022
Menambahkan topik pemecahan masalah	Menambahkan topik pemecahan masalah pelatihan model. Untuk informasi selengkapnya, lihat Pelatihan model pemecahan masalah.	17 Oktober 2022 Oktober 2022 Oktober 2022 Oktober 2022
Menambahkan topik tentang penggunaan pekerjaan Amazon SageMaker Ground Truth	Alih-alih memberi label pada gambar sendiri, Anda dapat menggunak an pekerjaan Amazon SageMaker Ground Truth untuk memberi label pada gambar untuk model klasifika si dan segmentasi gambar. Untuk informasi selengkapnya, lihat Menggunakan pekerjaan Amazon SageMaker Ground Truth.	19 Agustus 2022 Agustus 2022 Agustus 2022 Agustus 2020

Amazon Lookout for Vision sekarang menyediakan pelokalan anomali.	Anda dapat membuat model segmentasi yang menemukan lokasi pada gambar di mana berbagai jenis anomali (seperti goresan, penyok, atau sobek) hadir, label anomali dan ukuran anomali, Untuk informasi lebih lanjut, lihat Menjalankan model Amazon Lookout for Vision yang terlatih.	16 Agustus 2022 Agustus 2022 Agustus 2022 Agustus 2020
Amazon Lookout for Vision sekarang menyediakan inferensi CPU pada perangkat edge.	Model Amazon Lookout for Vision sekarang dapat digunakan untuk menjalank an inferensi secara lokal pada platform komputasi x86 yang menjalankan Linux hanya dengan CPU, tanpa memerlukan akselerator GPU. Untuk informasi selengkapnya, lihat Akselerator CPU.	16 Agustus 2022 Agustus 2022 Agustus 2022 Agustus 2020
Amazon Lookout for Vision sekarang dapat secara otomatis menskalakan unit inferensi.	Untuk membantu dengan lonjakan permintaan, Amazon Lookout for Vision sekarang dapat menskalakan jumlah unit inferensi yang digunakan model Anda. Untuk informasi selengkapnya, lihat Menjalank an model Amazon Lookout for Vision terlatih.	16 Agustus 2022 Agustus 2022 Agustus 2022 Agustus 2020

contoh Java ditambahkan	Panduan pengembang Amazon Lookout for Vision sekarang menyertakan contoh Java. Untuk informasi selengkapnya, lihat Memulai denganAWS SDK.	2 Mei 2022 Mei 2022 Mei 2022 Mei 2022
Ketersediaan umum penerapan model ke perangkat edge	Penerapan model ke perangkat edge yang dikelola oleh sekarangAWS IoT Greengrass Version 2 tersedia secara umum. Untuk informasi selengkapnya, lihat Menggunakan model Amazon Lookout for Vision di perangkat edge.	14 Maret 2022 Maret 2022 Maret 2022 Maret 2022
Informasi bucket konsol yang diperbarui	Informasi terbaru tentang konten bucket konsol dan pendekatan alternatif untuk membuat bucket konsol. Untuk informasi selengkapnya, lihat Langkah 4: Membuat bucket konsol.	7 Maret 2022 Maret 2022 Maret 2022 Maret 2022
Membuat file manifes dari file CSV	Anda sekarang dapat menyederhanakan pembuatan file manifes dengan menggunakan skrip yang membaca informasi klasifikasi dari file CSV. Untuk informasi selengkapnya, lihat Membuat file manifes dari file CSV.	10 Februari 2022 Februari 2022 Februari 2022 Februari 2022

Pratinjau rilis penerapan model ke perangkat edge	Rilis pratinjau penerapan model ke perangkat edge yang dikelola oleh sekarangAWS IoT Greengrass Version 2 tersedia. Untuk informasi selengkapnya, lihat Menggunakan model Amazon Lookout for Vision di perangkat edge.	Selasa, 07 Desember 2021
New Python dan Java 2 contoh ditambahkan	Ditambahkan Python dan Java 2 contoh untuk menganalisis gambar dengan`DetectAnomalies` . Untuk informasi selengkapnya, lihat Mendeteksi anomali dalam citra.	Selasa September 2021
KebijakanAWS terkelola baru ditambahkan.	Amazon Lookout for Vision menambahkan dukungan untuk kebijakan yangAWS dikelola. Untuk informasi selengkapnya, lihat kebijakan AWS terkelola untuk Amazon Lookout for Vision.	11 Mei 2021
Informasi unit inferensi yang diperbarui.	Menambahkan informasi yang menjelaskan unit inferensi dan bagaimana mereka dibebankan. Untuk informasi selengkapnya, lihat Menjalankan model Amazon Lookout for Vision terlatih.	15 Maret 2021

Ketersediaan umum untuk Amazon Lookout for Vision.	Amazon Lookout for Vision sekarang tersedia secara umum. Contoh kode Python diperbarui untuk menangani tugas asinkron seperti melatih model.	17 Februari 2021
Penandaan danAWS CloudFormation dukungan ditambahkan.	Sekarang Anda dapat menandai model Amazon Lookout for Vision dan membuat proyekAWS CloudFormation. Untuk informasi selengkapnya, lihat Menandai model dan Membuat proyek Amazon Lookout for Vision dengan AWS CloudFormation.	31 Januari 2021
Fitur dan panduan baru	Ini adalah rilis awal Amazon Lookout for Vision Amazon Lookout for Vision Panduan Developer.	1 Desember 2020

AWSGlosarium

Untuk AWS terminologi terbaru, lihat AWSglosarium di Referensi. Glosarium AWS

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.