Guida per gli sviluppatori

Amazon Lookout per Vision

Amazon Lookout per Vision: Guida per gli sviluppatori

Table of Contents

Che cos'è Amazon Lookout for Vision?

Puoi utilizzare Amazon Lookout for Vision per individuare i difetti visivi nei prodotti industriali, in modo accurato e su larga scala. È la visione artificiale per identificare componenti mancanti in un prodotto industriale, danni a veicoli o strutture, irregolarità nelle linee di produzione e persino difetti minuscoli nei sottili dischi di silicio o qualsiasi altro elemento fisico in cui la qualità è importante, come un condensatore mancante nei sottili dischi di silicio o qualsiasi altro elemento fisico in cui la qualità è importante, come la mancanza di un condensatore nei circuiti stampati.

Vantaggi principali:

Amazon Lookout per la visione fornisce rono rono rono rono rono rono rono rono rono rono rono rono

- Migliora i processi in modo rapido ed efficiente: puoi utilizzare Amazon Lookout for Vision per implementare l'ispezione basata sulla visione artificiale nei processi industriali in modo rapido ed efficiente, su larga scala. Puoi fornire solo 30 immagini di base valide e Lookout for Vision può creare automaticamente un modello ML personalizzato per il rilevamento dei difetti. È quindi possibile elaborare le immagini dalle telecamere IP, in batch o in tempo reale, per identificare in modo rapido e preciso anomalie come ammaccature, crepe e graffi.
- Aumenta rapidamente la qualità della produzione: con Amazon Lookout for Vision puoi ridurre i difetti nei processi di produzione, in tempo reale. Identifica e segnala le anomalie visive in una dashboard in modo da poter intervenire rapidamente per impedire che si verifichino altri difetti, aumentando la qualità della produzione e riducendo i costi.
- Riduci i costi operativi: Amazon Lookout for Vision segnala le tendenze nei dati delle ispezioni visive, ad esempio identificando i processi con la più alta percentuale di difetti o segnalando le variazioni recenti dei difetti. Utilizzando queste informazioni, è possibile determinare se pianificare la manutenzione sulla linea di processo o reindirizzare la produzione su un'altra macchina prima che si verifichino tempi di inattività costosi e non pianificati.

È la prima volta che utilizzi Amazon Lookout for Vision?

Se utilizzi Amazon Lookout for Vision prima volta, consigliamo di consultare le seguenti sezioni:

1. Configurazione di Amazon Lookout for Vision— In questa sezione, imposti i dettagli del tuo account.

2. Nozioni di base su Amazon Lookout for Vision— In questa sezione, scopri come creare il tuo primo modello Amazon Lookout for Vision.

Configurazione di Amazon Lookout for Vision

In questa sezione, crei un account AWS e configuri Amazon Lookout for Vision.

Per informazioni sulle AWS regioni che supportano Amazon Lookout for Vision, consulta Amazon Lookout for Vision Endpoints and Quotas.

Argomenti

Passaggio 1: Creare un account AWS

In questo passaggio, crei un AWS account e crei un utente amministrativo.

Argomenti

Registrarsi per creare un Account AWS

Se non disponi di un Account AWS, completa la procedura seguente per crearne uno.

Per registrarsi a un Account AWS

1. Apri la pagina all'indirizzo https://portal.aws.amazon.com/billing/signup.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

   Durante la registrazione di un Account AWS, viene creato un Utente root dell'account AWS. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWSnell'account. Come best

practice di sicurezza, assegna l'accesso amministrativo a un utente amministrativo e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

Al termine del processo di registrazione, riceverai un'e-mail di conferma da AWS. È possibile visualizzare l'attività corrente dell'account e gestire l'account in qualsiasi momento accedendo all'indirizzo https://aws.amazon.com/ e selezionando Il mio account.

Creazione di un utente amministratore

Dopo la registrazioneAccount AWS, proteggi Utente root dell'account AWSAWS IAM Identity Center, abilita e crea un utente amministrativo in modo da non utilizzare l'utente root per le attività quotidiane.

Protezione dell'Utente root dell'account AWS

1. Accedi alla AWS Management Console come proprietario dell'account scegliendo Utente root e immettendo l'indirizzo email del Account AWS. Nella pagina successiva, inserisci la password.

   Per informazioni sull'accesso utilizzando un utente root, consulta la pagina Accesso come utente root della Guida per l'utente di Accedi ad AWS.
2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

   Per ricevere istruzioni, consulta Abilitazione di un dispositivo MFA virtuale per l'utente root dell'Account AWS (console) nella Guida per l'utente IAM.

Creazione di un utente amministratore

1. Abilita IAM Identity Center.

   Per istruzioni, consulta Enabling AWS IAM Identity Center nella Guida AWS IAM Identity Center per l'utente.
2. In IAM Identity Center, concedi l'accesso amministrativo a un utente amministrativo.

   Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta Configurare l'accesso utente con le impostazioni predefinite IAM Identity Center directory nella Guida per l'AWS IAM Identity Centerutente.

Accesso come utente amministratore

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

  Per informazioni sull'accesso utilizzando un utente IAM Identity Center, consulta Accedere al portale di accesso AWS nella Guida per l'utente Accedi ad AWS.

Fase 2: Configurare le autorizzazioni

Per utilizzare Amazon Lookout for Vision, sono necessarie le autorizzazioni di accesso alla AWS console Lookout for Vision, alle operazioni SDK e al bucket Amazon S3 che usi per la formazione dei modelli.

> **Note**
>
> Se utilizzi solo le operazioni AWS SDK, puoi utilizzare policy relative alle operazioni SDK. AWS Per ulteriori informazioni, consulta Impostare le autorizzazioni dell'SDK.

Argomenti

- Impostazione dell'accesso alla console con policy gestite AWS
- Impostazione delle autorizzazioni per i bucket Amazon S3
- Assegnare le autorizzazioni

Impostazione dell'accesso alla console con policy gestite AWS

Utilizza le seguenti politiche AWS gestite per applicare le autorizzazioni di accesso appropriate per le operazioni della console Amazon Lookout for Vision e dell'SDK.

- AmazonLookoutVisionConsoleFullAccess— consente l'accesso completo alla console Amazon Lookout for Vision e alle operazioni SDK. Sono necessarie `AmazonLookoutVisionConsoleFullAccess` le autorizzazioni per creare il bucket della console. Per ulteriori informazioni, consulta Passaggio 3: Creare il bucket della console.
- AmazonLookoutVisionConsoleReadOnlyAccess— consente l'accesso in sola lettura alla console Amazon Lookout for Vision e alle operazioni SDK.

Per assegnare le autorizzazioni, consulta Assegnare le autorizzazioni.

Per informazioni sulle policy AWS gestite, consulta le policy gestite da AWS.

Impostazione delle autorizzazioni per i bucket Amazon S3

Amazon Lookout for Vision utilizza un bucket Amazon S3 per archiviare i seguenti file:

- Immagini del set di dati: immagini utilizzate per addestrare un modello. Per ulteriori informazioni, consulta Creare il tuo set di dati.
- File manifest in formato Amazon SageMaker Ground Truth. Ad esempio, il file manifesto generato dal SageMaker GroundTruth processo. Per ulteriori informazioni, consulta Creazione di un set di dati utilizzando un file manifest di Amazon SageMaker Ground Truth.
- L'output dell'addestramento del modello.

Se utilizzi la console, Lookout for Vision crea un bucket Amazon S3 (bucket console) per gestire i tuoi progetti. Le policy `LookoutVisionConsoleReadOnlyAccess` `LookoutVisionConsoleFullAccess` gestite includono le autorizzazioni di accesso di Amazon S3 per il bucket della console.

È possibile utilizzare il bucket della console per archiviare immagini del set di dati e file manifest in formato SageMaker Ground Truth. In alternativa, puoi utilizzare un bucket Amazon S3 diverso. Il bucket deve essere di proprietà del tuo account AWS e deve trovarsi nella AWS regione in cui utilizzi Lookout for Vision.

Per utilizzare un bucket diverso, aggiungi la seguente policy all'utente o al gruppo desiderato. Sostituisci `my-bucket` con il nome del bucket desiderato. Per informazioni sull'aggiunta di policy IAM, consulta Creazione di policy 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/*"
            ]
        }
    ]
}
```

Per assegnare le autorizzazioni, consulta Assegnare le autorizzazioni.

Assegnare le autorizzazioni

Per fornire l'accesso, aggiungi autorizzazioni ai tuoi utenti, gruppi o ruoli:

- Utenti e gruppi in AWS IAM Identity Center:

  Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina Create a permission set (Creazione di un set di autorizzazioni) nella Guida per l'utente di AWS IAM Identity Center.

- Utenti gestiti in IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate nella pagina Creating a role for a third-party identity provider (federation) (Creazione di un ruolo per un provider di identità di terze parti [federazione]) nella Guida per l'utente di IAM.

- Utenti IAM:
  - Crea un ruolo che l'utente possa assumere. Per istruzioni, consulta la pagina Creating a role for an IAM user (Creazione di un ruolo per un utente IAM) nella Guida per l'utente di IAM.
  - (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate nella pagina Aggiunta di autorizzazioni a un utente (console) nella Guida per l'utente di IAM.

Passaggio 3: Creare il bucket della console

Per utilizzare la console Amazon Lookout for Vision, è necessario un bucket Amazon S3 noto come bucket di console. Il bucket della console memorizza quanto segue:

- Immagini caricate in un set di dati con la console.
- Risultati dell'allenamento basato su modelli che inizi con la console.
- Risultati del rilevamento delle prove.
- File manifest temporanei che la console crea quando si utilizza la console per creare un set di dati etichettando automaticamente le immagini in un bucket S3. La console non elimina i file manifest.

Quando apri per la prima volta la console Amazon Lookout for Vision in una AWS nuova regione, Lookout for Vision crea il bucket di console per tuo conto. Prendi nota del nome del bucket della console perché potresti dover utilizzare il nome del bucket nelle operazioni dell'AWSSDK o nelle attività della console, come la creazione di un set di dati.

In alternativa, puoi creare il bucket della console utilizzando Amazon S3. Utilizza questo approccio se le policy dei bucket di Amazon S3 non consentono alla console Amazon Lookout for Vision di creare correttamente il bucket di console. Ad esempio, una policy che non consente la creazione automatica di un bucket Amazon S3.

> **Note**
>
> Se utilizzi solo l'AWSSDK e non la console Lookout for Vision, non è necessario creare il bucket della console. Puoi utilizzare un bucket S3 diverso con un nome a tua scelta.

<region><random value>Il formato del nome del bucket della console è `lookoutvision--`. Il valore casuale assicura che non vi sia un conflitto tra i nomi dei bucket.

Argomenti

- Creazione del bucket di console con la console Amazon Lookout for Vision
- Creazione del bucket di console con Amazon S3
- Impostazioni del bucket della console

Creazione del bucket di console con la console Amazon Lookout for Vision

Utilizza la seguente procedura per creare il bucket di console per una AWS regione con la console Amazon Lookout for Vision. Per informazioni sulle impostazioni del bucket S3 che abilitiamo, consulta. Impostazioni del bucket della console

Per creare il bucket della console utilizzando la console Amazon Lookout for Vision

1. Assicurati che l'utente o il gruppo che stai utilizzando `AmazonLookoutVisionConsoleFullAccess` disponga dell'autorizzazione. Per ulteriori informazioni, consulta Fase 2: Configurare le autorizzazioni.
2. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
3. Nella barra di navigazione, scegli Seleziona una regione. Quindi scegli la AWS regione per la quale desideri creare il bucket della console.
4. Scegliere Iniziare.
5. Se è la prima volta che apri la console nella regione AWS corrente, procedi come segue nella finestra di dialogo Prima configurazione:

   a. Copia il nome del bucket Amazon S3 indicato. Queste informazioni serviranno in seguito.

   b. Scegli Crea bucket S3 per consentire ad Amazon Lookout for Vision di creare il bucket della console per tuo conto.

   La finestra di dialogo di configurazione per la prima volta non viene visualizzata se il bucket di console per la regione corrente esiste già. AWS
6. Chiudi la finestra del browser.

Creazione del bucket di console con Amazon S3

Puoi usare Amazon S3 per creare il bucket della console. È necessario creare il bucket con il controllo delle versioni di Amazon S3 abilitato. Ti consigliamo di utilizzare una configurazione del ciclo di vita di Amazon S3 per rimuovere le versioni non correnti (precedenti) di un oggetto ed eliminare i caricamenti multiparte incompleti. Non consigliamo una configurazione del ciclo di vita che elimini le versioni correnti di un oggetto. Per informazioni sulle impostazioni dei bucket S3 che abilitiamo per i bucket di console creati con la console Amazon Lookout for Vision, consulta. Impostazioni del bucket della console

1. Decidi la AWS regione in cui desideri creare un bucket di console. Per informazioni sulle regioni supportate, consulta gli endpoint e le quote di Amazon Lookout for Vision.
2. Crea un bucket utilizzando le istruzioni della console S3 in Creazione di un bucket. Esegui questa operazione:

   a. Per il passaggio 3, specifica un nome di bucket preceduto da. `lookoutvision-`*`region`*`-`*`your-identifier` Passa `region` al codice regionale che hai scelto nel passaggio precedente. Passa `your-identifier` a un identificatore univoco di tua scelta. Ad esempio, `lookoutvision-us-east-1-my-console-bucket-1`

   b. Per il passaggio 4, scegli la AWS regione che desideri utilizzare.
3. Abilita il controllo delle versioni per il bucket seguendo le istruzioni della console S3 in Abilitare il controllo delle versioni sui bucket.
4. (Facoltativo) Specificate una configurazione del ciclo di vita per il bucket seguendo le istruzioni della console S3 in Impostazione della configurazione del ciclo di vita su un bucket. Effettua quanto segue per rimuovere le versioni non correnti (precedenti) di un oggetto ed eliminare i caricamenti multiparte incompleti. Non è necessario eseguire i passaggi 6, 8, 9, 10.

   a. Per il passaggio 5, scegli Applica a tutti gli oggetti nel bucket.

   b. Per il passaggio 7, seleziona Elimina definitivamente le versioni non correnti degli oggetti ed Elimina oggetti scaduti, i marker di eliminazione degli oggetti o i caricamenti multiparte incompleti.

   c. Per il passaggio 11, immettete il numero di giorni di attesa prima di eliminare le versioni non correnti di un oggetto.

   d. Per il passaggio 12, inserisci il numero di giorni di attesa prima di eliminare i caricamenti multiparte incompleti.

Impostazioni del bucket della console

Se crei il bucket di console con la console Amazon Lookout for Vision, abilitiamo le seguenti impostazioni sul bucket della console.

- Controllo delle versioni degli oggetti nel bucket della console.
- Crittografia lato server degli oggetti nel bucket della console.
- Una configurazione del ciclo di vita per l'eliminazione di oggetti non correnti (30 giorni) e i caricamenti incompleti in più parti (3 giorni).

- Blocca l'accesso pubblico al bucket della console.

Passaggio 4: Configurazione di AWS CLI e SDK AWS

I passaggi seguenti mostrano come installare AWS Command Line Interface (AWS CLI) e gli AWS SDK. Gli esempi in questa documentazione utilizzano gli AWS CLI SDK Python e JavaAWS.

Argomenti

- Installazione dell'SDK AWS
- Concessione dell'accesso programmatico
- Impostare le autorizzazioni dell’SDK
- Chiama un'azienda Amazon Lookout for Vision

Installazione dell'SDK AWS

Segui la procedura per scaricare e configurare gli SDK AWS.

Per installare e configurare AWS CLI e gli SDK AWS

- Scarica e installa AWS CLI e gli SDK AWS che desideri usare. Questa guida fornisce degli esempi per AWS CLI, Java e Python. Per informazioni sugli SDK AWS, consulta Strumenti per Amazon Web Services.

Concessione dell'accesso programmatico

Puoi eseguire AWS CLI e gli esempi di codice contenuti in questa guida sul tuo computer locale o in altri AWS ambienti, come un'istanza Amazon Elastic Compute Cloud. Per eseguire gli esempi, devi concedere l'accesso alle operazioni AWS SDK utilizzate dagli esempi.

Argomenti

- Eseguire il codice su un computer locale
- Esecuzione di codice in ambienti AWS

Eseguire il codice su un computer locale

Per eseguire il codice su un computer locale, ti consigliamo di utilizzare credenziali a breve termine per concedere a un utente l'accesso alle operazioni dell'AWSSDK. Per informazioni specifiche sull'esecuzione di AWS CLI e degli esempi di codice su un computer locale, consulta Utilizzo di un profilo su un computer locale.

Gli utenti hanno bisogno di un accesso programmatico se desiderano interagire con AWS esternamente a AWS Management Console. La modalità con cui concedere l'accesso programmatico dipende dal tipo di utente che accede ad AWS.

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporane e per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	Segui le istruzioni per l'interfa ccia che desideri utilizzare. • Per la AWS CLI, consulta la pagina Configurazione della AWS CLI per l'uso di AWS IAM Identity Center nella Guida per l'utente dell'AWS Command Line Interface. • Per gli SDK AWS, gli strumenti e le API AWS, consulta la pagina Autentica zione Centro identità IAM nella Guida di riferimento per SDK e strumenti AWS.
IAM	Utilizza credenziali temporane e per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	Segui le istruzioni in Utilizzo di credenziali temporanee con le risorse AWS nella Guida per l'utente IAM.

Quale utente necessita dell'accesso programmatico?	Per	Come
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, consulta la pagina Autenticazione tramite credenziali utente IAM nella Guida per l'utente dell'AWS Command Line Interface. • Per gli SDK e gli strumenti AWS, consulta la pagina Autenticazione con credenziali a lungo termine nella Guida di riferimento per SDK e strumenti AWS. • Per le API AWS, consulta la pagina Gestione delle chiavi di accesso per utenti IAM nella Guida per l'utente IAM.

Utilizzo di un profilo su un computer locale

È possibile eseguire AWS CLI e gli esempi di codice contenuti in questa guida con le credenziali a breve termine create in Eseguire il codice su un computer locale. Per ottenere le credenziali e altre informazioni sulle impostazioni, gli esempi utilizzano un profilo denominato `lookoutvision-access` Ad esempio:

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

L'utente rappresentato dal profilo deve disporre delle autorizzazioni per richiamare le operazioni dell'SDK Lookout for Vision e AWS altre operazioni SDK richieste dagli esempi. Per ulteriori informazioni, consulta Impostare le autorizzazioni dell'SDK. Per assegnare le autorizzazioni, consulta Assegnare le autorizzazioni.

Per creare un profilo che funzioni con AWS CLI e gli esempi di codice, scegli una delle seguenti opzioni. Assicurati che il nome del profilo che crei sia `lookoutvision-access`.

- Utenti gestiti da IAM — segui le istruzioni in Passaggio a un ruolo IAM (AWS CLI).
- Identità della forza lavoro (utenti gestiti da AWS IAM Identity Center) — segui le istruzioni in Configurazione dell'interfaccia a riga di comando di AWS CLI per usare AWS IAM Identity Center. Per gli esempi di codice, consigliamo di utilizzare un ambiente di sviluppo integrato (IDE), che supporta AWS Toolkit che abilita l'autenticazione tramite IAM Identity Center. Per gli esempi in Java, consulta Inizia a creare con Java. Per gli esempi in Python, consulta Inizia a creare con Python. Per ulteriori informazioni, consulta Credenziali IAM Identity.

> **Note**
>
> È possibile utilizzare il codice per ottenere credenziali a breve termine. Per ulteriori informazioni, consulta Passaggio a un ruolo IAM (AWS API). Per IAM Identity Center, ottieni le credenziali a breve termine per un ruolo seguendo le istruzioni in Ottenere le credenziali di ruolo IAM per l'accesso alla CLI.

Esecuzione di codice in ambienti AWS

Non è necessario utilizzare le credenziali utente per firmare chiamate AWS SDK in ambienti AWS Lambda, ad esempio codice di produzione in esecuzione in una funzione AWS. Al contrario, devi configurare un ruolo che definisce le autorizzazioni necessarie per il codice. Quindi assegnate il ruolo all'ambiente in cui viene eseguito il codice. Il modo in cui si assegna il ruolo e si rendono disponibili le credenziali temporanee varia a seconda dell'ambiente in cui viene eseguito il codice:

- AWS Lambdafunzione — utilizza le credenziali temporanee che Lambda fornisce automaticamente alla funzione quando assume il ruolo di esecuzione della funzione Lambda. Le credenziali sono disponibili nelle variabili di ambiente Lambda. Non è necessario specificare un profilo. Per ulteriori informazioni, consulta Ruolo di esecuzione Lambda.
- Amazon EC2 — utilizza il provider di credenziali endpoint per metadati delle istanze Amazon EC2. Il provider genera e aggiorna automaticamente le tue credenziali utilizzando il profilo dell'istanza Amazon EC2 che colleghi all'istanza Amazon EC2. Per ulteriori informazioni, consulta Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2
- Amazon Elastic Container Service — utilizza il provider di credenziali Container. Amazon ECS invia e aggiorna le credenziali a un endpoint di metadati. Un ruolo IAM dell'attività da te specificato

fornisce una strategia per la gestione delle credenziali utilizzate dall'applicazione. Per ulteriori informazioni, consulta la pagina relativa alla integrazione di con altri servizi AWS.

- Dispositivo core Greengrass: utilizza i certificati X.509 per connetterti a AWS IoT Core utilizzando i protocolli di autenticazione reciproca TLS. Questi certificati consentono ai dispositivi di interagire con AWS IoT senza credenziali AWS. Il provider di credenziali AWS IoT autentica i dispositivi utilizzando il certificato X.509 e rilascia le credenziali AWS sotto forma di token di sicurezza temporaneo con privilegi limitati. Per ulteriori informazioni, consulta la pagina relativa alla integrazione di con altri servizi AWS.

Per ulteriori informazioni sui provider di credenziali, consulta Fornitori di credenziali standardizzati.

Impostare le autorizzazioni dell'SDK

Per utilizzare le operazioni dell'SDK di Amazon Lookout for Vision, sono necessarie le autorizzazioni di accesso all'API Lookout for Vision e al bucket Amazon S3 utilizzato per la formazione dei modelli.

Argomenti

- Concessione delle autorizzazioni operative dell'SDK
- Concessione delle autorizzazioni per Amazon S3 Bucket
- Assegnare le autorizzazioni

Concessione delle autorizzazioni operative dell'SDK

Consigliamo pertanto di concedere solo le autorizzazioni richieste per eseguire un'attività (autorizzazioni con privilegio minimo). Ad esempio, per chiamare, è necessaria l'autorizzazione per DetectAnomalieseguire. `lookoutvision:DetectAnomalies` Per trovare le autorizzazioni per un'operazione, controlla il riferimento API.

Quando utilizzi un'applicazione per la prima volta, potresti non conoscere le autorizzazioni specifiche di cui hai bisogno, quindi puoi iniziare con autorizzazioni più ampie. Le policy gestite da AWS forniscono autorizzazioni per iniziare a utilizzare il prodotto.

- AmazonLookoutVisionFullAccess— consente l'accesso completo alle operazioni dell'SDK Amazon Lookout for Vision.
- AmazonLookoutVisionReadOnlyAccess— consente l'accesso alle operazioni SDK di sola lettura.

Le policy gestite per la console forniscono anche le autorizzazioni di accesso per le operazioni SDK. Per ulteriori informazioni, consulta Fase 2: Configurare le autorizzazioni.

Per informazioni sulle policy AWS gestite, consulta le policy gestite da AWS.

Quando conosci le autorizzazioni richieste dalla tua applicazione, riduci ulteriormente le autorizzazioni definendo le policy gestite dal cliente specifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta Policy gestite dal cliente .

> **Note**
>
> Le istruzioni per iniziare richiedono `s3:PutObject` autorizzazioni. Per ulteriori informazioni, consulta Fase 1: crea il file del manifesto e carica delle immagini.

Per assegnare le autorizzazioni, consulta Assegnare le autorizzazioni.

Concessione delle autorizzazioni per Amazon S3 Bucket

Per addestrare un modello, è necessario un bucket Amazon S3 con le autorizzazioni appropriate per archiviare le immagini, i file manifest e l'output di formazione. Il bucket deve essere di proprietà del tuo account AWS e deve trovarsi nella regione AWS in cui utilizzi Amazon Lookout for Vision.

Le policy gestite solo per SDK (`AmazonLookoutVisionFullAccesse` `AmazonLookoutVisionReadOnlyAccess`) non includono le autorizzazioni per i bucket Amazon S3 e devi applicare la seguente politica di autorizzazione per accedere ai bucket che usi, inclusi i bucket di console esistenti.

Le policy gestite dalla console (`AmazonLookoutVisionConsoleFullAccesse` `AmazonLookoutVisionConsoleReadOnlyAccess`) includono le autorizzazioni di accesso al bucket della console. Se si accede al bucket della console con operazioni SDK e si dispone delle autorizzazioni relative alle policy gestite dalla console, non è necessario utilizzare la seguente politica. Per ulteriori informazioni, consulta Fase 2: Configurare le autorizzazioni.

Decidere le autorizzazioni delle attività

Utilizza le seguenti informazioni per decidere quali autorizzazioni sono necessarie per le attività che desideri svolgere.

Creazione di un set di dati

Per creare un set di dati con CreateDataset, sono necessarie le seguenti autorizzazioni.

- `s3:GetBucketLocation`— consente a Lookout for Vision di confermare che il bucket si trova nella stessa regione in cui si utilizza Lookout for Vision.
- `s3:GetObject`— Consente l'accesso al file manifesto specificato nel parametro di input. `DatasetSource` Se si desidera specificare una versione esatta dell'oggetto S3 del file manifest, è necessario disporre `s3:GetObjectVersion` anche del file manifest. Per ulteriori informazioni, consulta Utilizzo del controllo delle versioni nei bucket S3.

Creare un modello

Per creare un modello con CreateModel, sono necessarie le seguenti autorizzazioni.

- `s3:GetBucketLocation`— consente a Lookout for Vision di confermare che il bucket si trova nella stessa regione in cui si utilizza Lookout for Vision.
- `s3:GetObject`— consente l'accesso alle immagini specificate nei set di dati di formazione e test del progetto.
- `s3:PutObject`— consente l'autorizzazione a memorizzare i risultati dell'allenamento nel bucket specificato. Si specifica la posizione del bucket di uscita nel `OutputConfig` parametro. Facoltativamente, è possibile limitare le autorizzazioni solo alle chiavi degli oggetti specificate nel `Prefix` campo del campo di input. `S3Location` Per ulteriori informazioni, vedere. OutputConfig

Accesso a immagini, file manifest e risultati di formazione

Le autorizzazioni per i bucket Amazon S3 non sono necessarie per visualizzare le risposte operative di Amazon Lookout for Vision. È necessaria `s3:GetObject` l'autorizzazione se si desidera accedere a immagini, file manifest e risultati di formazione a cui si fa riferimento nelle risposte operative.
Se stai accedendo a un oggetto Amazon S3 con versione, hai `s3:GetObjectVersion` bisogno dell'autorizzazione.

Impostazione della policy per i bucket di Amazon S3

Puoi utilizzare la seguente policy per specificare le autorizzazioni del bucket Amazon S3 necessarie per creare un set di dati (`CreateDataset`), creare un modello (`CreateModel`) e accedere a immagini, file manifest e output di formazione. Cambia il valore di *`my-bucket con il nome del bucket`* che desideri utilizzare.

Puoi adattare la politica alle tue esigenze. Per ulteriori informazioni, consulta Decidere le autorizzazioni delle attività. Aggiungi la politica all'utente desiderato. Per ulteriori informazioni, consulta Creazione di politiche 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"
                }
            }
        }
    ]
}
```

Per assegnare le autorizzazioni, consulta Assegnare le autorizzazioni.

Assegnare le autorizzazioni

Per fornire l'accesso, aggiungi autorizzazioni ai tuoi utenti, gruppi o ruoli:

- Utenti e gruppi in AWS IAM Identity Center:

  Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina Create a permission set (Creazione di un set di autorizzazioni) nella Guida per l'utente di AWS IAM Identity Center.

- Utenti gestiti in IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate nella pagina Creating a role for a third-party identity provider (federation) (Creazione di un ruolo per un provider di identità di terze parti [federazione]) nella Guida per l'utente di IAM.

- Utenti IAM:
  - Crea un ruolo che l'utente possa assumere. Per istruzioni, consulta la pagina Creating a role for an IAM user (Creazione di un ruolo per un utente IAM) nella Guida per l'utente di IAM.
  - (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate nella pagina Aggiunta di autorizzazioni a un utente (console) nella Guida per l'utente di IAM.

Chiama un'azienda Amazon Lookout for Vision

Esegui il codice seguente per confermare che puoi effettuare chiamate all'API Amazon Lookout for Vision. Il codice elenca i progetti presenti nel tuo account AWS, nella regione AWS corrente. Se non hai mai creato un progetto in precedenza, la risposta è vuota, ma conferma che puoi chiamare l'operazione `ListProjects`.

In generale, la chiamata di una funzione di esempio richiede un client AWS SDK Lookout for Vision e qualsiasi altro parametro richiesto. Il client AWS SDK Lookout for Vision è dichiarato nella funzione principale.

Se il codice fallisce, verifica che l'utente che utilizzi disponga delle autorizzazioni corrette. Verifica inoltre che la AWS regione che utilizzi come Amazon Lookout for Vision non sia disponibile in AWS tutte le regioni.

Per chiamare un'azienda Amazon Lookout for Vision

1. Se non lo si è ancora fatto, installare e configurare l'SDK AWS CLI e AWS Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il seguente codice di esempio per visualizzare i tuoi progetti.

   CLI

   Usa il comando `list-projects` per elencare i progetti nel tuo account.

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

   Python

   Questo codice è tratto dal repository degli esempi GitHub di AWS Documentation SDK. Guarda l'esempio completo qui.

   ```
   # 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

Questo codice è tratto dal repository degli esempi di AWS Documentation SDK. GitHub Guarda l'esempio completo qui.

```
/*
   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);
        }

    }

}
```

Fase 5: (Facoltativo) Utilizzo della propria chiave AWS Key Management Service

Puoi utilizzare AWS Key Management Service (KMS) per gestire la crittografia delle immagini di input archiviate nei bucket Amazon S3.

Per impostazione predefinita, le immagini sono crittografate con una chiave di proprietà e gestione di AWS. Puoi anche scegliere di utilizzare la tua chiave AWS Key Management Service (KMS) personale. Per ulteriori informazioni, consulta Concetti di AWS Key Management Service.

Se desideri utilizzare la tua chiave KMS, utilizza la seguente policy per specificare la chiave KMS. Cambia *kms_key_arn* nell'ARN della chiave KMS (o alias KMS ARN) che desideri utilizzare. In

alternativa, specifica di utilizzare qualsiasi chiave KMS. * Per informazioni sull'aggiunta della policy a un utente o a un ruolo, consulta Creazione di politiche 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"
                }
            }
        }
    ]
}
```

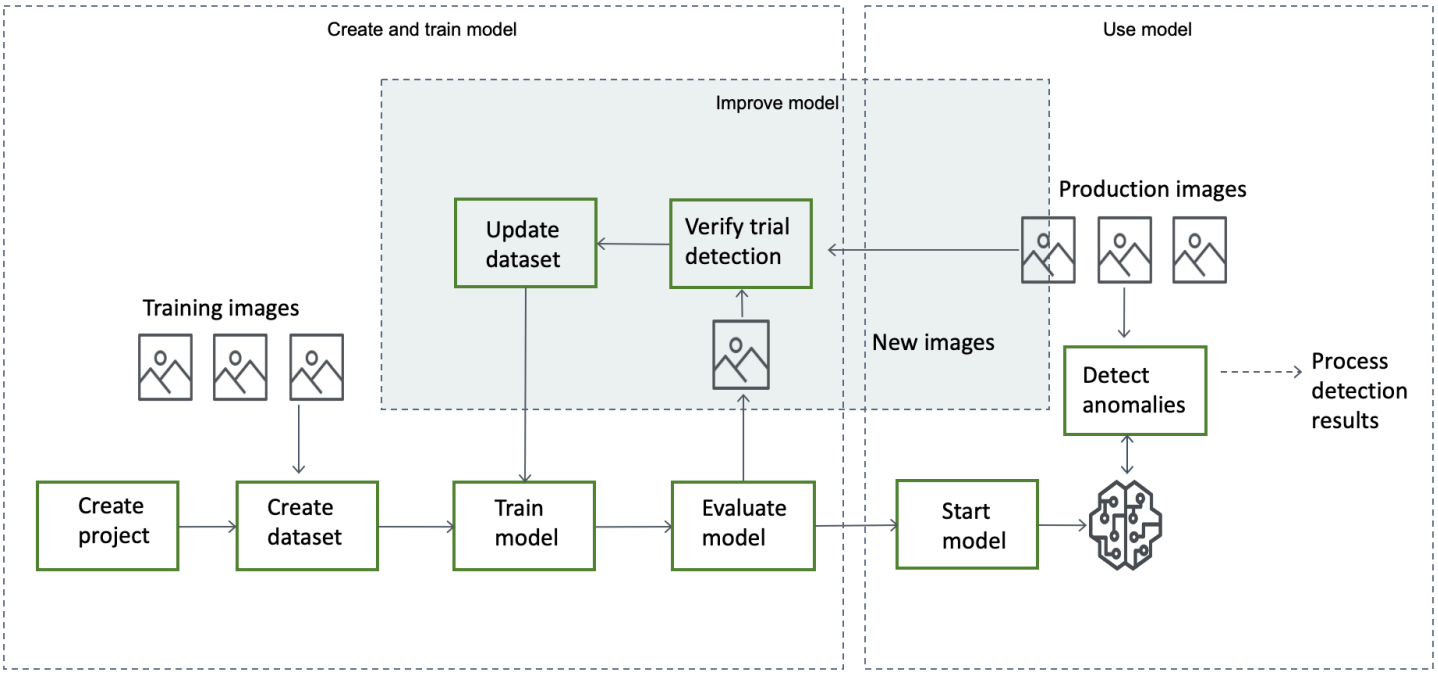
Comprendere Amazon Lookout for Vision

Puoi utilizzare Amazon Lookout for Vision per individuare difetti visivi nei prodotti industriali, in modo accurato e su larga scala, per attività quali:

- Rilevamento di parti danneggiate: individua i danni alla qualità, al colore e alla forma della superficie di un prodotto durante il processo di fabbricazione e assemblaggio.
- Identificazione dei componenti mancanti: determina i componenti mancanti in base all'assenza, alla presenza o al posizionamento degli oggetti. Ad esempio, un condensatore mancante su un circuito stampato.
- Individuazione dei problemi di processo: rileva i difetti con schemi ripetuti, ad esempio graffi ripetuti nello stesso punto su un wafer di silicone.

Con Lookout for Vision crei un modello di visione artificiale che prevede la presenza di anomalie in un'immagine. Fornisci le immagini che Amazon Lookout for Vision utilizza per addestrare e testare il tuo modello. Amazon Lookout for Vision fornisce metriche che puoi utilizzare per valutare e migliorare il tuo modello addestrato. Puoi ospitare il modello addestrato nelAWS cloud oppure puoi distribuirlo su un dispositivo edge. Una semplice operazione API restituisce le previsioni effettuate dal modello.

Il flusso di lavoro generale per la creazione, la valutazione e l'utilizzo di un modello è il seguente:



Argomenti

Seleziona il tuo tipo di modello

Prima di poter creare un modello, è necessario decidere il tipo di modello desiderato. È possibile creare due tipi di modello, classificazione delle immagini e segmentazione delle immagini. Sei tu a decidere quale tipo di modello creare in base al tuo caso d'uso.

Modello di classificazione delle immagini

Se hai solo bisogno di sapere se un'immagine contiene un'anomalia, ma non hai bisogno di conoscerne la posizione, crea un modello di classificazione delle immagini. Un modello di classificazione delle immagini prevede se un'immagine contiene un'anomalia. La previsione include la fiducia del modello nell'accuratezza della previsione. Il modello non fornisce alcuna informazione sulla posizione di eventuali anomalie riscontrate nell'immagine.

Modello di segmentazione delle immagini

Se hai bisogno di conoscere la posizione di un'anomalia, ad esempio la posizione di un graffio, crea un modello di segmentazione dell'immagine. I modelli Amazon Lookout for Vision utilizzano la segmentazione semantica per identificare i pixel di un'immagine in cui sono presenti i tipi di anomalie (come un graffio o una parte mancante).

> **Note**
>
> Un modello di segmentazione semantica individua diversi tipi di anomalia. Non fornisce informazioni di istanza per le singole anomalie. Ad esempio, se un'immagine contiene due ammaccature, Lookout for Vision restituisce informazioni su entrambe le ammaccature in un'unica entità che rappresenta il tipo di anomalia dell'ammaccatura.

Un modello di segmentazione di Amazon Lookout for Vision prevede quanto segue:

Classificazione

Il modello restituisce una classificazione per un'immagine analizzata (normale/anomalia), che include la fiducia del modello nella previsione. Le informazioni di classificazione vengono calcolate separatamente dalle informazioni sulla segmentazione e non si deve presumere una relazione tra di esse.

Segmentazione

Il modello restituisce una maschera di immagine che contrassegna i pixel in cui si verificano anomalie sull'immagine. Diversi tipi di anomalia sono codificati a colori in base al colore assegnato all'etichetta dell'anomalia nel set di dati. Un'etichetta di anomalia rappresenta il tipo di anomalia. Ad esempio, la maschera blu nell'immagine seguente indica la posizione di un tipo di anomalia da graffio riscontrata su un'auto.

Il modello restituisce il codice colore per ogni etichetta di anomalia nella maschera. Il modello restituisce anche la percentuale di copertura dell'immagine che ha un'etichetta di anomalia.

Con un modello di segmentazione Lookout for Vision, puoi utilizzare vari criteri per analizzare i risultati dell'analisi del modello. Ad esempio:

- Localizzazione delle anomalie: se hai bisogno di conoscere la posizione delle anomalie, utilizza le informazioni di segmentazione per visualizzare le maschere che coprono le anomalie.
- Tipi di anomalia: utilizza le informazioni di segmentazione per decidere se un'immagine contiene più di un numero accettabile di tipi di anomalia.
- Area di copertura: utilizza le informazioni di segmentazione per decidere se un tipo di anomalia copre più di un'area accettabile di un'immagine.
- Classificazione delle immagini: se non è necessario conoscere la posizione delle anomalie, utilizza le informazioni di classificazione per determinare se un'immagine contiene anomalie.

Per il codice di esempio, consulta Rilevamento di anomalie in un'immagine.

Dopo aver deciso il tipo di modello desiderato, si creano un progetto e un set di dati per gestire il modello. Utilizzando le etichette, è possibile classificare le immagini come normali o come anomalie. Le etichette identificano anche le informazioni di segmentazione come maschere e tipi di anomalia. Il modo in cui etichettate le immagini nel set di dati determina il tipo di modello che Lookout for Vision crea per voi.

L'etichettatura di un modello di segmentazione delle immagini è più complessa dell'etichettatura di un modello di classificazione delle immagini. Per addestrare un modello di segmentazione, è necessario classificare le immagini di addestramento come normali o anomale. È inoltre necessario definire maschere di anomalia e tipi di anomalia per ogni immagine anomala. Un modello di classificazione richiede solo di identificare le immagini di addestramento come normali o anomale.

Creazione del tuo modello

I passaggi per creare un modello sono la creazione di un progetto, la creazione di un set di dati e l'addestramento del modello sono i seguenti:

Crea un progetto

Crea un progetto per gestire i set di dati e i modelli che crei. Un progetto deve essere utilizzato per un singolo caso d'uso, ad esempio per rilevare anomalie in un singolo tipo di parte della macchina.

presente nella dashboard di controllo di controllo di controllo di controllo di controllo di controllo di controllo di controllo di controllo di controllo di controllo di controllo di controllo Per ulteriori informazioni, consulta Utilizzo Lookout for Vision di.

Ulteriori informazioni: Crea il tuo progetto.

Creazione di un set di dati

Per addestrare un modello Amazon Lookout for Vision ha bisogno di immagini di oggetti normali e anomali per il tuo caso d'uso. Fornisci queste immagini in un set di dati.

Un set di dati è un insieme di immagini ed etichette che descrivono tali immagini. Le immagini devono rappresentare un singolo tipo di oggetto su cui possono verificarsi anomalie. Per ulteriori informazioni, consulta Preparazione delle immagini per un set di dati.

Con Amazon Lookout for Vision puoi avere un progetto che utilizza un singolo set di dati o un progetto con set di dati di formazione e test separati. Si consiglia di utilizzare un singolo progetto di set di dati a meno che non si desideri un controllo più preciso su formazione, test e ottimizzazione delle prestazioni.

Si crea un set di dati importando le immagini. A seconda di come si importano le immagini, è possibile che anche le immagini siano etichettate. In caso contrario, si utilizza la console per etichettare le immagini.

Importazione di immagini

Se crei il set di dati con la console Lookout for Vision, puoi importare le immagini in uno dei modi seguenti:

- Importa immagini dal tuo computer locale. Le immagini non sono etichettate.
- Importa immagini da un bucket S3. Amazon Lookout for Vision può classificare le immagini utilizzando i nomi delle cartelle che le contengono. Utilizzare `normal` per immagini normali. Da utilizzare `anomaly` per immagini anomale. Non è possibile assegnare automaticamente etichette di segmentazione.
- Importa un file di manifesto Amazon SageMaker Ground Truth. Le immagini in un file manifest sono etichettate. Puoi creare e importare il tuo file manifest. Se hai molte immagini, prendi in considerazione l'utilizzo del servizio di etichettatura SageMaker Ground Truth. Quindi importi il file del manifesto di output dal job Amazon SageMaker Ground Truth.

Labeling delle immagini

Le etichette descrivono un'immagine in un set di dati. Le etichette specificano se un'immagine è normale o anomala (classificazione). Le etichette descrivono anche la posizione delle anomalie su un'immagine (segmentazione).

Se le tue immagini non sono etichettate, puoi usare la console per etichettarle.

Le etichette assegnate alle immagini nel set di dati determinano il tipo di modello creato da Lookout for Vision:

Classificazione delle immagini

Per creare un modello di classificazione delle immagini, utilizza la console Lookout for Vision per classificare le immagini nel set di dati come normali o come anomalie.

È inoltre possibile utilizzare l'`CreateDataset`operazione per creare un set di dati da un file di manifesto che include informazioni di classificazione.

Segmentazione delle immagini

Per creare un modello di segmentazione delle immagini, utilizza la console Lookout for Vision per classificare le immagini nel set di dati come normali o come anomalie. Specificate anche maschere di pixel per le aree anomale dell'immagine (se esistono) e un'etichetta di anomalia per le singole maschere di anomalia.

È inoltre possibile utilizzare l'`CreateDataset`operazione per creare un set di dati da un file di manifesto che include informazioni di segmentazione e classificazione.

Se il progetto ha set di dati di formazione e test separati, Lookout for Vision utilizza il set di dati di formazione per apprendere e determinare il tipo di modello. Dovresti etichettare le immagini nel set di dati di test allo stesso modo.

Ulteriori informazioni: Creazione del set di dati.

Addestra il tuo modello

La formazione crea un modello e lo addestra a prevedere la presenza di anomalie nelle immagini. Ogni volta che ti alleni, viene creata una nuova versione del modello.

All'inizio della formazione, Amazon Lookout for Vision sceglie l'algoritmo più adatto con cui addestrare il modello. Il modello viene addestrato e quindi testato. InNozioni di base su Amazon Lookout for Vision, si addestra un singolo progetto di set di dati, il set di dati viene suddiviso internamente per creare un set di dati di addestramento e un set di dati di test. Puoi anche creare un progetto con set di dati di formazione e test separati. In questa configurazione, Amazon Lookout for Vision addestra il modello con il set di dati di addestramento e testa il modello con il set di dati di test.

> **Important**
>
> Ti viene addebitato il tempo necessario per addestrare correttamente il tuo modello.

Ulteriori informazioni: Addestra il tuo modello.

Valutazione del tuo modello

Valuta le prestazioni del tuo modello utilizzando le metriche delle prestazioni create durante i test.

Utilizzando le metriche delle prestazioni, puoi comprendere meglio le prestazioni del tuo modello addestrato e decidere se sei pronto per utilizzarlo in produzione.

Ulteriori informazioni: Miglioramento del modello.

Se le metriche delle prestazioni indicano che sono necessari miglioramenti, puoi aggiungere altri dati di allenamento eseguendo un'attività di rilevamento delle prove con nuove immagini. Una volta completata l'attività, puoi verificare i risultati e aggiungere le immagini verificate al set di dati di allenamento. In alternativa, puoi aggiungere nuove immagini di allenamento direttamente al set di dati. Successivamente, riaddestrate il modello e ricontrollate le metriche delle prestazioni.

Ulteriori informazioni: verifica del modello con un'attività di rilevamento di prova.

Usa il tuo modello

Prima di poter utilizzare il modello nelAWS cloud, avviate il modello con l'StartModeloperazione. Puoi ottenere il comando`StartModel` CLI per il tuo modello dalla console.

Ulteriori informazioni: Avvia il tuo modello.

Un modello esperto di Amazon Lookout for Vision prevede se un'immagine di input contiene contenuti normali o anomali. Se il modello è un modello di segmentazione, la previsione include una maschera di anomalia che contrassegna i pixel in cui vengono rilevate le anomalie.

Per fare una previsione con il tuo modello, chiama l'DetectAnomaliesoperazione e passa un'immagine di input dal tuo computer locale. È possibile ottenere il comando CLI che chiama`DetectAnomalies` dalla console.

Ulteriori informazioni: Rileva le anomalie in un'immagine.

> **Important**
>
> Ti viene addebitato il costo per il tempo in cui il modello è in funzione.

Se non utilizzate più il modello, utilizzate l'StopModeloperazione per arrestarlo. presente nella console di controllo di controllo di controllo di controllo di controllo di controllo di controllo di controllo di controllo di controllo

Ulteriori informazioni: interrompi il tuo modello.

Usa il tuo modello su un dispositivo periferico

È possibile utilizzare un modello Lookout for Vision su un dispositivoAWS IoT Greengrass Version 2 principale.

Ulteriori informazioni: Utilizzo del modello Amazon Lookout for Vision su un dispositivo edge.

Usa il pannello di controllo di controllo

Puoi utilizzare la dashboard per ottenere una panoramica di tutti i tuoi progetti e informazioni di panoramica per i singoli progetti.

Ulteriori informazioni: utilizza il pannello di controllo.

Nozioni di base su Amazon Lookout for Vision

Prima di iniziare queste istruzioni introduttive, ti consigliamo di leggereComprendere Amazon Lookout for Vision.

Le istruzioni introduttive mostrano come utilizzare la creazione di un modello di segmentazione delle immagini di esempio. Se desideri creare un modello di classificazione delle immagini di esempio, vediset di dati di classificazione delle immagini.

Se desideri provare rapidamente un modello di esempio, forniamo immagini di allenamento di esempio e immagini di maschere. Forniamo anche uno script Python che crea un file manifest per la segmentazione delle immagini. Utilizzate il file manifest per creare un set di dati per il vostro progetto e non è necessario etichettare le immagini nel set di dati. Quando crei un modello con le tue immagini, devi etichettare le immagini nel set di dati. Per ulteriori informazioni, consulta Creare il tuo set di dati.

Le immagini che forniamo sono di cookie normali e anomali. Un cookie anomalo presenta una crepa nella forma del biscotto. Il modello che addestra con le immagini prevede una classificazione (normale o anomala) e trova l'area (maschera) delle crepe in un cookie anomalo, come mostrato nell'esempio seguente.

Argomenti

Fase 1: crea il file del manifesto e carica delle immagini

In questa procedura, cloni l'archivio della documentazione di Amazon Lookout for Vision sul tuo computer. Utilizzi quindi uno script Python (versione 3.7 o superiore) per creare un file manifest e caricare le immagini di training e le immagini delle maschere in una posizione Amazon S3 specificata. Utilizzate il file manifest per creare il vostro modello. Successivamente, utilizzi le immagini di prova nel repository locale per provare il tuo modello.

Per creare il file manifest e caricare immagini

1. Configura Amazon Lookout for Vision seguendo le istruzioni in Configurazione di Amazon Lookout for Vision. Assicurati di installare l'AWSSDK per Python.
2. Nella AWS regione in cui desideri utilizzare Lookout for Vision, crea un bucket S3.
3. Nel bucket Amazon S3 crea una cartella denominata. `getting-started`
4. Annotare l'URI Amazon S3 e il nome della risorsa Amazon (ARN) per la cartella. Li usi per configurare le autorizzazioni ed eseguire lo script.
5. Assicurati che l'utente che chiama lo script disponga delle autorizzazioni per chiamare l'`s3:PutObject`operazione. È possibile utilizzare la seguente policy. Per assegnare le autorizzazioni, vedere. Assegnare le autorizzazioni

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

6. Assicurati di avere un profilo locale denominato `lookoutvision-access` e che l'utente del profilo disponga delle autorizzazioni del passaggio precedente. Per ulteriori informazioni, consulta Utilizzo di un profilo su un computer locale.
7. Scarica il file zip, getting-started.zip. Il file zip contiene il set di dati introduttivo e lo script di configurazione.

8. Decomprimi il file `getting-started.zip`.
9. Al prompt dei comandi procedere come segue:

   a. Accedere alla cartella `getting-started`.

   b. Esegui il comando seguente per creare un file manifest e caricare le immagini di allenamento e le maschere di immagini nel percorso Amazon S3 indicato nel passaggio 4.

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

   c. Al termine dello script, annota il percorso del `train.manifest` file che lo script visualizza dopo`Create dataset using manifest file:`. Il percorso dovrebbe essere simile as`3://`*`path to getting started folder`*`/manifests/train.manifest`.

Fase 2: creazione del modello

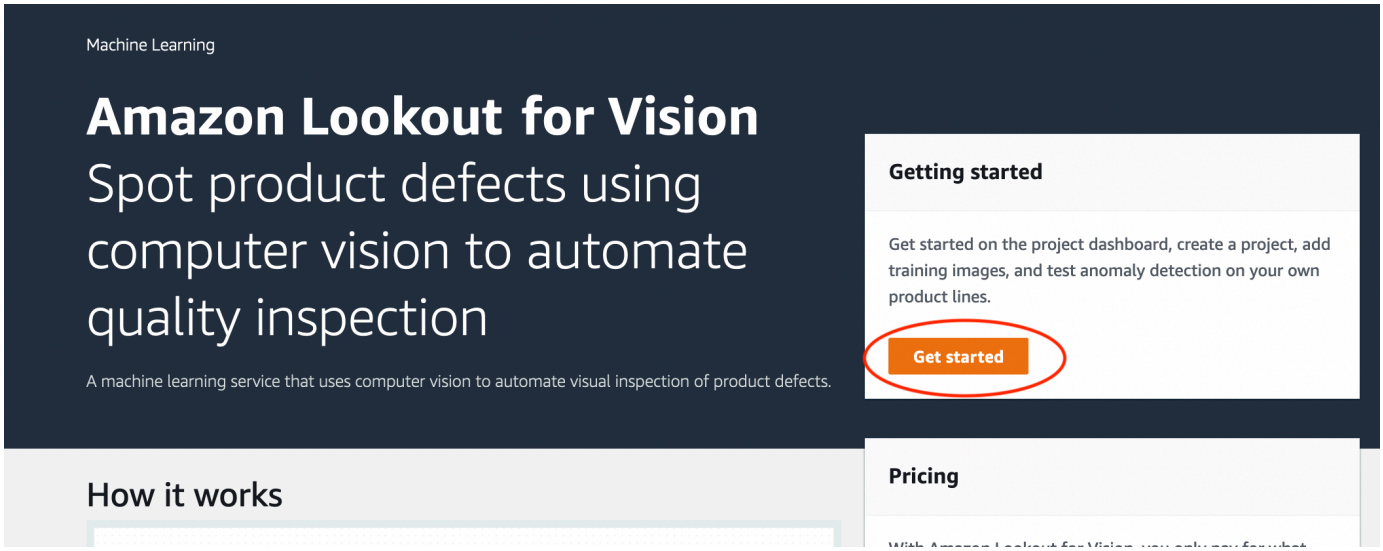
In questa procedura, crei un progetto e un set di dati utilizzando le immagini e il file manifest che hai precedentemente caricato nel tuo bucket Amazon S3. Quindi si crea il modello e si visualizzano i risultati della valutazione dell'addestramento dei modelli.

Poiché crei il set di dati dal file manifest introduttivo, non è necessario etichettare le immagini del set di dati. Quando crei un set di dati con le tue immagini, devi etichettare le immagini. Per ulteriori informazioni, consulta Immagini etichettate.

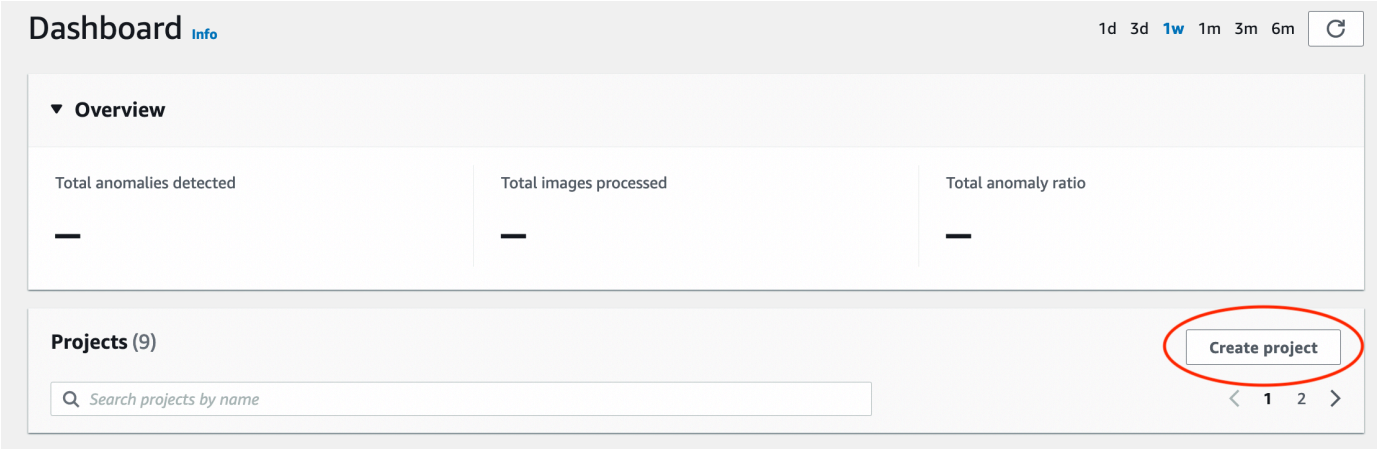
> **Important**
>
> Ti viene addebitato il costo di un addestramento di successo di un modello.

Per creare un modello

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Assicurati di essere nella stessa AWS regione in cui hai creato il bucket Amazon S3. Fase 1: crea il file del manifesto e carica delle immagini Per modificare la Regione, scegliere il nome della Regione attualmente visualizzata nella barra di navigazione. Quindi seleziona la Regione alla quale desideri passare.
3. Scegliere Inizia.



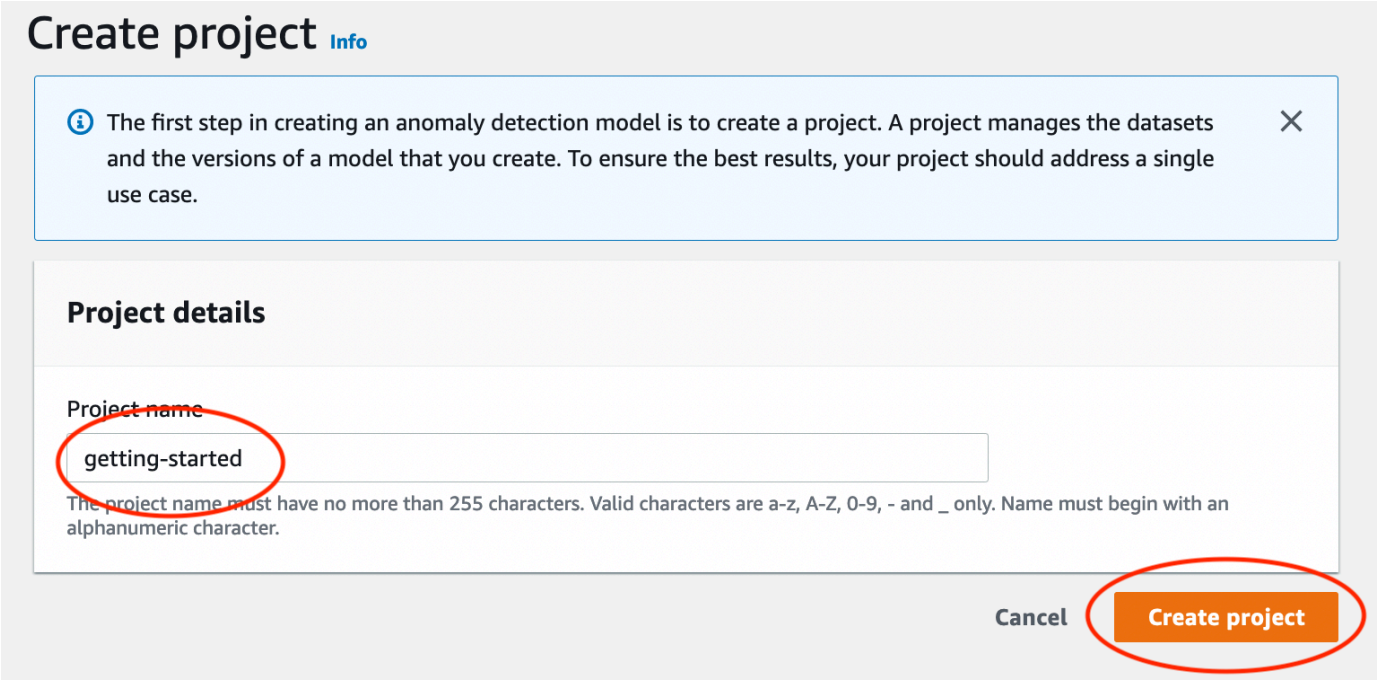
4. Nella sezione Progetti, scegli Crea progetto.



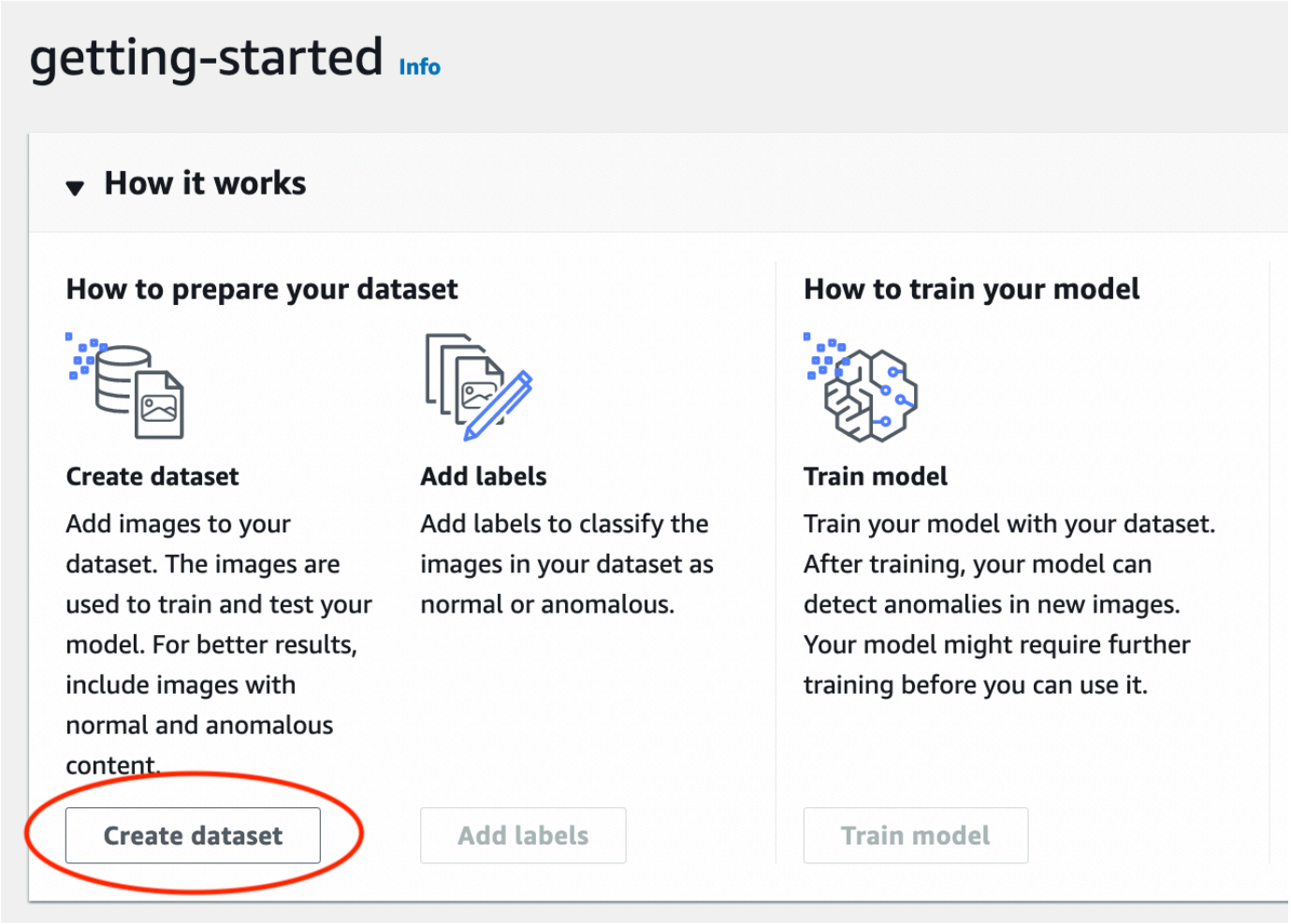
5. Nella pagina Crea progetto procedere come segue:

   a. In Nome progetto, inserisci`getting-started`.

   b. Seleziona Create project (Crea progetto).



6. Nella pagina del progetto, nella sezione Come funziona, scegli Crea set di dati.



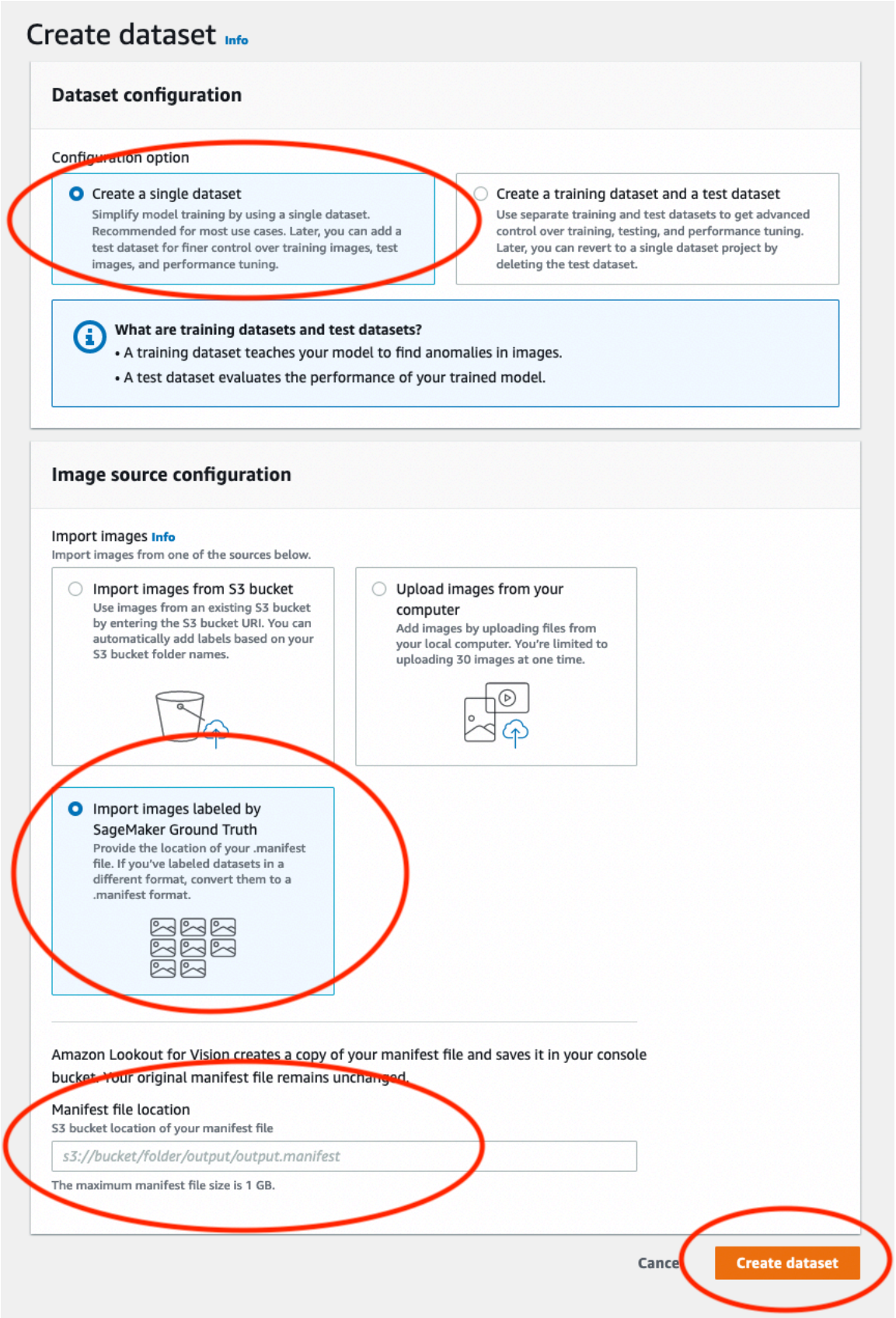
7. Nella pagina Crea set di dati procedere come segue:

   a. Scegli Crea un singolo set di dati.

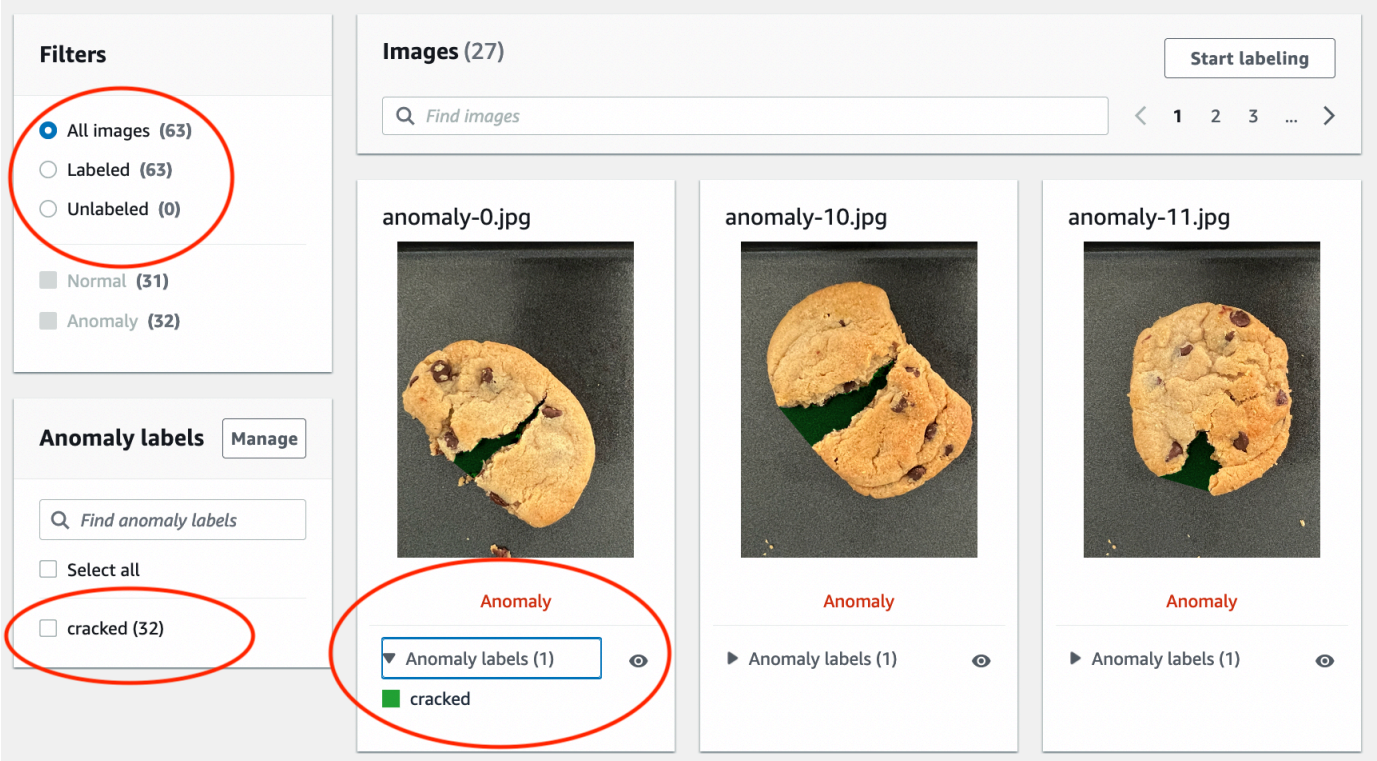
   b. Nella sezione Configurazione della sorgente dell'immagine, scegli Importa immagini etichettate da SageMaker Ground Truth.

   c. Per la posizione del file.manifest, inserisci la posizione Amazon S3 del file manifest che hai annotato nel passaggio 6.c. di. Fase 1: crea il file del manifesto e carica delle immagini La posizione di Amazon S3 dovrebbe essere simile a `s3://path to getting started folder/manifests/train.manifest`

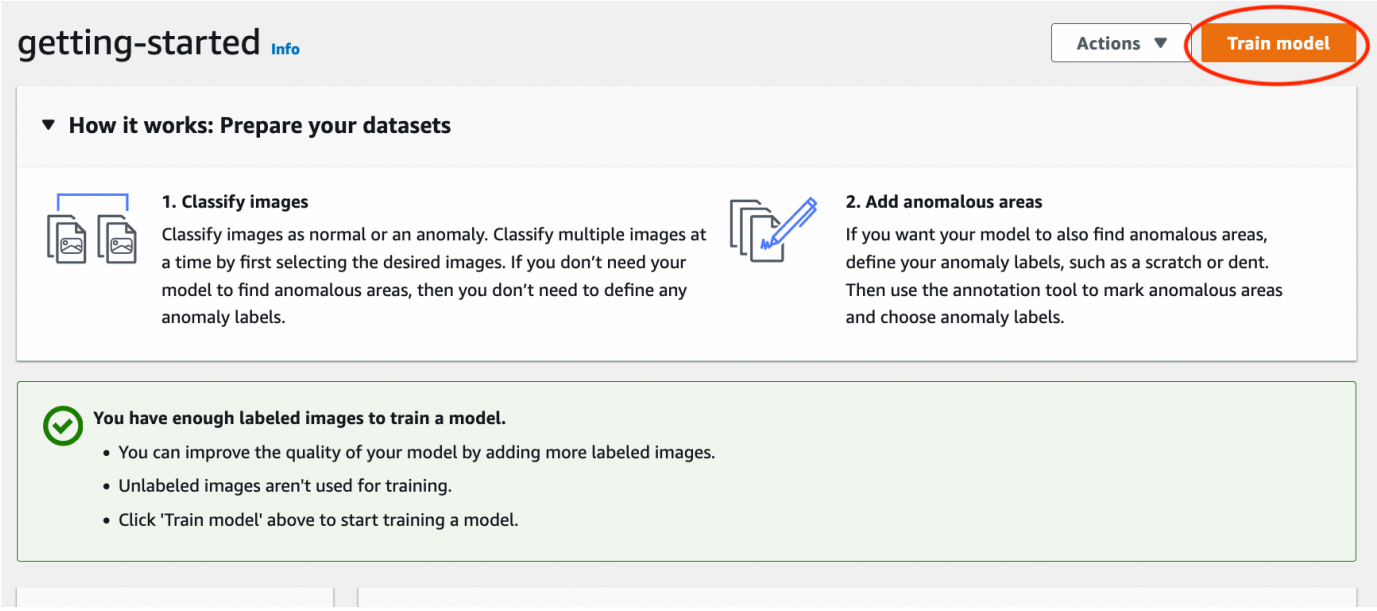
   d. Scegli Crea set di dati.


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.
Cance
Create dataset

8. Nella pagina dei dettagli del progetto, nella sezione Immagini, visualizza le immagini del set di dati. È possibile visualizzare le informazioni sulla classificazione e sulla segmentazione delle immagini (etichette di maschere e anomalie) per ogni immagine del set di dati. Puoi anche cercare immagini, filtrare le immagini in base allo stato dell'etichettatura (etichettate/senza etichetta) o filtrare le immagini in base alle etichette di anomalia ad esse assegnate.

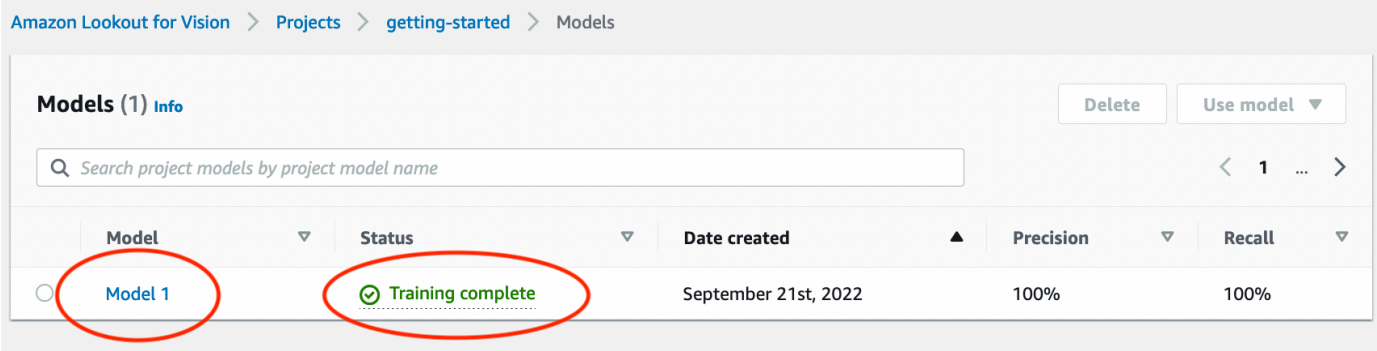


9. Nella pagina dei dettagli del progetto, scegli Modello ferroviario.



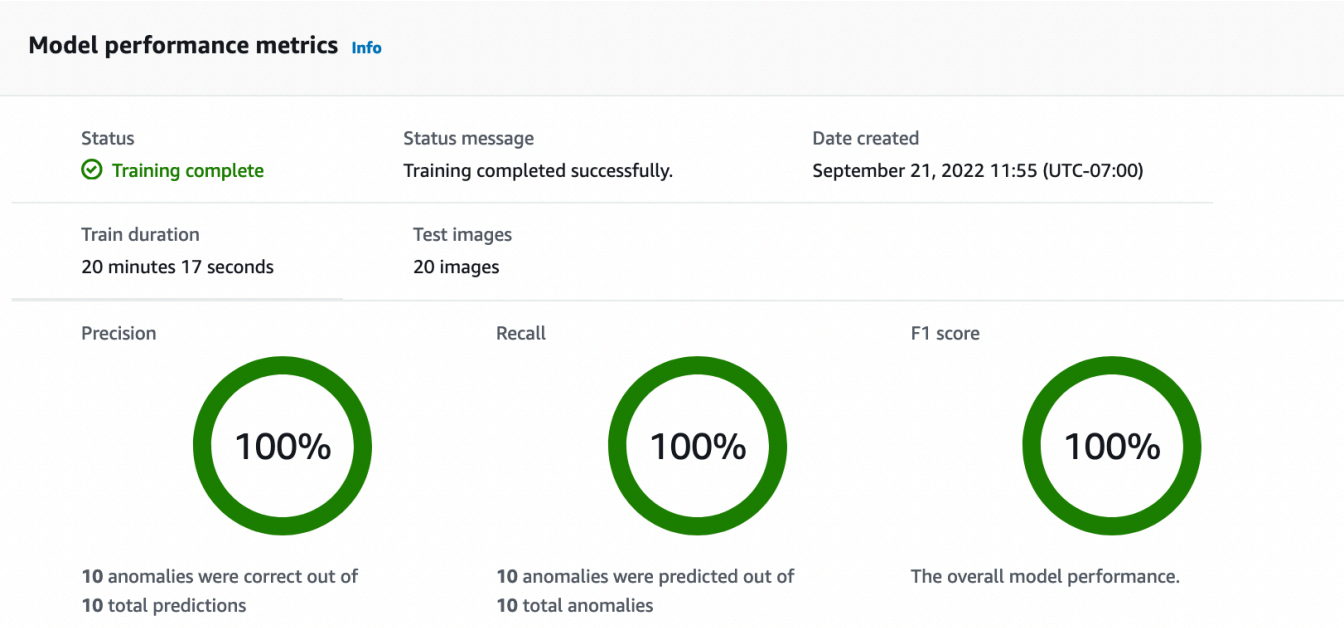
10. Nella pagina dei dettagli del modello di treno, scegli Modello di treno.

11. Nella sezione Vuoi addestrare il tuo modello? finestra di dialogo scegliere Modello di treno.
12. Nella pagina Modelli del progetto, puoi vedere che la formazione è iniziata. Controlla lo stato corrente visualizzando la colonna Stato per la versione del modello. L'addestramento del modello richiede almeno 30 minuti. L'allenamento è terminato con successo quando lo stato cambia in Allenamento completato.
13. Al termine dell'allenamento, scegli il modello Modello 1 nella pagina Modelli.

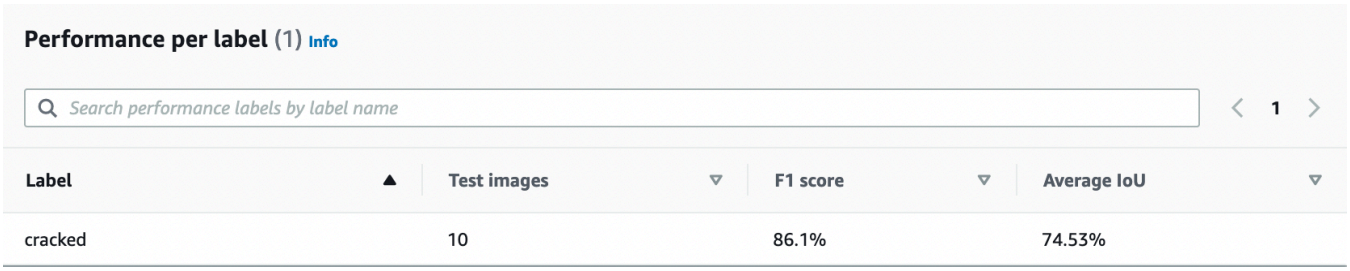


14. Nella pagina dei dettagli del modello, visualizza i risultati della valutazione nella scheda Metriche delle prestazioni. Esistono metriche per quanto segue:

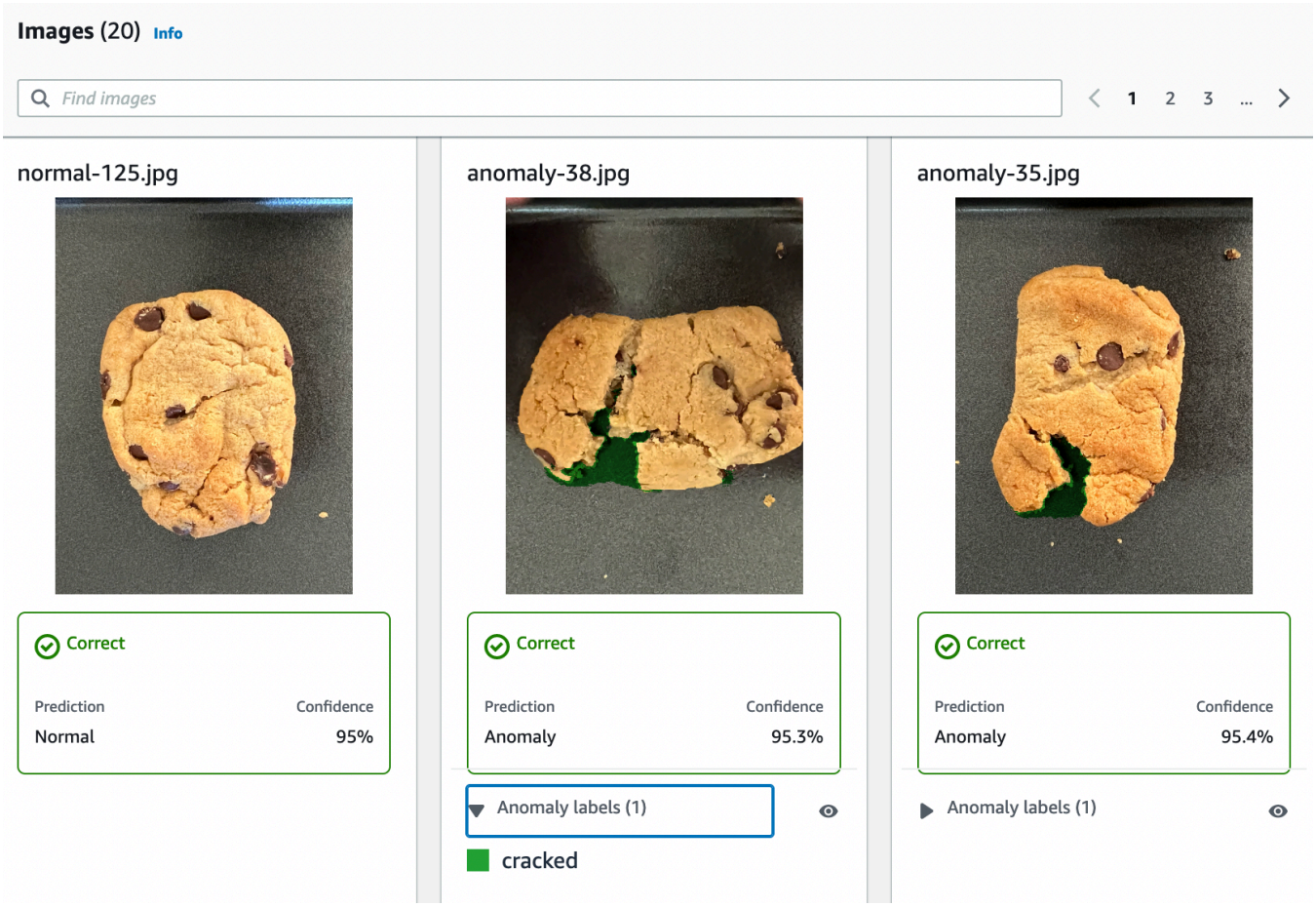
    - Metriche complessive delle prestazioni del modello (precisione, richiamo e punteggio F1) per le previsioni di classificazione effettuate dal modello.



    - Metriche delle prestazioni per le etichette delle anomalie presenti nelle immagini del test (IoU medio, punteggio F1)



- Previsioni per le immagini di test (classificazione, maschere di segmentazione ed etichette di anomalie)



Poiché l'allenamento modello non è deterministico, i risultati della valutazione potrebbero differire dai risultati mostrati in questa pagina. Per ulteriori informazioni, consulta Come migliorare il modello Amazon Lookout for Vision.

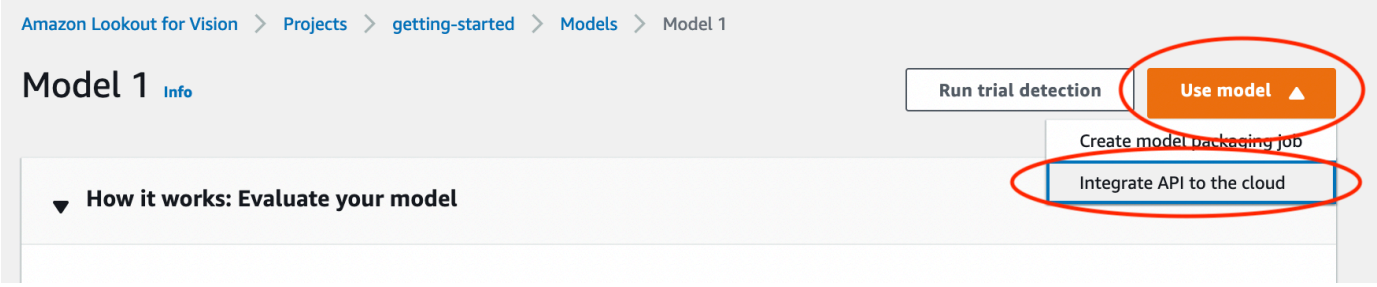
Fase 3: avvio del modello

In questo passaggio, iniziate a ospitare il modello in modo che sia pronto per analizzare le immagini. Per ulteriori informazioni, consulta Esecuzione del modello Amazon Lookout for Vision addestrato.

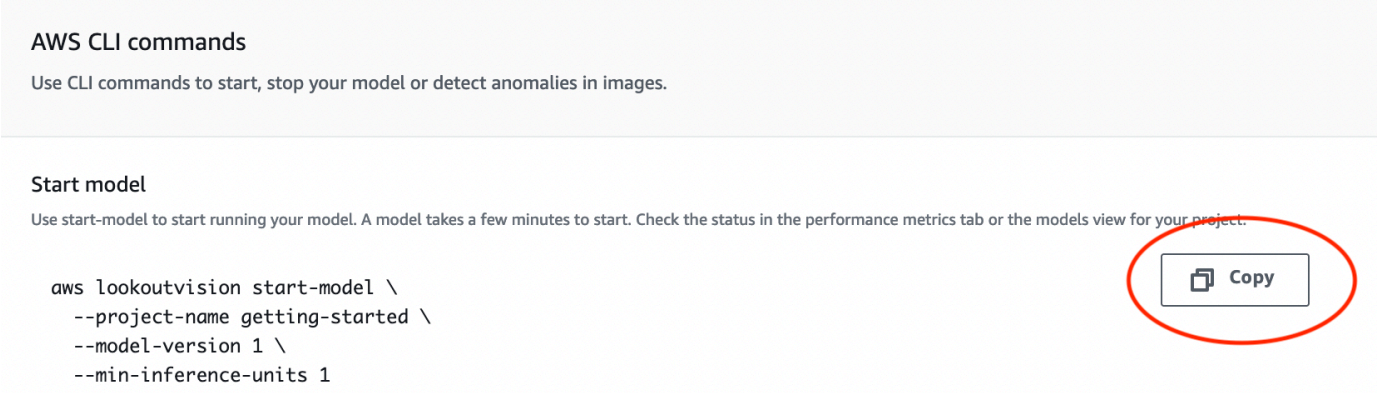
> **Note**
>
> Ti viene addebitato per il tempo di funzionamento del modello. Fai entrare il tuo modelloFase 5: Arresta il modello.

Per avviare il modello.

1. Nella pagina dei dettagli del modello, scegli Usa modello, quindi scegli Integrate API to the cloud.



2. Nella sezione AWS CLIcomandi, copia il `start-model` AWS CLI comando.



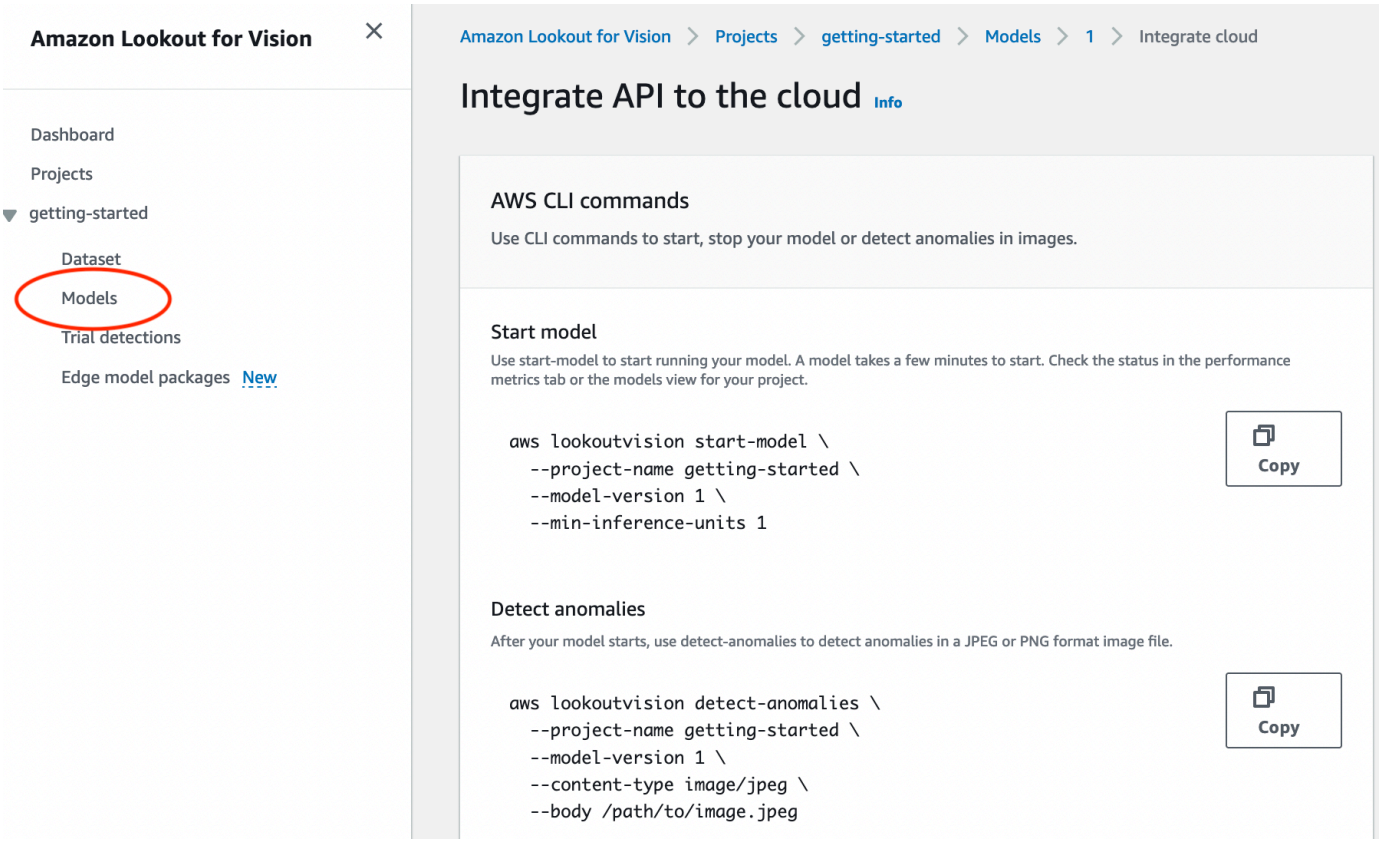
3. Assicurati che AWS CLI sia configurato per funzionare nella stessa AWS regione in cui utilizzi la console Amazon Lookout for Vision. Per modificare la AWS regione AWS CLI utilizzata, vedereInstallazione dell'SDK AWS.
4. Al prompt dei comandi, avviate il modello immettendo il `start-model` comando. Se utilizzi il `lookoutvision` profilo per ottenere le credenziali, aggiungi il `--profile lookoutvision-access` parametro. Ad esempio:

```
aws lookoutvision start-model \
  --project-name getting-started \
  --model-version 1 \
  --min-inference-units 1 \
  --profile lookoutvision-access
```

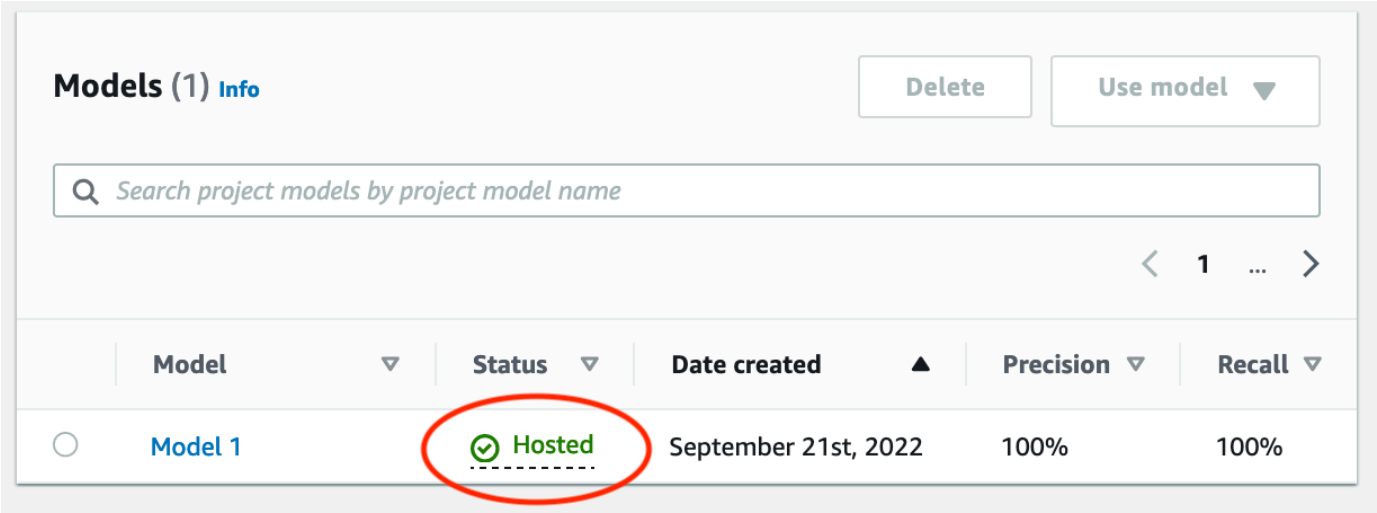
Se la chiamata ha esito positivo, viene visualizzato il seguente output:

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

5. Nella console scegliere Modelli nel riquadro di navigazione.



6. Attendere che lo stato del modello (Modello 1) nella colonna Stato venga visualizzato Hosted. Se hai già addestrato un modello nel progetto, attendi il completamento della versione più recente del modello.

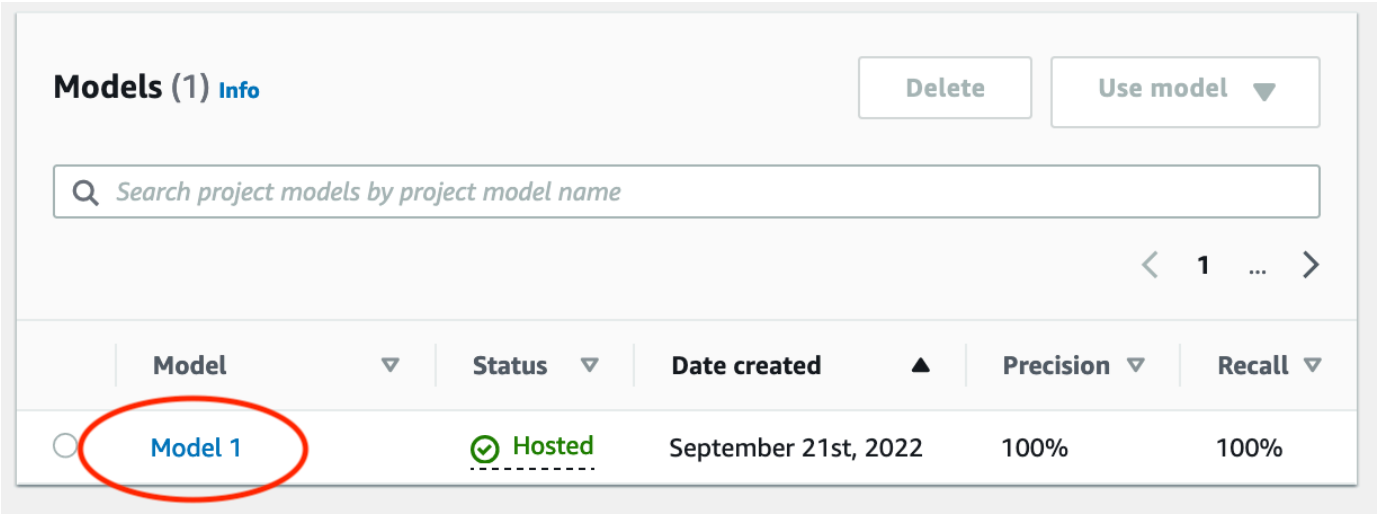


Fase 4: Analisi di un'immagine

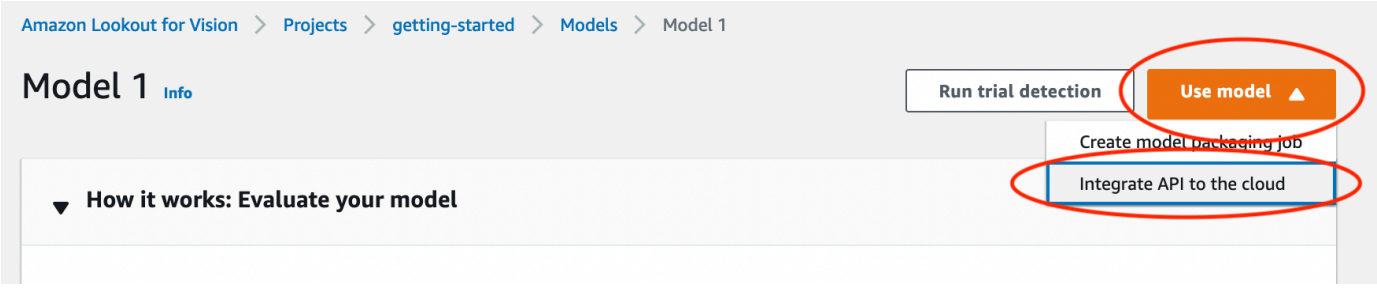
In questa fase verrà analizzata un'immagine con il modello. Forniamo immagini di esempio che puoi utilizzare nella `test-images` cartella Guida introduttiva dell'archivio della documentazione di Lookout for Vision sul tuo computer. Per ulteriori informazioni, consulta Rilevamento di anomalie in un'immagine.

Per analizzare un'immagine

1. Nella pagina Modelli, scegli il modello Modello 1.



2. Nella pagina dei dettagli del modello, scegli Usa modello, quindi scegli Integrate API to the cloud.



3. Nella sezione AWS CLIcomandi, copia il `detect-anomalies` AWS CLI comando.

4. Al prompt dei comandi, analizza un'immagine anomala inserendo il `detect-anomalies` comando del passaggio precedente. Per il `--body` parametro, specifica un'immagine anomala dalla `test-images` cartella introduttiva sul tuo computer. Se utilizzi il `lookoutvision` profilo per ottenere le credenziali, aggiungi il `--profile lookoutvision-access` parametro. Ad esempio:

```
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
```

L'output visualizzato dovrebbe essere simile al seguente:

```
{
    "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. Nell'output, tenere presente quanto segue:

   - `IsAnomalous`è un valore booleano per la classificazione prevista. `true`se l'immagine è anomala, altrimenti. `false`
   - `Confidence`è un valore variabile che rappresenta la fiducia di Amazon Lookout for Vision nella previsione. 0 è il livello di confidenza più basso, 1 è l'affidabilità massima.
   - `Anomalies`è un elenco di anomalie rilevate nell'immagine. `Name`è l'etichetta dell'anomalia. `PixelAnomaly`include l'area percentuale totale dell'anomalia (`TotalPercentageArea`) e un colore (`Color`) per l'etichetta dell'anomalia. L'elenco include anche un'anomalia di «sfondo» che copre l'area al di fuori delle anomalie rilevate nell'immagine.
   - `AnomalyMask`è un'immagine di maschera che mostra la posizione delle anomalie sull'immagine analizzata.

   È possibile utilizzare le informazioni nella risposta per visualizzare una combinazione dell'immagine analizzata e maschera delle anomalie, come illustrato nell'esempio seguente. Per il codice di esempio, consulta Visualizzazione delle informazioni su classificazione e segmentazione.


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

6. Al prompt dei comandi, analizza un'immagine normale dalla `test-images` cartella introduttiva. Se utilizzi il `lookoutvision` profilo per ottenere le credenziali, aggiungi il `--profile lookoutvision-access` parametro. Ad esempio:

```
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
```

L'output visualizzato dovrebbe essere simile al seguente:

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

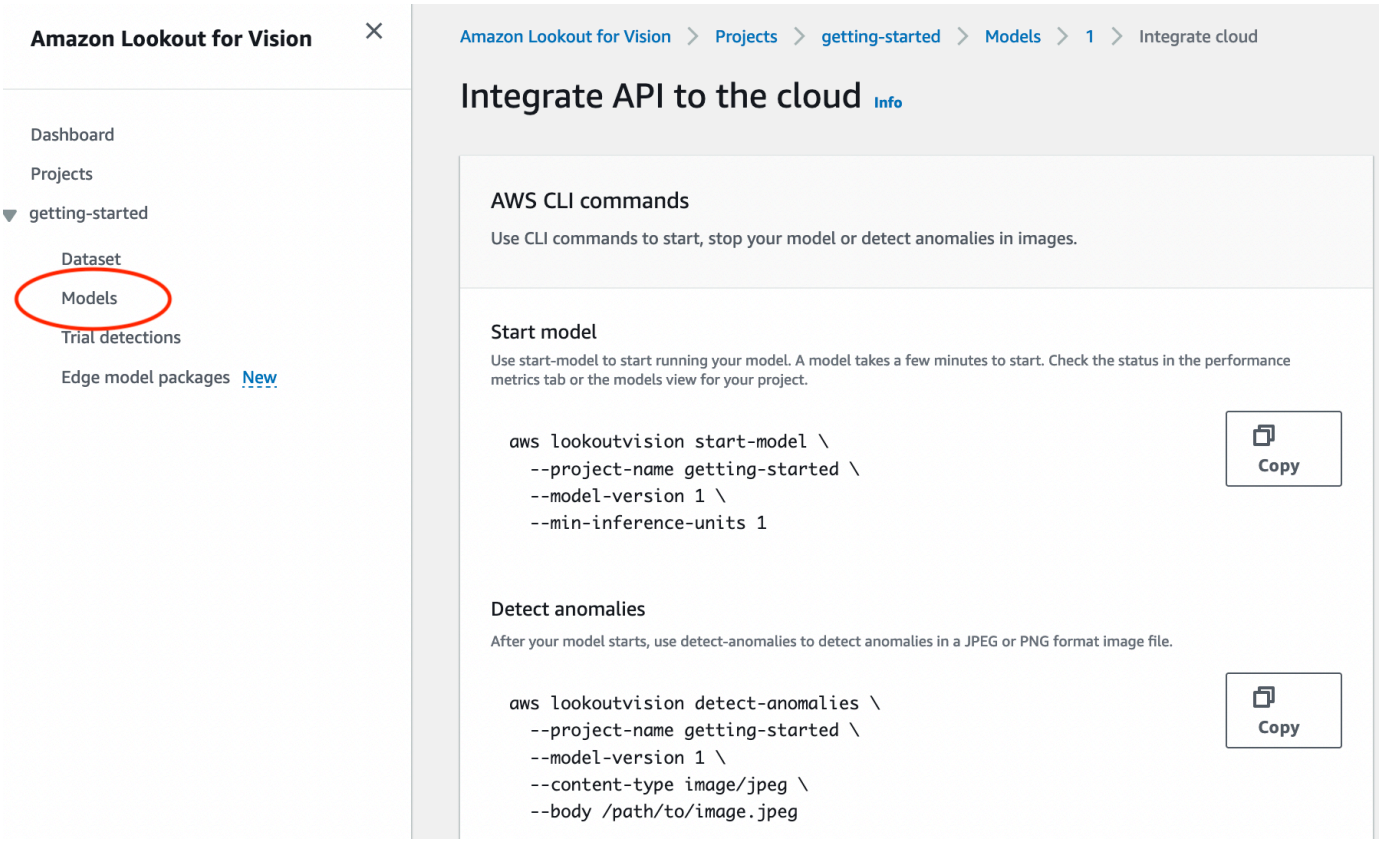
7. Nell'output, nota che il `false` valore di `IsAnomalous` classifica l'immagine come priva di anomalie. `Confidence`Utilizzalo per determinare la tua fiducia nella classificazione. Inoltre, l'`Anomalies`array ha solo l'etichetta dell'`background`anomalia.

Fase 5: Arresta il modello

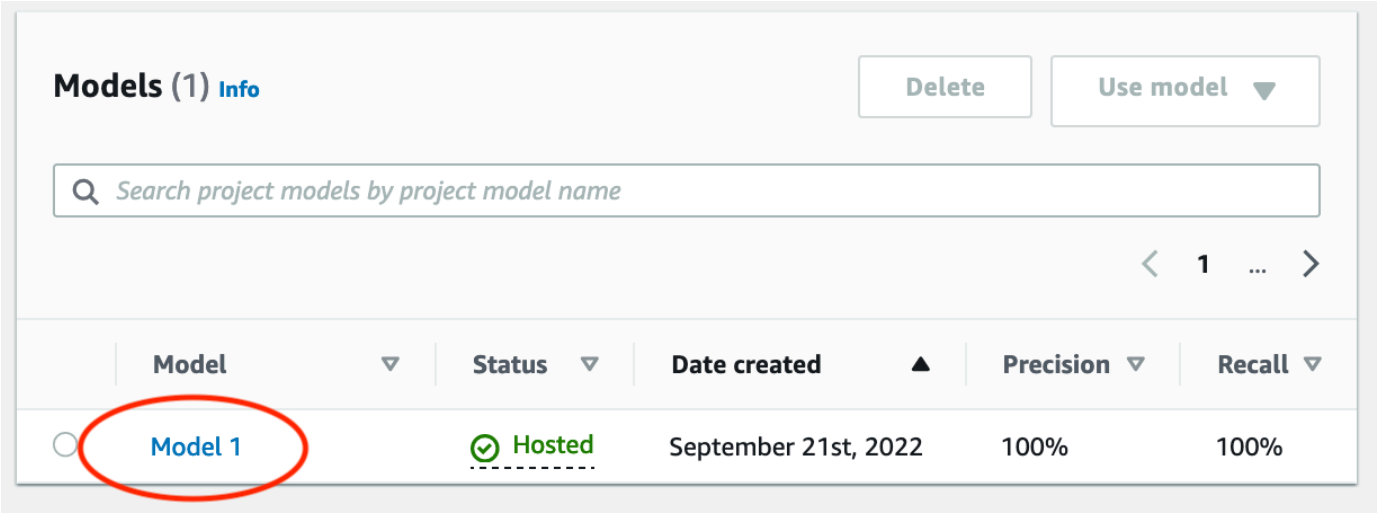
In questa fase verrà interrotto l'hosting del modello. Ti viene addebitato il periodo di funzionamento del tuo modello. Se non utilizzi il modello, è necessario interromperlo. È possibile riavviare il modello quando è di nuovo necessario. Per ulteriori informazioni, consulta Avvio del modello Amazon Lookout for Vision.

Per fermare il modello.

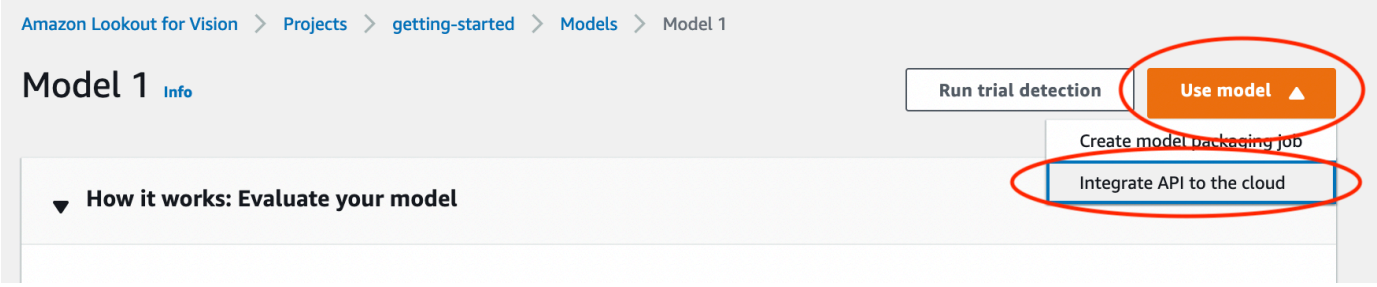
1. Nel riquadro di navigazione scegliere Modelli.



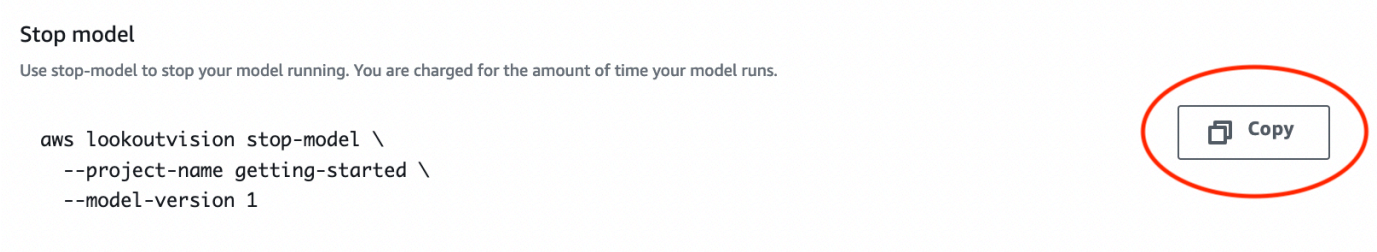
2. Nella pagina Modelli, scegli il modello Modello 1.



3. Nella pagina dei dettagli del modello, scegli Usa modello, quindi scegli Integrate API to the cloud.



4. Nella sezione AWS CLIcomandi, copia il `stop-model` AWS CLI comando.



5. Al prompt dei comandi, arrestate il modello immettendo il `stop-model` AWS CLI comando del passaggio precedente. Se utilizzi il `lookoutvision` profilo per ottenere le credenziali, aggiungi il `--profile lookoutvision-access` parametro. Ad esempio:

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

Se la chiamata ha esito positivo, viene visualizzato il seguente output:

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

6. Torna alla console, scegli Modelli nella pagina di navigazione a sinistra.
7. Il modello si è fermato quando lo stato del modello nella colonna Stato è Allenamento completato.

Fasi successive

Quando sei pronto a creare un modello con le tue immagini, inizia seguendo le istruzioni inCreare il tuo progetto. Le istruzioni includono i passaggi per creare un modello con la console Amazon Lookout for Vision e con l'AWSSDK.

Se vuoi provare altri set di dati di esempio, vediCodice e set di dati di esempio.

Creazione del modello Amazon Lookout for Vision

Un modello Amazon Lookout for Vision è un modello di apprendimento automatico che prevede la presenza di anomalie nelle nuove immagini trovando modelli nelle immagini utilizzate per addestrare il modello. Questa sezione mostra come creare e addestrare un modello. Dopo aver addestrato il modello, ne valutate le prestazioni. Per ulteriori informazioni, consulta Come migliorare il modello Amazon Lookout for Vision.

Prima di creare il primo modello, ti consigliamo di leggere Comprendere Amazon Lookout for Vision eNozioni di base su Amazon Lookout for Vision. Se utilizzi l'AWSSDK, leggiChiama un'azienda Amazon Lookout for Vision.

Argomenti

- Creare il tuo progetto
- Creare il tuo set di dati
- Immagini etichettate
- Addestrare il modello
- Risoluzione dei problemi relativi all'addestramento

Creare il tuo progetto

Un progetto Amazon Lookout for Vision è un raggruppamento di risorse necessarie per creare e gestire un modello Lookout for Vision. Un progetto gestisce quanto segue:

- Set di dati: le immagini e le etichette delle immagini utilizzate per addestrare un modello. Per ulteriori informazioni, consulta Creare il tuo set di dati.
- Modello: il software che addestra per rilevare le anomalie. È possibile disporre di più versioni di un modello. Per ulteriori informazioni, consulta Addestrare il modello.

Si consiglia di utilizzare un progetto per un singolo caso d'uso, ad esempio per il rilevamento di anomalie in un singolo tipo di parte della macchina.

> **Note**
>
> Puoi utilizzarlo AWS CloudFormation per effettuare il provisioning e configurare progetti Amazon Lookout for Vision. Per ulteriori informazioni, consulta Creare le risorse Amazon

Amazon Lookout for Vision Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon AmazonAWS CloudFormation.

Per visualizzare i tuoi progetti, consulta Visualizzazione dei progetti o apri ilUtilizzo Lookout for Vision di. Per eliminare un modello, vedereEliminazione di un modello.

Argomenti

- Creazione di un progetto (console)
- Creare un progetto (SDK)

Creazione di un progetto (console)

La procedura seguente mostra come creare un progetto utilizzando la console.

Creazione di un progetto (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Nel pannello di navigazione a sinistra, scegliere Progetti.
3. Seleziona Crea progetto.
4. In Nome progetto immetti un nome per il progetto.
5. Seleziona Crea progetto. Viene visualizzata la pagina dei dettagli del progetto.
6. Segui i passaggi indicati Creare il tuo set di dati per creare il tuo set di dati.

Creare un progetto (SDK)

Utilizzi l'CreateProjectoperazione per creare un progetto Amazon Lookout for Vision. Il modulo di risposta `CreateProject` include il nome del progetto e l'Amazon Resource Name (ARN) del progetto. Successivamente, chiama CreateDatasetper aggiungere un set di dati di formazione e test al tuo progetto. Per ulteriori informazioni, consulta Creazione di un set di dati con un file manifest (SDK).

Per visualizzare i progetti che hai creato in un progetto, chiama. `ListProjects` Per ulteriori informazioni, consulta Visualizzazione dei progetti.

Per creare un progetto (SDK)

1. Se non lo si è ancora fatto, installare e configurare AWS CLI e AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il seguente codice di esempio per creare un modello.

   CLI

   Modificate il valore `project-name` di con il nome che desiderate utilizzare per il progetto.

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

   Python

   Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

   ```
       @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

Questo codice è tratto dal repository degli esempi di AWS Documentation SDK. GitHub Guarda l'esempio completo qui.

```
/**
 * 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. Segui i passaggi indicati Creazione di un set di dati utilizzando un file manifest di Amazon SageMaker Ground Truth per creare il tuo set di dati.

Creare il tuo set di dati

Un set di dati contiene le immagini e le etichette assegnate utilizzate per addestrare e testare un modello. Puoi creare il set di dati per il tuo progetto con la console Amazon Lookout for Vision o con CreateDatasetl'operazione. Le immagini del set di dati devono essere etichettate in base al tipo di modello che desideri creare (classificazione delle immagini o segmentazione delle immagini).

Argomenti

Preparazione delle immagini per un set di dati

È necessaria una raccolta di immagini per creare un set di dati. Le immagini devono essere in formato PNG o JPEG. Il numero e il tipo di immagini necessarie dipendono dal fatto che il progetto abbia un unico set di dati o set di dati di formazione e test separati.

Progetto con set di dati singolo

Per creare un modello di classificazione delle immagini, è necessario quanto segue per iniziare la formazione:

- Almeno 20 immagini di oggetti normali.
- Almeno 10 immagini di oggetti anomali.

Per creare un modello di segmentazione delle immagini, è necessario quanto segue per iniziare la formazione:

- Almeno 20 immagini per ogni tipo di anomalia.
- Ogni immagine anomala (immagine con tipi di anomalia presenti) deve avere un solo tipo di anomalia.
- Almeno 20 immagini di oggetti normali.

Progetto separato di set di dati di formazione e test

Per creare un modello di classificazione delle immagini, è necessario quanto segue:

- Almeno 10 immagini di oggetti normali nel set di dati di addestramento.
- Almeno 10 immagini di oggetti normali nel set di dati del test.
- Almeno 10 immagini di oggetti anomali nel set di dati del test.

Per creare un modello di segmentazione delle immagini, è necessario quanto segue:

- Ogni set di dati necessita di almeno 10 immagini per ogni tipo di anomalia.
- Ogni immagine anomala (immagine con tipi di anomalia presenti) deve contenere solo un tipo di anomalia.
- Ogni set di dati deve avere almeno 10 immagini di oggetti normali.

Per creare un modello di qualità superiore, utilizzate un numero di immagini superiore al minimo. Se stai creando un modello di segmentazione, ti consigliamo di includere immagini con più tipi di anomalie, ma queste non rientrano nel conteggio minimo di cui Lookout for Vision ha bisogno per iniziare la formazione.

Le immagini devono appartenere a un unico tipo di oggetto. Inoltre, è necessario disporre di condizioni di acquisizione delle immagini coerenti, come il posizionamento della fotocamera, l'illuminazione e la posa dell'oggetto.

Tutte le immagini nei set di dati di addestramento e test devono avere le stesse dimensioni. Successivamente, le immagini che analizzerai con il modello addestrato devono avere le stesse dimensioni delle immagini dei set di dati di addestramento e test. Per ulteriori informazioni, consulta Rilevamento di anomalie in un'immagine.

Tutte le immagini di addestramento e test devono essere immagini uniche, preferibilmente di oggetti unici. Le immagini normali devono catturare le normali variazioni dell'oggetto da analizzare. Le immagini anomale devono catturare un campione diversificato di anomalie.

Amazon Lookout for Vision fornisce immagini di esempio che puoi utilizzare. Per ulteriori informazioni, consulta set di dati di classificazione delle immagini.

Per i limiti relativi alle immagini, consultaQuote.

Creazione del set di dati

Quando crei il set di dati per il tuo progetto, scegli la configurazione iniziale del set di dati del progetto. Puoi anche scegliere da dove importare le immagini da Lookout for Vision.

Scelta della configurazione del set di dati per il progetto

Quando crei il primo set di dati del tuo progetto, scegli una delle seguenti configurazioni di set di dati:

- Set di dati singolo: un singolo progetto di set di dati utilizza un unico set di dati per addestrare e testare il modello. L'utilizzo di un singolo set di dati semplifica la formazione lasciando che Amazon Lookout for Vision scelga le immagini di formazione e test. Durante la formazione, Amazon Lookout for Vision divide internamente il set di dati in un set di dati di addestramento e un set di dati di test. Non hai accesso ai set di dati suddivisi. Ti consigliamo di utilizzare un singolo progetto di set di dati per la maggior parte degli scenari.
- Set di dati di addestramento e test separati: se desideri un controllo più preciso su formazione, test e ottimizzazione delle prestazioni, puoi configurare il progetto in modo che disponga di set di dati di formazione e test separati. Utilizza un set di dati di test separato se desideri controllare le immagini utilizzate per i test o se disponi già di un set di immagini di riferimento che desideri utilizzare.

Puoi aggiungere un set di dati di test a un singolo progetto di set di dati esistente. Il singolo set di dati diventa quindi il set di dati di addestramento. Se rimuovi il set di dati di test da un progetto con set di dati di addestramento e test separati, il progetto diventa un singolo progetto di set di dati. Per ulteriori informazioni, consulta Eliminazione di un set di dati.

Importazione di immagini

Quando crei un set di dati, scegli da dove importare le immagini. A seconda di come si importano le immagini, le immagini potrebbero già essere etichettate. Se le immagini non sono etichettate dopo aver creato il set di dati, vedi. Immagini etichettate

Puoi creare un set di dati e importarne le immagini in uno dei seguenti modi:

- Importa immagini dal tuo computer locale. Le immagini non sono etichettate. Puoi aggiungere o etichettare utilizzando la console Lookout for Vision.
- Importa immagini da un bucket S3. Amazon Lookout for Vision può classificare le immagini utilizzando i nomi delle cartelle per etichettare le immagini. Usalo `normal` per immagini normali. Utilizzare `anomaly` per immagini anomale. Non è possibile assegnare automaticamente etichette di segmentazione.
- Importa un file manifest di Amazon SageMaker Ground Truth, che include immagini etichettate. Puoi creare e importare il tuo file manifest. Se hai molte immagini, prendi in considerazione l'utilizzo del servizio di etichettatura SageMaker Ground Truth. Quindi importi il file manifesto di output dal job Amazon SageMaker Ground Truth. Se necessario, puoi utilizzare la console Lookout for Vision per aggiungere o modificare etichette.

Se utilizzi l'AWSSDK, crei un set di dati con un file manifest di Amazon SageMaker Ground Truth. Per ulteriori informazioni, consulta Creazione di un set di dati utilizzando un file manifest di Amazon SageMaker Ground Truth.

Se, dopo aver creato il set di dati, le immagini sono etichettate, puoi addestrare il modello. Se le immagini non sono etichettate, aggiungete le etichette in base al tipo di modello che desiderate creare. Per ulteriori informazioni, consulta Immagini etichettate.

Puoi aggiungere altre immagini a un set di dati esistente. Per ulteriori informazioni, consulta Aggiungere immagini al set di dati.

Creazione di un set di dati utilizzando immagini archiviate nel computer locale

È possibile creare un set di dati utilizzando immagini caricate direttamente dal computer. Si possono caricare fino a 30 immagini alla volta. In questa procedura, è possibile creare un singolo progetto di set di dati o un progetto con set di dati di addestramento e test separati.

> **Note**
>
> Se hai appena completatoCreare il tuo progetto, la console dovrebbe mostrare la dashboard del progetto e non è necessario eseguire i passaggi da 1 a 3.

Creare un dataset utilizzando immagini in un computer locale (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Nel pannello di navigazione a sinistra, scegliere Progetti.
3. Nella pagina Progetti, scegliere il progetto a cui aggiungere un dataset.
4. Nella pagina dei dettagli del progetto, scegli Crea set di dati.
5. Scegli la scheda Set di dati singolo o la scheda Set di dati separati per addestramento e test e segui i passaggi.

   Single dataset

   a. Nella sezione Configurazione del set di dati, scegli Crea un singolo set di dati.

b. Nella sezione Configurazione dell'origine dell'immagine, scegli Carica immagini dal tuo computer.

c. Scegliere Creare dataset.

d. Nella pagina del set di dati, scegli Aggiungi immagini.

e. Scegliere le immagini che si desiderano caricare nel dataset dai file del tuo computer. Si possono trascinare le immagini o scegliere quelle che si desiderano caricare dal tuo computer locale.

f. Scegliere Upload images (Caricare immagini).

Separate training and test datasets

a. Nella sezione Configurazione del set di dati, scegli Crea un set di dati di addestramento e un set di dati di test.

b. Nella sezione Training dataset details (Dettagli del dataset di allenamento), scegliere Upload images from your computer (Caricare immagini dal tuo computer).

c. Nella sezione Test dataset details (Dettagli del dataset di test), scegliere Upload images from your computer (Caricare immagini dal tuo computer).

> **Note**
>
> I dataset di addestramento e test possono avere diverse fonti di immagini.

d. Scegliere Creare dataset. Viene visualizzata una pagina del set di dati con una scheda Addestramento e una scheda Test per i rispettivi set di dati.

e. Scegliere Actions (Azioni), quindi scegliere Add images to training dataset (Aggiungere immagini al dataset di addestramento).

f. Scegliere le immagini che si desiderano caricare nel dataset. Si possono trascinare le immagini o scegliere quelle che si desiderano caricare dal tuo computer locale.

g. Scegliere Upload images (Caricare immagini).

h. Ripetere le fasi 5e - 5g. Per il passaggio 5e, scegliere Actions (Azioni), quindi scegliere Add images to test dataset (Aggiungere immagini al dataset di test).

6. Seguire i passaggi indicati in Immagini etichettate per etichettare le tue immagini.

7. Seguire i passaggi indicati in Addestrare il modello per addestrare il modello.

Creazione di un set di dati utilizzando immagini archiviate in un bucket Amazon S3

Puoi creare un set di dati utilizzando immagini archiviate in un bucket Amazon S3. Con questa opzione, puoi utilizzare la struttura delle cartelle nel tuo bucket Amazon S3 per classificare automaticamente le tue immagini. Puoi archiviare le immagini nel bucket della console o in un altro bucket Amazon S3 nel tuo account.

Configurazione delle cartelle per l'etichettatura automatica

Durante la creazione del dataset, si può scegliere di assegnare nomi alle etichette delle immagini in base al nome della cartella che contiene le immagini. Le cartelle devono essere secondarie del percorso della cartella Amazon S3 specificato nell'URI S3 quando crei il set di dati.

Di seguito è riportata la `train` cartella per le immagini di esempio di Getting Started. Se specifichi la posizione della cartella Amazon S3 come`S3-bucket/circuitboard/train/`, alle immagini nella cartella `normal` viene assegnata l'etichetta. `Normal` Alle immagini contenute nella cartella `anomaly` viene assegnata l'etichetta`Anomaly`. I nomi delle sottocartelle più profonde non vengono utilizzati per etichettare le immagini.

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

Creazione di un set di dati utilizzando immagini da un bucket Amazon S3

La procedura seguente crea un set di dati utilizzando le immagini di esempio di classificazione archiviate in un bucket Amazon S3. Per utilizzare le tue immagini, crea la struttura di cartelle descritta in. Configurazione delle cartelle per l'etichettatura automatica

La procedura mostra anche come creare un singolo progetto di set di dati o un progetto che utilizza set di dati di addestramento e test separati.

Se non scegli di etichettare automaticamente le tue immagini, devi etichettare le immagini dopo la creazione dei set di dati. Per ulteriori informazioni, consulta Classificazione delle immagini (console).

> **Note**
>
> Se hai appena completatoCreare il tuo progetto, la console dovrebbe mostrare la dashboard del progetto e non devi eseguire i passaggi da 1 a 4.

Per creare un set di dati utilizzando immagini archiviate in un bucket Amazon S3

1. Se non l'hai già fatto, carica le immagini introduttive nel tuo bucket Amazon S3. Per ulteriori informazioni, consulta set di dati di classificazione delle immagini.
2. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
3. Nel pannello di navigazione a sinistra, scegliere Progetti.
4. Nella pagina Progetti, scegliere il progetto a cui aggiungere un dataset. Viene visualizzata la pagina di dettagli del progetto.
5. Scegliere Creare dataset. Viene visualizzata la pagina Creare dataset.

   > **Tip**
   >
   > Se stai seguendo le istruzioni introduttive, scegli Crea un set di dati di allenamento e un set di dati di test.

6. Scegli la scheda Set di dati singolo o la scheda Set di dati di addestramento e test separati e segui i passaggi.

Single dataset

a. Nella sezione Configurazione del set di dati, scegli Crea un singolo set di dati.
b. Inserisci le informazioni per i passaggi da 7 a 9 nella sezione Configurazione dell'origine dell'immagine.

Separate training and test datasets

a. Nella sezione Configurazione del set di dati, scegli Crea un set di dati di addestramento e un set di dati di test.
b. Per il tuo set di dati di allenamento, inserisci le informazioni per i passaggi da 7 a 9 nella sezione Dettagli del set di dati di allenamento.
c. Per il set di dati del test, inserisci le informazioni per i passaggi da 7 a 9 nella sezione Dettagli del set di dati del test.

> **Note**
>
> I dataset di addestramento e test possono avere diverse fonti di immagini.

7. Scegliere Import images from Amazon S3 bucket (Importa immagini dal bucket Amazon S3).
8. Nell'URI S3, inserire la posizione del bucket Amazon S3 e il percorso della cartella. Cambia con `bucket` il nome del bucket Amazon S3.

   a. Se stai creando un singolo progetto di set di dati o un set di dati di formazione, inserisci quanto segue:

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

   b. Se stai creando un set di dati di test, inserisci quanto segue:

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

9. Scegliere Automatically attach labels to images based on the folder (Allegare automaticamente etichette alle immagini secondo la cartella).
10. Scegliere Creare dataset. Si apre una pagina del set di dati con le immagini etichettate.
11. Seguire i passaggi indicati in Addestrare il modello per addestrare il modello.

Creazione di un set di dati utilizzando un file manifest di Amazon SageMaker Ground Truth

Un file manifesto contiene informazioni sulle immagini e sulle etichette delle immagini che puoi utilizzare per addestrare e testare un modello. Puoi archiviare un file manifest in un bucket Amazon S3 e utilizzarlo per creare un set di dati. Puoi creare il tuo file manifest o utilizzare un file manifest esistente, come l'output di un job di Amazon SageMaker Ground Truth.

Argomenti

Utilizzo di un job Amazon Sagemaker Ground Truth

L'etichettatura delle immagini può richiedere molto tempo. Ad esempio, possono essere necessari 10 secondi per disegnare con precisione una maschera attorno a un'anomalia. Se hai centinaia di immagini, potrebbero essere necessarie diverse ore per etichettarle. In alternativa all'etichettatura personalizzata delle immagini, prendi in considerazione l'utilizzo di Amazon SageMaker Ground Truth.

Con Amazon SageMaker Ground Truth, puoi utilizzare i lavoratori di Amazon Mechanical Turk, un'azienda fornitrice di tua scelta, o una forza lavoro interna privata per creare un set di immagini etichettato. Per ulteriori informazioni, consulta Use Amazon SageMaker Ground Truth to Label Data.

L'utilizzo di Amazon Mechanical Turk prevede un costo. Inoltre, potrebbero essere necessari diversi giorni per completare un processo di etichettatura di Amazon Ground Truth. Se il costo è un problema o se devi addestrare rapidamente il tuo modello, ti consigliamo di utilizzare la console Amazon Lookout for Vision per etichettare le tue immagini.

Puoi utilizzare un processo di etichettatura di Amazon SageMaker Ground Truth per etichettare immagini adatte ai modelli di classificazione delle immagini e ai modelli di segmentazione delle immagini. Al termine del processo, utilizzi il file manifest di output per creare un set di dati Amazon Lookout for Vision.

Classificazione delle immagini

Per etichettare le immagini per un modello di classificazione delle immagini, crea un processo di etichettatura per un'attività di classificazione delle immagini (etichetta singola).

Segmentazione delle immagini

Per etichettare le immagini per un modello di segmentazione delle immagini, create un processo di etichettatura per un'attività di classificazione delle immagini (etichetta singola). Quindi, concatenate il lavoro per creare un lavoro di etichettatura per un'attività di segmentazione semantica delle immagini.

È inoltre possibile utilizzare un processo di etichettatura per creare un file manifesto parziale per un modello di segmentazione dell'immagine. Ad esempio, è possibile classificare le immagini con un'attività di classificazione delle immagini (etichetta singola). Dopo aver creato un set di dati Lookout for Vision con l'output del lavoro, utilizza la console Amazon Lookout for Vision per aggiungere maschere di segmentazione ed etichette di anomalia alle immagini del set di dati.

Etichettatura delle immagini con Amazon SageMaker Ground Truth

La procedura seguente mostra come etichettare le immagini con le attività di etichettatura delle immagini di Amazon SageMaker Ground Truth. La procedura crea un file manifesto di classificazione delle immagini e, facoltativamente, concatena l'attività di etichettatura delle immagini per creare un file manifesto di segmentazione delle immagini. Se desideri che il tuo progetto abbia un set di dati di test separato, ripeti questa procedura per creare il file manifesto per il set di dati di test.

Per etichettare le immagini con Amazon SageMaker Ground Truth (Console)

1. Crea un lavoro Ground Truth per un'attività di classificazione delle immagini (etichetta singola) seguendo le istruzioni in Create a Labeling Job (Console).

   a. Per il passaggio 10, scegli Immagine dal menu a discesa della categoria Attività e scegli Classificazione delle immagini (etichetta singola) come tipo di attività.

   b. Per la fase 16, nella sezione dello strumento di etichettatura per la classificazione delle immagini (etichetta singola), aggiungi due etichette: normale e anomalia.
2. Attendi che la forza lavoro finisca di classificare le tue immagini.
3. Se state creando un set di dati per un modello di segmentazione delle immagini, effettuate le seguenti operazioni. Altrimenti vai al passaggio 4.

   a. Nella console Amazon SageMaker Ground Truth, apri la pagina Labeling jobs.

   b. Scegli il lavoro che hai creato in precedenza. Viene abilitato il menu Actions (Azioni).

   c. Nel menu Actions (Operazioni) scegli Copy (Copia). Si apre la pagina dei dettagli del lavoro.

   d. Nel tipo di attività, scegli la segmentazione semantica.

   e. Seleziona Avanti.

   f. Nella sezione Strumento di etichettatura della segmentazione semantica, aggiungi etichette di anomalia per ogni tipo di anomalia che desideri che il tuo modello trovi.
   g. Scegli Crea.
   h. Attendi che la forza lavoro applichi un'etichetta alle tue immagini.
4. Apri la console Ground Truth e apri la pagina Labeling jobs.
5. Se state creando un modello di classificazione delle immagini, scegliete il lavoro creato nel passaggio 1. Se state creando un modello di segmentazione delle immagini, scegliete il lavoro creato nel passaggio 3.
6. Nel riepilogo del lavoro di etichettatura, apri la posizione S3 nella posizione del set di dati di output. Nota la posizione del file manifest, che dovrebbe essere. `s3://output-dataset-location/manifests/output/output.manifest`
7. Ripetete questa procedura se desiderate creare un file manifesto per un set di dati di test. Altrimenti, segui le istruzioni riportate Creare il set di dati per creare un set di dati con il file manifest.

Creare il set di dati

Utilizzate questa procedura per creare un set di dati in un progetto Lookout for Vision con il file manifest annotato nel passaggio 6 di. Etichettatura delle immagini con Amazon SageMaker Ground Truth Il file manifest crea il set di dati di addestramento per un singolo progetto di set di dati. Se desideri che il tuo progetto abbia un set di dati di test separato, puoi eseguire un altro job di Amazon SageMaker Ground Truth per creare un file manifest per il set di dati di test. Oppure puoi creare tu stesso il file manifesto. Puoi anche importare immagini nel tuo set di dati di test da un bucket Amazon S3 o dal tuo computer locale. (Potrebbe essere necessario etichettare le immagini prima di poter addestrare il modello).

Questa procedura presuppone che il progetto non abbia alcun set di dati.

Per creare un set di dati con Lookout for Vision (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegliere Iniziare.
3. Nel pannello di navigazione a sinistra, scegliere Progetti.
4. Scegli il progetto che desideri aggiungere da utilizzare con il file manifest.
5. Nella sezione Come funziona, scegli Crea set di dati.

6. Scegli la scheda Set di dati singolo o la scheda Set di dati di addestramento e test separati e segui i passaggi.

   Single dataset

   1. Scegli Crea un singolo set di dati.
   2. Nella sezione Configurazione dell'origine dell'immagine, scegli Importa immagini etichettate da SageMaker Ground Truth.
   3. Per la posizione del file.manifest, inserisci la posizione del file manifest che hai annotato nel passaggio 6 di. Etichettatura delle immagini con Amazon SageMaker Ground Truth

   Separate training and test datasets

   1. Scegli Crea un set di dati di addestramento e un set di dati di test.
   2. Nella sezione Dettagli del set di dati di addestramento, scegli Importa immagini etichettate da SageMaker Ground Truth.
   3. Nella posizione del file.manifest, la posizione del file manifest annotato nel passaggio 6 di. Etichettatura delle immagini con Amazon SageMaker Ground Truth
   4. Nella sezione Dettagli del set di dati di test, scegli Importa immagini etichettate da SageMaker Ground Truth.
   5. Nella posizione del file.manifest, la posizione del file manifest annotato nel passaggio 6 di. Etichettatura delle immagini con Amazon SageMaker Ground Truth Ricordate che è necessario un file manifest separato per il set di dati di test.
7. Seleziona Invia.
8. Seguire i passaggi indicati in Addestrare il modello per addestrare il modello.

Creazione di un file manifest

È possibile creare un set di dati importando un file manifest in formato SageMaker Ground Truth. Se le immagini sono etichettate in un formato diverso da un file manifest SageMaker Ground Truth, utilizzate le seguenti informazioni per creare un file manifest in formato SageMaker Ground Truth.

I file manifest sono in formato righe JSON, dove ogni riga è un oggetto JSON completo che rappresenta le informazioni di etichettatura di un'immagine. Esistono diversi formati per la classificazione e la segmentazione delle immagini. I file manifesto devono essere codificati utilizzando la codifica UTF-8.

> **Note**
>
> Gli esempi di righe JSON in questa sezione sono formattati per garantire la leggibilità.

Le immagini a cui fa riferimento un file manifest devono essere situate nello stesso bucket Amazon S3. Il file manifest può trovarsi in un bucket diverso. È necessario specificare la posizione di un'immagine nel campo `source-ref` di una riga JSON.

È possibile creare un file manifesto utilizzando il codice. Il notebook Python di Amazon Lookout for Vision Lab mostra come creare un file manifesto di classificazione delle immagini per le immagini di esempio del circuito stampato. In alternativa, puoi utilizzare il codice di esempio Datasets nel Code Examples Repository. AWS È possibile creare facilmente un file manifest utilizzando un file CSV (Comma Separated Values). Per ulteriori informazioni, consulta Creazione di un file manifesto di classificazione da un file CSV.

Argomenti

Definizione delle righe JSON per la classificazione delle immagini

Definisci una riga JSON per ogni immagine che desideri utilizzare in un file manifest di Amazon Lookout for Vision. Se desideri creare un modello di classificazione, la riga JSON deve includere una classificazione delle immagini che sia normale o anomala. Una riga JSON è in formato SageMaker Ground Truth Classification Job Output. Un file manifest è composto da una o più righe JSON, una per ogni immagine che si desidera importare.

Per creare un file manifesto per immagini classificate

1. Creare un file di testo vuoto.
2. Aggiungere una riga JSON per ogni immagine che si vuole importare. Ogni riga JSON dovrebbe avere un aspetto simile alla seguente:

```
{"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. Salvare il file.

> **Note**
>
> È possibile utilizzare l'estensione `.manifest`, ma non è necessaria.

4. Creare un dataset utilizzando il file manifest che si è creato. Per ulteriori informazioni, consulta Creazione di un file manifest.

Linee JSON di classificazione

In questa sezione, imparerai come creare una riga JSON che classifica un'immagine come normale o anomala.

Anomalia (riga JSON)

La seguente riga JSON mostra un'immagine etichettata come anomalia. Si noti che il valore di è. `class-name` `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
}
```

Linea JSON normale

La seguente riga JSON mostra un'immagine etichettata come normale. Si noti che il valore di `class-name` è. `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
}
```

Chiavi e valori della riga JSON

Le seguenti informazioni descrivono le chiavi e i valori di una riga JSON di Amazon Lookout for Vision.

source-ref

(Obbligatorio) La posizione dell'immagine di Amazon S3. Il formato è `"s3://BUCKET/OBJECT_PATH"`. Le immagini in un dataset importato devono essere archiviate nello stesso bucket Amazon S3.

etichetta di anomalia

(Obbligatorio) L'etichetta dell'attributo. Usa la chiave `anomaly-label` o un altro nome di chiave a tua scelta. Il valore chiave (`0`nell'esempio precedente) è richiesto da Amazon Lookout for Vision, ma non viene utilizzato. Il manifesto di output creato da Amazon Lookout for Vision converte il valore `1` in un'immagine anomala e un valore `0` in un'immagine normale. Il valore di `class-name` determina se l'immagine è normale o anomala.

Devono esserci metadati corrispondenti identificati dal nome del campo con l'aggiunta di -metadata. Ad esempio, `"anomaly-label-metadata"`.

anomaly-label-metadata

(Obbligatorio) Metadati sull'attributo etichetta. Il nome del campo deve essere lo stesso dell'attributo etichetta con l'aggiunta di -metadata.

confidence

(Facoltativo) Attualmente non utilizzato da Amazon Lookout for Vision. Se specifichi un valore, usa un valore di`1`.

job-name (nome lavoro)

(Facoltativo) Un nome che si scegle per il lavoro che elabora l'immagine.

class-name (nome classe)

(Obbligatorio) Se l'immagine contiene contenuti normali, specificatelo`normal`, altrimenti specificatelo`anomaly`. Se il valore di `class-name` è qualsiasi altro valore, l'immagine viene aggiunta al set di dati come immagine senza etichetta. Per etichettare un'immagine, consulta. Aggiungere immagini al set di dati

annotato dall'uomo

(Obbligatorio) Specificare `"yes"`, se l'annotazione è stata completata da una persona fisica. In caso contrario, specificare `"no"`.

Data di creazione

(Facoltativo) La data e l'ora del Coordinated Universal Time (UTC) in cui è stata creata l'etichetta.

type (tipo)

(Obbligatorio) Il tipo di processo da applicare all'immagine. Per le etichette di anomalia a livello di immagine, il valore è. `"groundtruth/image-classification"`

Definizione delle righe JSON per la segmentazione delle immagini

Definisci una riga JSON per ogni immagine che desideri utilizzare in un file manifest di Amazon Lookout for Vision. Se desideri creare un modello di segmentazione, la riga JSON deve includere informazioni di segmentazione e classificazione dell'immagine. Un file manifest è composto da una o più righe JSON, una per ogni immagine che si desidera importare.

Per creare un file manifesto per immagini segmentate

1. Creare un file di testo vuoto.

2. Aggiungere una riga JSON per ogni immagine che si vuole importare. Ogni riga JSON dovrebbe avere un aspetto simile alla seguente:

```
{"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. Salvare il file.

> **Note**
>
> È possibile utilizzare l'estensione `.manifest`, ma non è necessaria.

4. Creare un dataset utilizzando il file manifest che si è creato. Per ulteriori informazioni, consulta Creazione di un file manifest.

Linee di segmentazione JSON

In questa sezione, imparerai come creare una riga JSON che includa informazioni sulla segmentazione e la classificazione di un'immagine.

La seguente riga JSON mostra un'immagine con informazioni di segmentazione e classificazione. `anomaly-label-metadata`contiene informazioni sulla classificazione. `anomaly-mask-refe` `anomaly-mask-ref-metadata` contengono informazioni sulla segmentazione.

```
{
    "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"
    }
}
```

Chiavi e valori della linea JSON

Le seguenti informazioni descrivono le chiavi e i valori di una riga JSON di Amazon Lookout for Vision.

source-ref

(Obbligatorio) La posizione dell'immagine di Amazon S3. Il formato è "s3://*BUCKET*/*OBJECT_PATH*". Le immagini in un dataset importato devono essere archiviate nello stesso bucket Amazon S3.

etichetta di anomalia

(Obbligatorio) L'etichetta dell'attributo. Usa la chiave `anomaly-label` o un altro nome di chiave a tua scelta. Il valore chiave (`1`nell'esempio precedente) è richiesto da Amazon Lookout for Vision, ma non viene utilizzato. Il manifesto di output creato da Amazon Lookout for Vision converte il valore `1` in un'immagine anomala e un valore `0` in un'immagine normale. Il valore di `class-name` determina se l'immagine è normale o anomala.

Devono esserci metadati corrispondenti identificati dal nome del campo con l'aggiunta di -metadata. Ad esempio, `"anomaly-label-metadata"`.

anomaly-label-metadata

(Obbligatorio) Metadati sull'attributo etichetta. Contiene informazioni sulla classificazione. Il nome del campo deve essere lo stesso dell'attributo etichetta con l'aggiunta di -metadata.

confidence

(Facoltativo) Attualmente non utilizzato da Amazon Lookout for Vision. Se specifichi un valore, usa un valore di`1`.

job-name (nome lavoro)

(Facoltativo) Un nome che si scegle per il lavoro che elabora l'immagine.

class-name (nome classe)

(Obbligatorio) Se l'immagine contiene contenuti normali, specificatelo`normal`, altrimenti specificatelo`anomaly`. Se il valore di `class-name` è qualsiasi altro valore, l'immagine viene aggiunta al set di dati come immagine senza etichetta. Per etichettare un'immagine, consulta. Aggiungere immagini al set di dati

annotato dall'uomo

(Obbligatorio) Specificare `"yes"`, se l'annotazione è stata completata da una persona fisica. In caso contrario, specificare `"no"`.

Data di creazione

(Facoltativo) La data e l'ora del Coordinated Universal Time (UTC) in cui è stata creata l'etichetta.

type (tipo)

(Obbligatorio) Il tipo di processo da applicare all'immagine. Usa il valore`"groundtruth/image-classification"`.

anomaly-mask-ref

(Obbligatorio) La posizione in Amazon S3 dell'immagine della maschera. `anomaly-mask-ref`Utilizzalo per il nome della chiave o usa un nome chiave a tua scelta. La chiave deve terminare con`-ref`. L'immagine della maschera deve contenere maschere colorate per ogni tipo di `internal-color-map` anomalia. Il formato è `"s3://BUCKET/OBJECT_PATH"`. Le immagini in un dataset importato devono essere archiviate nello stesso bucket Amazon S3. L'immagine della maschera deve essere un'immagine in formato Portable Network Graphic (PNG).

anomaly-mask-ref-metadata

(Obbligatorio) Metadati di segmentazione per l'immagine. `anomaly-mask-ref-metadata`Utilizzatelo per il nome della chiave o utilizzate un nome chiave a vostra scelta. Il nome della chiave deve terminare con`-ref-metadata`.

internal-color-map

(Obbligatorio) Una mappa di colori che corrisponde ai singoli tipi di anomalia. I colori devono corrispondere ai colori dell'immagine della maschera (`anomaly-mask-ref`).

key

(Obbligatorio) La chiave per accedere alla mappa. La voce `0` deve contenere il nome della classe BACKGROUND che rappresenta le aree al di fuori delle anomalie dell'immagine.

class-name (nome classe)

(Obbligatorio) Il nome del tipo di anomalia, ad esempio scratch o dent.

colore esadecimale

(Obbligatorio) Il colore esadecimale per il tipo di anomalia, ad esempio. `#2ca02c` Il colore deve corrispondere a un colore in`anomaly-mask-ref`. Il valore per il tipo di `BACKGROUND` anomalia è sempre`#ffffff`.

confidence

(Obbligatorio) Attualmente non è utilizzato da Amazon Lookout for Vision, ma è richiesto un valore float.

annotato dall'uomo

(Obbligatorio) Specificare `"yes"`, se l'annotazione è stata completata da una persona fisica. In caso contrario, specificare `"no"`.

Data di creazione

(Facoltativo) La data e l'ora del Coordinated Universal Time (UTC) in cui sono state create le informazioni sulla segmentazione.

type (tipo)

(Obbligatorio) Il tipo di processo da applicare all'immagine. Usa il valore. `"groundtruth/semantic-segmentation"`

Creazione di un file manifesto di classificazione da un file CSV

Questo esempio di script Python semplifica la creazione di un file manifesto di classificazione utilizzando un file CSV (Comma Separated Values) per etichettare le immagini. Creare il file CSV

Un file manifest descrive le immagini utilizzate per addestrare un modello. Un file manifest è costituito da una o più righe JSON. Ogni riga JSON descrive una singola immagine. Per ulteriori informazioni, consulta Definizione delle righe JSON per la classificazione delle immagini.

Un file CSV rappresenta dati tabulari su più righe in un file di testo. I campi in una riga sono separati da una virgola. Per ulteriori informazioni, consultare valori separati da virgola. Per questo script, ogni riga del file CSV include la posizione S3 di un'immagine e la classificazione delle anomalie per l'immagine (o). `normal` `anomaly` Ogni riga corrisponde a una riga JSON nel file manifest.

Ad esempio, il seguente file CSV descrive alcune delle immagini contenute nelle immagini di esempio.

```
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
```

Lo script genera righe JSON per ogni riga. Ad esempio, quanto segue è una riga JSON per la prima riga (`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"}}
```

Se il tuo file CSV non include il percorso Amazon S3 per le immagini, usa l'argomento della riga di comando per specificare `--s3-path` il percorso Amazon S3 verso le immagini.

Prima di creare il file manifest, lo script verifica la presenza di immagini duplicate nel file CSV e tutte le classificazioni di immagini che non sono o. `normal anomaly` Se vengono rilevati duplicati di immagini o errori di classificazione delle immagini, lo script esegue le seguenti operazioni:

- Registra la prima voce valida per tutte le immagini in un file CSV deduplicato.
- Registra le occorrenze duplicate di un'immagine nel file degli errori.
- Registra le classificazioni delle immagini non presenti `normal` o presenti `anomaly` nel file degli errori.
- Non crea un file manifesto.

Il file degli errori include il numero di riga in cui si trova un'immagine duplicata o un errore di classificazione nel file CSV di input. Utilizza il file CSV degli errori per aggiornare il file CSV di input, quindi esegui nuovamente lo script. In alternativa, utilizza il file CSV degli errori per aggiornare il file CSV deduplicato, che contiene solo voci di immagine uniche e immagini senza errori di classificazione delle immagini. Esegui nuovamente lo script con il file CSV deduplicato aggiornato.

Se non vengono rilevati duplicati o errori nel file CSV di input, lo script elimina il file CSV di immagine deduplicata e il file degli errori, poiché sono vuoti.

In questa procedura, creare il file CSV ed eseguire lo script Python per creare il file manifest. Lo script è stato testato con Python versione 3.7.

Creare un file manifest da un file CSV

1. Crea un file CSV con i seguenti campi in ogni riga (una riga per immagine). Non aggiungere una riga di intestazione al file CSV.

Campo 1	Campo 2
Il nome dell'immagine o il percorso Amazon S3 all'immagine. Ad esempio, `s3://s3bucket/circuitboard/train/anomaly/train-anomaly_10.jpg` . Non	La classificazione delle anomalie per l'immagine (`normal` o). `anomaly`

Campo 1	Campo 2
si può avere una combinazione tra immagini con il percorso Amazon S3 e senza.	

Ad esempio, `s3://s3bucket/circuitboard/train/anomaly/image_10.jpg,anomaly` o `image_11.jpg,normal`

2. Salvare il file CSV.
3. Eseguire il seguente Python script. Fornire gli argomenti seguenti:

   - `csv_file`: il file CVS che si è creato nella fase 1.
   - (Facoltativo)`--s3-path` *`s3://path_to_folder/`*: il percorso Amazon S3 da aggiungere ai nomi dei file immagine (campo 1). Utilizzare `--s3-path` se le immagini nel campo 1 non contengono già un percorso 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. Se si verificano immagini duplicate o si verificano errori di classificazione:
   a. Utilizzate il file degli errori per aggiornare il file CSV deduplicato o il file CSV di input.
   b. Esegui nuovamente lo script con il file CSV deduplicato aggiornato o il file CSV di input aggiornato.
5. Se intendi utilizzare un set di dati di test, ripeti i passaggi da 1 a 4 per creare un file manifest per il set di dati di test.
6. Se necessario, copia le immagini dal tuo computer nel percorso del bucket Amazon S3 specificato nella colonna 1 del file CSV (o specificato nella `--s3-path` riga di comando). Per copiare le immagini, inserisci il seguente comando al prompt dei comandi.

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

7. Segui le istruzioni riportate Creazione di un set di dati con un file manifest (console) per creare un set di dati. Se utilizzi l'AWSSDK, consulta. Creazione di un set di dati con un file manifest (SDK)

Creazione di un set di dati con un file manifest (console)

La procedura seguente mostra come creare un set di dati di addestramento o test importando un file manifest in SageMaker formato archiviato in un bucket Amazon S3.

Dopo aver creato il set di dati, puoi aggiungere altre immagini al set di dati o aggiungere etichette alle immagini. Per ulteriori informazioni, consulta Aggiungere immagini al set di dati.

Per creare un set di dati utilizzando un file manifest in formato SageMaker Ground Truth (console)

1. Crea o usa un file manifest in formato Ground Truth SageMaker compatibile con Amazon Lookout for Vision. Per ulteriori informazioni, consulta Creazione di un file manifest.
2. Accedi alla AWS Management Console e apri la console di Amazon S3 all'indirizzo https://console.aws.amazon.com/s3/.

3. In un bucket Amazon S3, crea una cartella per contenere il file manifest.
4. Carica il tuo file manifest nella cartella che hai appena creato.
5. Nel bucket Amazon S3, crea una cartella per archiviare le tue immagini.
6. Caricare le immagini nella cartella che è appena stata creata.

> **Important**
>
> Il valore del `source-ref` campo in ogni riga JSON deve essere mappato alle immagini nella cartella.

7. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
8. Scegliere Iniziare.
9. Nel pannello di navigazione a sinistra, scegliere Progetti.
10. Scegli il progetto che desideri aggiungere da utilizzare con il file manifest.
11. Nella sezione Come funziona, scegli Crea set di dati.
12. Scegli la scheda Set di dati singolo o la scheda Set di dati di addestramento e test separati e segui i passaggi.

    Single dataset

    1. Scegli Crea un singolo set di dati.
    2. Nella sezione Configurazione dell'origine dell'immagine, scegli Importa immagini etichettate da SageMaker Ground Truth.
    3. Per la posizione del file.manifest, inserisci la posizione del tuo file manifest.

    Separate training and test datasets

    1. Scegli Crea un set di dati di addestramento e un set di dati di test.
    2. Nella sezione Dettagli del set di dati di addestramento, scegli Importa immagini etichettate da SageMaker Ground Truth.
    3. Nella posizione del file.manifest, inserisci la posizione del file del manifesto di allenamento.
    4. Nella sezione Dettagli del set di dati di test, scegli Importa immagini etichettate da SageMaker Ground Truth.
    5. Nella posizione del file.manifest, inserisci la posizione del tuo file manifest di test.

> **Note**
>
> I dataset di addestramento e test possono avere diverse fonti di immagini.

13. Seleziona Invia.
14. Seguire i passaggi indicati in Addestrare il modello per addestrare il modello.

Amazon Lookout for Vision crea un set di dati nella cartella bucket Amazon S3. `datasets` Il file originale rimane invariato `.manifest`.

Creazione di un set di dati con un file manifest (SDK)

L'CreateDatasetoperazione viene utilizzata per creare i set di dati associati a un progetto Amazon Lookout for Vision.

Se desideri utilizzare un singolo set di dati per addestramento e test, crea un unico set di dati con il `DatasetType` valore impostato su. `train` Durante l'addestramento, il set di dati viene suddiviso internamente per creare un set di dati di addestramento e test. Non hai accesso ai set di dati di addestramento e test divisi. Se desideri un set di dati di test separato, effettua una seconda chiamata a `CreateDataset` con il set di `DatasetType` valori. `test` Durante l'addestramento, i set di dati di addestramento e test vengono utilizzati per addestrare e testare il modello.

Facoltativamente, puoi utilizzare il `DatasetSource` parametro per specificare la posizione di un file manifest in formato SageMaker Ground Truth utilizzato per popolare il set di dati. In questo caso, la chiamata a è asincrona. `CreateDataset` Per verificare lo stato corrente, invoca `DescribeDataset`. Per ulteriori informazioni, consulta Visualizzazione dei set di dati. Se si verifica un errore di convalida durante l'importazione, il valore di `Status` viene impostato su CREATE_FAILED e viene impostato il messaggio di stato (). `StatusMessage`

> **Tip**
>
> Se state creando un set di dati con il set di dati di esempio per iniziare, utilizzate il file manifest (`getting-started/dataset-files/manifests/train.manifest`) in cui viene creato lo script. Fase 1: crea il file del manifesto e carica delle immagini
> Se state creando un set di dati con le immagini di esempio del circuito stampato, avete due opzioni:

1. Crea il file manifest usando il codice. Il notebook Python di Amazon Lookout for Vision Lab mostra come creare il file manifest per le immagini di esempio del circuito stampato. In alternativa, usa il codice di esempio Datasets nel Code Examples Repository. AWS
2. Se hai già utilizzato la console Amazon Lookout for Vision per creare un set di dati con le immagini di esempio del circuito stampato, riutilizza i file manifest creati per te da Amazon Lookout for Vision. Le posizioni dei file manifest di addestramento e test sono. `s3://bucket/datasets/project name/train or test/manifests/output/output.manifest`

Se non si specifica`DatasetSource`, viene creato un set di dati vuoto. In questo caso, la chiamata a `CreateDataset` è sincrona. Successivamente, è possibile etichettare le immagini nel set di dati chiamando. UpdateDatasetEntries Per il codice di esempio, consultare Aggiungere altre immagini (SDK).

Se desideri sostituire un set di dati, elimina prima il set di dati esistente con DeleteDatasete poi crea un nuovo set di dati dello stesso tipo chiamando. `CreateDataset` Per ulteriori informazioni, consulta Eliminazione di un set di dati.

Dopo aver creato i set di dati, puoi creare il modello. Per ulteriori informazioni, consulta Addestramento di un modello (SDK).

È possibile visualizzare le immagini etichettate (righe JSON) all'interno di un set di dati chiamando. ListDatasetEntries Puoi aggiungere immagini etichettate chiamando. `UpdateDatasetEntries`

Per visualizzare informazioni sui set di dati di test e addestramento, vedere. Visualizzazione dei set di dati

Per creare un set di dati (SDK)

1. Se non lo si è ancora fatto, installare e configurare l'SDK AWS CLI e AWS Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Utilizza il seguente codice di esempio per creare un set di dati.

   CLI

   Imposta i valori seguenti:

   - `project-name`al nome del progetto a cui vuoi associare il set di dati.

- `dataset-type`al tipo di set di dati che vuoi creare (`train`o`test`).
- `dataset-source`alla posizione Amazon S3 del file manifest.
- `Bucket`al nome del bucket Amazon S3 che contiene il file manifest.
- `Key`al percorso e al nome del file manifest nel 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

Questo codice è tratto dal repository degli esempi di AWS Documentation SDK. GitHub Guarda l'esempio completo qui.

```
    @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

Questo codice è tratto dal repository degli esempi di AWS Documentation SDK. GitHub Guarda l'esempio completo qui.

```
/**
 * 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. Addestra il tuo modello seguendo i passaggi riportati in. Addestramento di un modello (SDK)

Immagini etichettate

Puoi utilizzare la console Amazon Lookout for Vision per aggiungere o modificare le etichette assegnate alle immagini nel tuo set di dati. Se utilizzi l'SDK, le etichette fanno parte del file manifest che fornisci. CreateDataset Puoi aggiornare le etichette per un'immagine UpdateDatasetEntrieschiamando. Per il codice di esempio, consultare Aggiungere altre immagini (SDK).

Scelta del tipo di modello

Le etichette assegnate alle immagini determinano il tipo di modello creato da Lookout for Vision. Se il progetto ha un set di dati di test separato, etichetta le immagini nello stesso modo.

Modello di classificazione delle immagini

Per creare un modello di classificazione delle immagini, si classifica ogni immagine come normale o anomala. Per ogni immagine, fai. Classificazione delle immagini (console)

Modello di segmentazione delle immagini

Per creare un modello di segmentazione dell'immagine, classificate ogni immagine come normale o anomala. Per ogni immagine anomala, specificate anche una maschera di pixel per ogni area anomala dell'immagine e un'etichetta di anomalia per il tipo di anomalia all'interno della maschera di pixel. Ad esempio, la maschera blu nell'immagine seguente indica la posizione di un tipo di anomalia da graffio su un'auto. È possibile specificare più di un tipo di etichetta di anomalia in un'immagine. Per ogni immagine, faiSegmentazione delle immagini (console).

Classificazione delle immagini (console)

Si utilizza la console Lookout for Vision per classificare le immagini in un set di dati come normali o come anomalie. Le immagini non classificate non vengono utilizzate per addestrare il modello.

Se stai creando un modello di segmentazione delle immagini, salta questa procedura ed eseguilaSegmentazione delle immagini (console), che include i passaggi per classificare le immagini.

> **Note**
>
> Se hai appena completatoCreare il tuo set di dati, al momento la console dovrebbe mostrare la dashboard del modello e non devi eseguire i passaggi da 1 a 4.

Per classificare le immagini (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Nel pannello di navigazione a sinistra, scegliere Progetti.
3. Nella pagina Progetti scegliere il progetto da usare.
4. Nel riquadro di navigazione a sinistra del progetto, scegli Dataset.
5. Se disponi di set di dati di addestramento e test separati, scegli la scheda relativa al set di dati che desideri utilizzare.
6. Scegli Inizia l'etichettatura.
7. Scegli Seleziona tutte le immagini in questa pagina.
8. Se le immagini sono normali, scegli Classizza come normale, altrimenti scegli Classizza come anomalia. Sotto ogni immagine viene visualizzata un'etichetta.
9. Se devi cambiare l'etichetta di un'immagine, procedi come segue:

    a. Scegli Anomalia o Normale sotto l'immagine.

    b. Se non riesci a determinare l'etichetta corretta per un'immagine, ingrandisci l'immagine scegliendo l'immagine nella galleria.

    > **Note**
    >
    > Puoi filtrare le etichette delle immagini scegliendo l'etichetta o lo stato dell'etichetta desiderato nella sezione Filtri.

10. Ripeti i passaggi 7-9 su ogni pagina, se necessario, fino a quando tutte le immagini nel set di dati non sono state etichettate correttamente.
11. Seleziona Salvataggio delle modifiche.

12. Se hai finito di etichettare le tue immagini, puoi addestrare il tuo modello.

Segmentazione delle immagini (console)

Se state creando un modello di segmentazione delle immagini, dovete classificare le immagini come normali o anomale. È inoltre necessario aggiungere informazioni di segmentazione alle immagini anomale. Per specificare le informazioni sulla segmentazione, dovete innanzitutto specificare le etichette di anomalia per ogni tipo di anomalia, ad esempio un'ammaccatura o un graffio, che desiderate che il modello trovi. Quindi specificate una maschera di anomalia e un'etichetta di anomalia per ogni anomalia sulle immagini anomale nel set di dati.

> **Note**
>
> Se stai creando un modello di classificazione delle immagini, non è necessario segmentare le immagini e non è necessario specificare etichette di anomalia.

Argomenti

- Specificare le etichette di anomalia
- Etichettare un'immagine
- Segmentazione di un'immagine con lo strumento di annotazione

Specificare le etichette di anomalia

Si definisce un'etichetta di anomalia per ogni tipo di anomalia presente nelle immagini del set di dati.

Specificare le etichette di anomalia

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Nel pannello di navigazione a sinistra, scegliere Progetti.
3. Nella pagina Progetti scegliere il progetto da usare.
4. Nel riquadro di navigazione a sinistra del progetto, scegli Dataset.
5. In Etichette di anomalia scegli Aggiungi etichette di anomalia. Se in precedenza hai aggiunto un'etichetta di anomalia, scegli Gestisci.
6. Nella finestra di dialogo , procedi come segue:

a. Inserisci l'etichetta di anomalia che desideri aggiungere e scegli Aggiungi etichetta di anomalia.
b. Ripeti il passaggio precedente finché non hai inserito tutte le etichette di anomalia che desideri vengano trovate dal modello.
c. (Facoltativo) Scegliete l'icona di modifica per modificare il nome dell'etichetta.
d. (Facoltativo) Scegliete l'icona di eliminazione per eliminare una nuova etichetta di anomalia. Non è possibile eliminare i tipi di anomalia attualmente utilizzati dal set di dati.

7. Scegli Conferma per aggiungere le nuove etichette di anomalia al set di dati.

Dopo aver specificato le etichette di anomalia, etichettate le immagini facendo. Etichettare un'immagine

Etichettare un'immagine

Per etichettare un'immagine per la segmentazione dell'immagine, classificatela come normale o come anomalia. Quindi, utilizzate lo strumento di annotazione per segmentare l'immagine disegnando maschere che coprano bene le aree per ogni tipo di anomalia presente nell'immagine.

Per etichettare un'immagine

1. Se disponi di set di dati di addestramento e test separati, scegli la scheda relativa al set di dati che desideri utilizzare.
2. Se non l'hai già fatto, specifica i tipi di anomalia per il tuo set di dati facendo. Specificare le etichette di anomalia
3. Scegli Inizia l'etichettatura.
4. Scegli Seleziona tutte le immagini in questa pagina.
5. Se le immagini sono normali, scegli Classizza come normale, altrimenti scegli Classizza come anomalia.
6. Per modificare l'etichetta per una singola immagine, scegli Normale o Anomalia sotto l'immagine.

> **Note**
>
> Puoi filtrare le etichette delle immagini scegliendo l'etichetta, o lo stato dell'etichetta, desiderato nella sezione Filtri. È possibile ordinare in base al punteggio di affidabilità nella sezione Opzioni di ordinamento.

7. Per ogni immagine anomala, scegli l'immagine per aprire lo strumento di annotazione. Aggiungi informazioni sulla segmentazione facendo. Segmentazione di un'immagine con lo strumento di annotazione
8. Seleziona Salvataggio delle modifiche.
9. Se hai finito di etichettare le tue immagini, puoi addestrare il tuo modello.

Segmentazione di un'immagine con lo strumento di annotazione

Si utilizza lo strumento di annotazione per segmentare un'immagine contrassegnando le aree anomale con una maschera.

Per segmentare un'immagine con lo strumento di annotazione

1. Apri lo strumento di annotazione selezionando l'immagine nella galleria del set di dati. Se necessario, scegli Avvia etichettatura per accedere alla modalità di etichettatura.
2. Nella sezione Etichette di anomalia, scegli l'etichetta di anomalia che desideri contrassegnare. Se necessario, scegli Aggiungi etichette di anomalia per aggiungere una nuova etichetta di anomalia.
3. Scegli uno strumento di disegno nella parte inferiore della pagina e disegna maschere che coprano perfettamente le aree anomale per l'etichetta dell'anomalia. L'immagine seguente è un esempio di maschera che copre strettamente un'anomalia.

Quello che segue è un esempio di maschera scadente che non copre bene un'anomalia.

4. Se hai più immagini da segmentare, scegli Avanti e ripeti i passaggi 2 e 3.
5. Scegli Invia e chiudi per completare la segmentazione delle immagini.

Addestrare il modello

Dopo aver creato i set di dati e aver etichettato le immagini, puoi addestrare il tuo modello. Come parte del processo di formazione, viene utilizzato un set di dati di test. Se si dispone di un singolo progetto di set di dati, le immagini nel set di dati vengono automaticamente suddivise in un set di dati di test e un set di dati di addestramento come parte del processo di formazione. Se il progetto ha un set di dati di addestramento e uno di test, questi vengono utilizzati per addestrare e testare separatamente il set di dati.

Una volta completato l'addestramento, è possibile valutare le prestazioni del modello e apportare i miglioramenti necessari. Per ulteriori informazioni, consulta Come migliorare il modello Amazon Lookout for Vision.

Per addestrare il tuo modello, Amazon Lookout for Vision crea una copia delle immagini di formazione e test di origine. Per impostazione predefinita, le immagini copiate sono crittografate con una chiave di proprietà e gestione di AWS. Puoi anche scegliere di utilizzare la tua chiave AWS Key

Management Service (KMS) personale. Per ulteriori informazioni, consulta Concetti di AWS Key Management Service. Le tue immagini di origine non sono modificate.

Puoi assegnare metadati al tuo modello sotto forma di tag. Per ulteriori informazioni, consulta Modelli di etichettatura.

Ogni volta che si addestra un modello, viene creata una nuova versione del modello. Se non hai più bisogno di una versione di un modello, puoi eliminarla. Per ulteriori informazioni, consulta Eliminazione di un modello.

Ti viene addebitato il tempo necessario per addestrare correttamente il tuo modello. Per ulteriori informazioni, consulta Orari di formazione.

Per visualizzare i modelli esistenti in un progetto,Visualizzazione dei modelli.

> **Note**
>
> Se hai appena completato Creare il tuo set di dati oAggiungere immagini al set di dati. Al momento la console dovrebbe mostrare la dashboard del modello e non è necessario eseguire i passaggi da 1 a 4.

Argomenti

- Addestramento di un modello (console)
- Addestramento di un modello (SDK)

Addestramento di un modello (console)

La procedura seguente mostra come addestrare il modello utilizzando la console.

Per addestrare il tuo modello (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Nel pannello di navigazione a sinistra, scegli Progetti.
3. Nella pagina Progetti, scegli il progetto che contiene il modello che desideri addestrare.

4. Nella pagina dei dettagli del progetto, scegli Train model. Il pulsante Train model è disponibile se hai abbastanza immagini etichettate per addestrare il modello. Se il pulsante non è disponibile, aggiungi altre immagini finché non avrai un numero sufficiente di immagini etichettate.
5. (Facoltativo) Se desideri utilizzare la tua chiave di crittografia AWS KMS, procedi come segue:
   a. In Crittografia dei dati di immagine scegli Personalizza le impostazioni (avanzate) di crittografia.
   b. In encryption.aws_kms_key inserisci l'Amazon Resource Name (ARN) della tua chiave o scegli una chiave AWS KMS esistente. Per creare una nuova chiave, scegli Crea una chiave AWS IMS.
6. (Facoltativo) Se desideri aggiungere tag al modello, procedi come segue:
   a. Nella sezione Tag, seleziona Aggiungi nuovo tag.
   b. Inserisci i seguenti dati:
      i. Il nome della chiave in Chiave.
      ii. Il valore della chiave in Valore.
   c. Per aggiungere altri tag, ripeti i passaggi 6a e 6b.
   d. (Facoltativo) Se desideri rimuovere un tag, scegli Rimuovi accanto al tag da rimuovere. Se stai rimuovendo un tag salvato in precedenza, questo viene rimosso quando salvi le modifiche.
7. Scegli Addestra modello.
8. Nella finestra di dialogo Vuoi addestrare il tuo modello?, scegli Addestra modello.
9. Nella vista Modelli, puoi vedere che l'allenamento è iniziato e puoi controllare lo stato attuale visualizzando la `Status` colonna relativa alla versione del modello. L'addestramento di un modello richiede tempo.
10. Al termine dell'allenamento, è possibile valutarne le prestazioni. Per ulteriori informazioni, consulta Come migliorare il modello Amazon Lookout for Vision.

Addestramento di un modello (SDK)

Si utilizza l'CreateModeloperazione per avviare l'addestramento, il test e la valutazione di un modello. Amazon Lookout for Vision addestra il modello utilizzando il set di dati di addestramento e test associato al progetto. Per ulteriori informazioni, consulta Creare un progetto (SDK).

Ogni volta che chiami`CreateModel`, viene creata una nuova versione del modello. Il modulo di risposta `CreateModel` include la versione del modello.

Ti viene addebitato un costo per ogni formazione modello di successo. Utilizzate il parametro `ClientToken` di input per evitare addebiti dovuti a ripetizioni inutili o accidentali del training sul modello da parte degli utenti. `ClientToken`è un parametro di input idempotente che garantisce `CreateModel` che l'allenamento venga completato una sola volta per uno specifico set di parametri. Una chiamata ripetuta a `CreateModel` con lo stesso `ClientToken` valore assicura che l'allenamento non venga ripetuto. Se non fornisci un valore per`ClientToken`, l'SDK AWS che stai utilizzando inserisce un valore per te. In questo modo si evita che tentativi ripetuti dopo un errore di rete avviino più processi di formazione, ma dovrai fornire un valore personalizzato per i tuoi casi d'uso. Per ulteriori informazioni, consulta CreateModel.

Il completamento dell'addestramento richiede tempo. Per verificare lo stato attuale, chiamate `DescribeModel` e passate il nome del progetto (specificato nella chiamata a`CreateProject`) e la versione del modello. Il `status` campo indica lo stato attuale dell'addestramento del modello. Per il codice di esempio, consultare Visualizzazione dei modelli (SDK).

Se la formazione ha successo, puoi valutare il modello. Per ulteriori informazioni, consulta Come migliorare il modello Amazon Lookout for Vision.

Per visualizzare i modelli che hai creato in un progetto, chiama`ListModels`. Per il codice di esempio, consultare Visualizzazione dei modelli.

Per addestrare modello (SDK)

1. Se non lo hai ancora fatto, installa e configura AWS CLI e gli SDK AWS. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il seguente codice di esempio per addestrare un modello.

   CLI

   Imposta i valori seguenti:

   - `project-name`al nome del progetto che contiene il modello che desideri creare.
   - `output-config`nella posizione in cui si desidera salvare i risultati dell'allenamento. Sostituisci i valori seguenti:
     - `output bucket`con il nome del bucket Amazon S3 in cui Amazon Lookout for Vision salva i risultati della formazione.

- `output folder`con il nome della cartella in cui desideri salvare i risultati dell'allenamento.
- `Key`con il nome di una chiave tag.
- `Value`con un valore a cui associarsi`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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Guarda l'esempio completo qui.

```
    @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

Questo codice è tratto dal repository degli esempi di AWS Documentation SDK. GitHub Guarda l'esempio completo qui.

```
/**
 * 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. Al termine dell'allenamento, è possibile valutarne le prestazioni. Per ulteriori informazioni, consulta Come migliorare il modello Amazon Lookout for Vision.

Risoluzione dei problemi relativi all'addestramento

I problemi con il file manifesto o le immagini di addestramento possono causare il fallimento dell'addestramento del modello. Prima di riqualificare il modello, verificate i seguenti potenziali problemi.

I colori delle etichette delle anomalie non corrispondono al colore delle anomalie nell'immagine della maschera

Se state addestrando un modello di segmentazione dell'immagine, il colore dell'etichetta di anomalia nel file manifesto deve corrispondere al colore dell'immagine della maschera. La riga JSON per un'immagine nel file manifest contiene dei metadati (`internal-color-map`) che indicano ad Amazon Lookout for Vision quale colore corrisponde a un'etichetta di anomalia. Ad esempio, il colore dell'etichetta di `scratch` anomalia nella riga JSON seguente è. `#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"
    }
}
```

Se i colori dell'immagine della maschera non corrispondono ai valori in`hex-color`, la formazione ha esito negativo ed è necessario aggiornare il file manifest.

Per aggiornare i valori dei colori in un file manifesto

1. Utilizzando un editor di testo, aprite il file manifesto che avete usato per creare il set di dati.
2. Per ogni riga (immagine) JSON, verifica che i colori (`hex-color`) all'interno del `internal-color-map` campo corrispondano ai colori delle etichette di anomalia nell'immagine della maschera.

   È possibile ottenere la posizione dell'immagine della maschera dal *`anomaly-mask`*`-ref` campo. Scarica l'immagine sul tuo computer e usa il seguente codice per ottenere i colori di un'immagine.

   ```
   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. Per ogni immagine con un'assegnazione di colori errata, aggiorna il `hex-color` campo nella riga JSON dell'immagine.
4. Salvate il file manifesto dell'aggiornamento.
5. Eliminare il set di dati esistente dal progetto.
6. Crea un nuovo set di dati nel progetto con il file manifest aggiornato.

7. Addestra il modello.

In alternativa, per i passaggi 5 e 6, puoi aggiornare singole immagini nel set di dati richiamando l'UpdateDatasetEntriesoperazione e fornendo righe JSON aggiornate per le immagini che desideri aggiornare. Per il codice di esempio, consultare Aggiungere altre immagini (SDK).

Le immagini delle maschere non sono in formato PNG

Se state addestrando un modello di segmentazione delle immagini, le immagini delle maschere devono essere in formato PNG. Se create un set di dati da un file manifest, assicuratevi che le immagini della maschera a cui fate riferimento *anomaly-mask*`-ref` siano in formato PNG. Se le immagini della maschera non sono in formato PNG, è necessario convertirle in formato PNG. Non è sufficiente rinominare l'estensione di un file di immagine in`.png`.

Le immagini delle maschere che crei nella console Amazon Lookout for Vision o con SageMaker un lavoro Ground Truth vengono create in formato PNG. Non è necessario modificare il formato di queste immagini.

Per correggere immagini mascherate in formato non PNG in un file manifesto

1. Utilizzando un editor di testo, aprite il file manifesto che avete usato per creare il set di dati.
2. Per ogni riga JSON (immagine) assicuratevi che l'immagine faccia *anomaly-mask*`-ref` riferimento a un'immagine in formato PNG. Per ulteriori informazioni, consulta Creazione di un file manifest.
3. Salva il file manifesto aggiornato.
4. Eliminare il set di dati esistente dal progetto.
5. Crea un nuovo set di dati nel progetto con il file manifest aggiornato.
6. Addestra il modello.

Le etichette di segmentazione o classificazione sono imprecise o mancanti

Etichette mancanti o imprecise possono causare errori nella formazione o creare un modello con prestazioni scadenti. Ti consigliamo di etichettare tutte le immagini nel set di dati. Se non etichettate tutte le immagini e l'addestramento del modello fallisce o se le prestazioni del modello sono scadenti, aggiungete altre immagini.

Verifica quanto segue:

- Se state creando un modello di segmentazione, le maschere devono coprire accuratamente le anomalie sulle immagini del set di dati. Per controllare le maschere nel tuo set di dati, visualizza le immagini nella galleria dei set di dati del progetto. Se necessario, ridisegna le maschere delle immagini. Per ulteriori informazioni, consulta Segmentazione delle immagini (console).

- Assicurati che le immagini anomale presenti nelle immagini del tuo set di dati siano classificate. Se stai creando un modello di segmentazione delle immagini, assicurati che le immagini anomale abbiano etichette di anomalia e maschere di immagine.

  È importante ricordare quale tipo di modello (segmentazione o classificazione) stai creando. Un modello di classificazione non richiede maschere di immagini su immagini anomale. Non aggiungere maschere alle immagini dei set di dati destinate a un modello di classificazione.

  Per aggiornare le etichette mancanti

  1. Apri la galleria dei set di dati del progetto.
  2. Filtra le immagini senza etichetta per vedere quali immagini non hanno etichette.
  3. Esegui una di queste operazioni:
     - Se stai creando un modello di classificazione delle immagini, classifica ogni immagine senza etichetta.
     - Se state creando un modello di segmentazione delle immagini, classificate e segmentate ogni immagine senza etichetta.
  4. Se state creando un modello di segmentazione delle immagini, aggiungete delle maschere a tutte le immagini anomale classificate che non presentano maschere.
  5. Addestra il modello.

Se scegli di non correggere le etichette scadenti o mancanti, ti consigliamo di aggiungere altre immagini etichettate o di rimuovere le immagini interessate dal set di dati. Puoi aggiungerne altre dalla console o utilizzando l'UpdateDatasetEntriesoperazione. Per ulteriori informazioni, consulta Aggiungere immagini al set di dati.

Se scegli di rimuovere le immagini, devi ricreare il set di dati senza le immagini interessate, poiché non puoi eliminare un'immagine da un set di dati. Per ulteriori informazioni, consulta Rimuovere immagini dal set di dati.

Come migliorare il modello Amazon Lookout for Vision

Durante l'allenamento, Lookout for Vision testa il modello con il set di dati di test e utilizza i risultati per creare metriche delle prestazioni. Puoi utilizzare le metriche delle prestazioni per valutare le prestazioni del tuo modello. Se necessario, puoi adottare misure per migliorare i tuoi set di dati e quindi riqualificare il modello.

Se sei soddisfatto delle prestazioni del modello, puoi iniziare a usarlo. Per ulteriori informazioni, consulta Esecuzione del modello Amazon Lookout for Vision addestrato.

Argomenti

Fase 1: Valuta le prestazioni del tuo modello

È possibile accedere alle metriche delle prestazioni dalla console e dall'DescribeModeloperazione. Amazon Lookout for Vision fornisce metriche riassuntive delle prestazioni per il set di dati di test e i risultati previsti per tutte le singole immagini. Se il modello è un modello di segmentazione, la console mostra anche le metriche di riepilogo per ogni etichetta di anomalia.

Per visualizzare le metriche delle prestazioni e testare le previsioni delle immagini nella console, consultaVisualizzazione dei parametri relativi alle prestazioni (console). Per informazioni sull'accesso alle metriche delle prestazioni e alle previsioni delle immagini di prova con l'`DescribeModel`operazione, vedereVisualizzazione dei parametri relativi alle prestazioni (SDK).

Metriche di classificazione delle immagini

Amazon Lookout for Vision fornisce le seguenti metriche riassuntive per le classificazioni effettuate da un modello durante i test:

- Precisione
- Recall
- Punteggio F1

Metriche del modello di segmentazione delle immagini

Se il modello è un modello di segmentazione delle immagini, Amazon Lookout for Vision fornisce metriche riassuntive di classificazione delle immagini e metriche riassuntive delle prestazioni per ogni etichetta di anomalia:

- Punteggio F1
- Intersezione media su Union (IoU)

Precisione

La metrica di precisione risponde alla domanda: quando il modello prevede che un'immagine contiene un'anomalia, con che frequenza tale previsione è corretta?

La precisione è una metrica utile per le situazioni in cui il costo di un falso positivo è elevato. Ad esempio, il costo della rimozione di una parte della macchina che non è difettosa da una macchina assemblata.

Amazon Lookout for Vision fornisce un valore metrico di precisione riepilogativo per l'intero set di dati di test.

La precisione è la frazione di anomalie correttamente previste (veri positivi) rispetto a tutte le anomalie previste (veri e falsi positivi). La formula per la precisione è la seguente.

Valore di precisione = veri positivi/(veri positivi+falsi positivi)

I valori possibili per la precisione vanno da 0 a 1. La console Amazon Lookout for Vision mostra la precisione come valore percentuale (0-100).

Un valore di precisione più elevato indica che la maggior parte delle anomalie previste sono corrette. Ad esempio, supponiamo che il modello preveda che 100 immagini siano anomale. Se 85 delle previsioni sono corrette (i veri positivi) e 15 sono errate (i falsi positivi), la precisione viene calcolata come segue:

85 veri positivi/ (85 veri positivi+15 falsi positivi) = 0,85 valore di precisione

Tuttavia, se il modello prevede correttamente solo 40 immagini su 100 previsioni di anomalie, il valore di precisione risultante è inferiore a 0,40 (ovvero 40/(40 + 60) = 0,40). In questo caso, il modello sta facendo più previsioni errate che previsioni corrette. Per risolvere questo problema, valuta la

possibilità di apportare miglioramenti al tuo modello. Per ulteriori informazioni, consulta Fase 2: miglioramento del modello.

Per ulteriori informazioni, consulta la voce Precisione e recupero.

Recall

La metrica di richiamo risponde alla domanda: del numero totale di immagini anomale nel set di dati del test, quante sono correttamente previste come anomale?

La metrica di richiamo è utile per le situazioni in cui il costo di un falso negativo è elevato. Ad esempio, quando il costo della mancata rimozione di una parte difettosa è elevato. Amazon Lookout for Vision fornisce un valore metrico di richiamo riepilogativo per l'intero set di dati di test.

Il richiamo è la frazione delle immagini di test anomale che sono state rilevate correttamente. È una misura della frequenza con cui il modello è in grado di prevedere correttamente un'immagine anomala, quando è effettivamente presente nelle immagini del set di dati di test. La formula per il recupero viene calcolata come segue:

Valore di richiamo = veri positivi/(veri positivi+falsi negativi)

L'intervallo di richiamo è compreso tra 0 e 1. La console Amazon Lookout for Vision visualizza il richiamo come valore percentuale (0-100).

Un valore di richiamo più elevato indica che la maggior parte delle immagini anomale sono state identificate correttamente. Ad esempio, supponiamo che il set di dati di test contenga 100 immagini anomale. Se il modello rileva correttamente 90 delle 100 immagini anomale, il richiamo è il seguente:

90 veri positivi/ (90 veri positivi+10 falsi negativi) = 0,90 valore di richiamo

Un valore di richiamo di 0,90 indica che il modello prevede correttamente la maggior parte delle immagini anomale nel set di dati di test. Se il modello prevede correttamente solo 20 delle immagini anomale, il richiamo è inferiore a 0,20 (ovvero 20/(20 + 80) = 0,20).

In questo caso, è necessario considerare l'opportunità di apportare miglioramenti al modello. Per ulteriori informazioni, consulta Fase 2: miglioramento del modello.

Per ulteriori informazioni, consulta la voce Precisione e recupero.

Punteggio F1

Amazon Lookout for Vision fornisce un punteggio medio delle prestazioni del modello per il set di dati di test. In particolare, le prestazioni del modello per la classificazione delle anomalie vengono misurate dalla metrica del punteggio F1, che è la media armonica dei punteggi di precisione e richiamo.

Il punteggio F1 è una misura aggregata che tiene conto sia della precisione che del richiamo. Il punteggio relativo alle prestazioni del modello è un valore compreso tra 0 e 1. Più alto è il valore, migliori sono le prestazioni del modello sia in termini di richiamo che di precisione. Ad esempio, per un modello con una precisione di 0,9 e un richiamo di 1,0, il punteggio F1 è 0,947.

Se il modello non funziona bene, ad esempio, con una bassa precisione di 0,30 e un richiamo elevato di 1,0, il punteggio F1 è 0,46. Analogamente, se la precisione è alta (0,95) e il richiamo è basso (0,20), il punteggio F1 è 0,33. In entrambi i casi, il punteggio F1 è basso, il che indica problemi con il modello.

Per ulteriori informazioni consulta la voce F1 score.

Intersezione media su Union (IoU)

La percentuale media di sovrapposizione tra le maschere di anomalia nelle immagini di test e le maschere di anomalia previste dal modello per le immagini di prova. Amazon Lookout for Vision restituisce l'IoU medio per ogni etichetta di anomalia e viene restituito solo dai modelli di segmentazione delle immagini.

Un valore percentuale basso indica che il modello non corrisponde accuratamente alle maschere previste per un'etichetta con le maschere nelle immagini di prova.

L'immagine seguente ha un IOU basso. La maschera arancione è la previsione del modello e non copre strettamente la maschera blu che rappresenta la maschera in un'immagine di prova.

L'immagine seguente ha un IOU più alto. La maschera blu (immagine di prova) è strettamente coperta dalla maschera arancione (maschera prevista).

Risultati dei test

Durante il test, il modello prevede la classificazione per ogni immagine di prova nel set di dati del test. Il risultato di ogni previsione viene confrontato con l'etichetta (normale o anomalia) dell'immagine di prova corrispondente come segue:

- Prevedere correttamente che un'immagine è anomala è considerato un vero positivo.
- La previsione errata che un'immagine sia anomala è considerata un falso positivo.
- Prevedere correttamente che un'immagine è normale è considerato un vero negativo.
- La previsione errata che un'immagine sia normale è considerata un falso negativo.

Se il modello è un modello di segmentazione, prevede anche maschere ed etichette di anomalia per la posizione delle anomalie sull'immagine di test.

Amazon Lookout for Vision utilizza i risultati dei confronti per generare le metriche delle prestazioni.

Fase 2: miglioramento del modello

Le metriche delle prestazioni potrebbero mostrare che puoi migliorare il tuo modello. Ad esempio, se il modello non rileva tutte le anomalie nel set di dati del test, il modello ha un richiamo basso (ovvero, la metrica di richiamo ha un valore basso). Per migliorare il modello, considera quanto segue:

- Verifica che le immagini dei set di dati di addestramento e test siano etichettate correttamente.
- Riduci la variabilità delle condizioni di acquisizione delle immagini, come l'illuminazione e la posa degli oggetti, e addestra il tuo modello su oggetti dello stesso tipo.

- Assicurati che le tue immagini mostrino solo il contenuto richiesto. Ad esempio, se il progetto rileva anomalie nelle parti della macchina, assicurati che non ci siano altri oggetti nelle immagini.
- Aggiungi altre immagini etichettate al tuo treno e ai set di dati di prova. Se il set di dati di test ha un richiamo e una precisione eccellenti ma il modello funziona male quando viene distribuito, il set di dati di test potrebbe non essere sufficientemente rappresentativo ed è necessario estenderlo.
- Se il tuo set di dati di test risulta scarso in termini di richiamo e precisione, considera la corrispondenza tra le anomalie e le condizioni di acquisizione delle immagini nei set di dati di addestramento e di test. Se le immagini di allenamento non sono rappresentative delle anomalie e delle condizioni previste, ma le immagini nelle immagini di test lo sono, aggiungi immagini al set di dati di addestramento con le anomalie e le condizioni previste. Se le immagini del set di dati di test non sono nelle condizioni previste, ma le immagini di addestramento sì, aggiorna il set di dati di test.

  Per ulteriori informazioni, consulta Aggiungere altre immagini. Un modo alternativo per aggiungere immagini etichettate al set di dati di allenamento consiste nell'eseguire un'attività di rilevamento di prova e verificare i risultati. È quindi possibile aggiungere le immagini verificate al set di dati di addestramento. Per ulteriori informazioni, consulta Verifica del modello con un'attività di rilevamento delle versioni di prova.
- Assicurati di avere immagini normali e anomale sufficientemente diverse nel set di dati di allenamento e test. Le immagini devono rappresentare il tipo di immagini normali e anomale che il modello incontrerà. Ad esempio, quando si analizzano i circuiti stampati, le immagini normali devono rappresentare le variazioni di posizione e di saldatura dei componenti, come resistori e transistor. Le immagini anomale devono rappresentare i diversi tipi di anomalie che il sistema potrebbe riscontrare, ad esempio componenti fuori posto o mancanti.
- Se il tuo modello ha un IOU medio basso per i tipi di anomalia rilevati, controlla gli output della maschera del modello di segmentazione. In alcuni casi d'uso, ad esempio i graffi, il modello potrebbe produrre graffi molto simili a quelli reali nelle immagini di prova, ma con una sovrapposizione di pixel ridotta. Ad esempio, due linee parallel distanti 1 pixel l'una dall'altra. In questi casi, Average IOU è un indicatore inaffidabile per misurare il successo della previsione.
- Se le dimensioni dell'immagine sono ridotte o la risoluzione dell'immagine è bassa, prendi in considerazione l'acquisizione di immagini con una risoluzione più elevata. Le dimensioni dell'immagine possono variare da 64 x 64 pixel a 4096 pixel X 4096 pixel.
- Se la dimensione dell'anomalia è ridotta, valuta la possibilità di dividere le immagini in riquadri separati e di utilizzare le immagini affiancate per l'addestramento e i test. Ciò consente al modello di vedere i difetti di dimensioni maggiori in un'immagine.

Dopo aver migliorato il set di dati di addestramento e test, riaddestrate e rivalutate il modello. Per ulteriori informazioni, consulta Addestrare il modello.

Se le metriche mostrano che il modello ha prestazioni accettabili, puoi verificarne le prestazioni aggiungendo i risultati di un'attività di rilevamento di prova al set di dati di test. Dopo la riqualificazione, le metriche delle prestazioni dovrebbero confermare le metriche delle prestazioni dell'allenamento precedente. Per ulteriori informazioni, consulta Verifica del modello con un'attività di rilevamento delle versioni di prova.

Visualizzazione dei parametri relativi alle prestazioni

Puoi ottenere le metriche delle prestazioni dalla console e chiamando l'`DescribeModel`operazione.

Argomenti

- Visualizzazione dei parametri relativi alle prestazioni (console)
- Visualizzazione dei parametri relativi alle prestazioni (SDK)

Visualizzazione dei parametri relativi alle prestazioni (console)

Al termine dell'allenamento, la console visualizza le metriche delle prestazioni.

La console Amazon Lookout for Vision mostra le seguenti metriche prestazionali per le classificazioni effettuate durante i test:

- Precisione
- Recall
- Punteggio F1

Se il modello è un modello di segmentazione, la console mostra anche le seguenti metriche prestazionali per ciascuna etichetta di anomalia:

- Il numero di immagini di prova in cui è stata trovata l'etichetta dell'anomalia.
- Punteggio F1
- Intersezione media su Union (IoU)

La sezione panoramica dei risultati del test mostra le previsioni totali corrette ed errate per le immagini nel set di dati del test. Puoi anche vedere le assegnazioni di etichette previste ed effettive per le singole immagini nel set di dati di test.

La procedura seguente mostra come ottenere i parametri relativi alle prestazioni dalla visualizzazione dell'elenco modello di un progetto.

Come visualizzare i parametri relativi alle prestazioni (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegliere Inizia.
3. Nel pannello di navigazione a sinistra, scegliere Progetti.
4. Nella vista Progetti, scegli il progetto contenente la versione del modello che desideri visualizzare.
5. Nel pannello di navigazione a sinistra, sotto il nome del progetto, scegliere Modelli.
6. Nella visualizzazione dell'elenco dei modelli, scegliere la versione del modello che si desidera modificare.
7. Nella pagina dei dettagli del modello, visualizza le metriche delle prestazioni nella scheda Metriche delle prestazioni.
8. Tieni presente quanto segue:

    a. La sezione Metriche delle prestazioni del modello contiene le metriche complessive del modello (precisione, richiamo, punteggio F1) per le previsioni di classificazione effettuate dal modello per le immagini di test.

    b. Se il modello è un modello di segmentazione delle immagini, la sezione Prestazioni per etichetta contiene il numero di immagini di test in cui è stata trovata l'etichetta dell'anomalia. Vengono visualizzate anche le metriche (punteggio F1, IoU medio) per ogni etichetta di anomalia.

    c. La sezione Panoramica dei risultati dei test fornisce i risultati per ogni immagine di test utilizzata da Lookout for Vision per valutare il modello. Include le voci seguenti:

        - Il numero totale di previsioni di classificazione corrette (vere positive) e errate (false negative) (normali o anomalie) per tutte le immagini del test.

- La previsione di classificazione per ogni immagine di prova. Se vedi Corretto sotto un'immagine, la classificazione prevista corrisponde alla classificazione effettiva dell'immagine. Altrimenti il modello non ha classificato correttamente l'immagine.
- Con un modello di segmentazione dell'immagine, vengono visualizzate le etichette di anomalia assegnate dal modello all'immagine e le maschere sull'immagine che corrispondono ai colori delle etichette di anomalia.

Visualizzazione dei parametri relativi alle prestazioni (SDK)

È possibile utilizzare l'DescribeModeloperazione per ottenere le metriche riassuntive delle prestazioni (classificazione) per il modello, il manifesto di valutazione e i risultati della valutazione per un modello.

Ottenere le metriche riassuntive delle prestazioni

Le metriche riassuntive delle prestazioni per le previsioni di classificazione effettuate dal modello durante i test (PrecisioneRecall, ePunteggio F1) vengono restituite nel`Performance` campo restituito da una chiamata a`DescribeModel`.

```
        "Performance": {
            "F1Score": 0.8,
            "Recall": 0.8,
            "Precision": 0.9
        },
```

Il`Performance` campo non include le metriche sulle prestazioni delle etichette di anomalia restituite da un modello di segmentazione. Puoi prenderli dal`EvaluationResult` campo. Per ulteriori informazioni, consulta Esame del risultato della valutazione.

Per informazioni sulle metriche di riepilogo delle prestazioni, vedereFase 1: Valuta le prestazioni del tuo modello. Per il codice di esempio, consulta Visualizzazione dei modelli (SDK).

Utilizzo del manifesto di valutazione

Il manifesto di valutazione fornisce le metriche di previsione dei test per le singole immagini utilizzate per testare un modello. Per ogni immagine nel set di dati del test, una riga JSON contiene le informazioni del test originale (verità fondamentale) e la previsione del modello per l'immagine. Amazon Lookout for Vision archivia il manifesto di valutazione in un bucket Amazon S3. È possibile ottenere la posizione dal`EvaluationManifest` campo nella risposta dell'`DescribeModel`operazione.

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

Il formato del nome del file è`EvaluationManifest-`*`project name`*`.json`. Per il codice di esempio, consulta Visualizzazione dei modelli.

Nella seguente riga JSON di esempio, questa`class-name` è la verità fondamentale per il contenuto dell'immagine. In questo esempio l'immagine contiene un'anomalia. Il`confidence` campo mostra la fiducia che Amazon Lookout for Vision ripone nella previsione.

```
{
    "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"
    }
```

```
}
```

Esame del risultato della valutazione

Il risultato della valutazione presenta le seguenti metriche aggregate delle prestazioni (classificazione) per l'intero set di immagini di test:

- Precisione
- Recall
- Curva ROC (non mostrata nella console)
- Precisione media (non mostrata nella console)
- Punteggio F1

Il risultato della valutazione include anche il numero di immagini utilizzate per addestrare e testare il modello.

Se il modello è un modello di segmentazione, il risultato della valutazione include anche le seguenti metriche per ogni etichetta di anomalia trovata nel set di dati del test:

- Precisione
- Recall
- Punteggio F1
- Intersezione media su Union (IoU)

Amazon Lookout for Vision archivia il risultato della valutazione in un bucket Amazon S3. È possibile ottenere la posizione controllando il valore di`EvaluationResult` nella risposta dell'`DescribeModel`operazione.

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

Il formato del nome del file è`EvaluationResult-`*`project name`*`.json`. Per un esempio, consulta Visualizzazione dei modelli.

Lo schema seguente mostra il risultato della valutazione.

```
   {
       "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.
                   }
               ]
           }
       }
   }
```

Verifica del modello con un'attività di rilevamento delle versioni di prova

Se desideri verificare o migliorare la qualità del tuo modello, puoi eseguire un'attività di rilevamento di prova. Un'attività di rilevamento di prova rileva le anomalie nelle nuove immagini fornite.

Puoi verificare i risultati del rilevamento e aggiungere le immagini verificate al tuo set di dati. Se disponi di set di dati di addestramento e test separati, le immagini verificate vengono aggiunte al set di dati di addestramento.

Puoi verificare le immagini dal computer locale o quelle che si trovano in un bucket Amazon S3. Se desideri aggiungere immagini verificate al set di dati, le immagini che si trovano in un bucket S3 devono trovarsi nello stesso bucket S3 delle immagini nel set di dati.

> **Note**
>
> Per eseguire un'attività di rilevamento delle versioni di prova, assicurati che il bucket S3 abbia abilitato il controllo delle versioni. Per ulteriori informazioni consulta Uso del controllo Versioni multiple. Il bucket della console viene creato con il controllo delle versioni abilitato.

Per impostazione predefinita, le immagini sono crittografate con una chiave che AWS possiede e gestisce. Puoi anche scegliere di utilizzare la tua chiave AWS Key Management Service (KMS). Per ulteriori informazioni consulta Concetti relativi a AWS Key Management Service.

Argomenti

- Esecuzione di un'attività di rilevamento delle prove
- Verifica dei risultati del rilevamento delle sperimentazioni
- Correzione delle etichette di segmentazione con lo strumento di annotazione

Esecuzione di un'attività di rilevamento delle prove

Eseguire la procedura seguente per eseguire un'attività di rilevamento della prova.

Per eseguire un rilevamento della versione di prova (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.

2. Scegliere Inizia.
3. Nel pannello di navigazione a sinistra, scegliere Progetti.
4. Nella vista Progetti, scegli il progetto contenente la versione del modello che desideri visualizzare.
5. Nel pannello di navigazione a sinistra, sotto il nome del progetto, scegliere Rilevamenti di prova.
6. Nella vista dei rilevamenti delle versioni di prova, scegli Esegui il rilevamento della versione di prova.
7. Nella pagina Esegui il rilevamento delle versioni di prova, inserisci un nome per l'attività di rilevamento delle versioni di prova in Nome attività.
8. In Scegli modello, scegli la versione di quel modello che desideri utilizzare.
9. Importa le immagini in base alla fonte delle immagini come segue:

   - Se stai importando le immagini sorgente da un bucket Amazon S3, inserisci l'URI S3.

     > **Tip**
     >
     > Se stai usando le immagini di esempio Getting Started, usa la cartella extra_images. L'URI Amazon S3 è`s3://your bucket/circuitboard/extra_images`.

   - Se stai caricando immagini dal tuo computer, aggiungile dopo aver scelto Rileva anomalie.
10. (Facoltativo) Se desideri utilizzare la tua chiave di crittografia AWS KMS, procedi come segue:

    a. Per la crittografia dei dati delle immagini, scegli Personalizza le impostazioni di crittografia (avanzate).

    b. In encryption.aws_kms_key, immetti l'Amazon Resource Name (ARN) della chiavi o scegli una chiave AWS KMS esistente. Per creare una nuova chiave, scegli Crea una chiave AWS IMS.
11. Scegli Rileva anomalie e quindi scegli Esegui il rilevamento della versione di prova per avviare l'attività di rilevamento della versione di prova.
12. Controlla lo stato attuale nella visualizzazione dei rilevamenti delle prove. Il completamento del rilevamento della prova potrebbe richiedere un po' di tempo.

Verifica dei risultati del rilevamento delle sperimentazioni

La verifica dei risultati di una sperimentazione può aiutarti a migliorare il tuo modello.

Se le metriche delle prestazioni sono scadenti, migliora il modello eseguendo un rilevamento di prova e quindi aggiungi immagini verificate al set di dati (set di dati di addestramento, se disponi di set di dati separati).

Se le metriche prestazionali del modello sono buone, ma i risultati di una prova di rilevamento sono scadenti, puoi migliorare il tuo modello aggiungendo immagini verificate al set di dati (set di dati di addestramento). Se disponi di un set di dati di test separato, considera l'aggiunta di altre immagini al set di dati di test.

Dopo aver aggiunto immagini verificate al set di dati, riaddestrate e rivalutate il modello. Per ulteriori informazioni, consulta Addestrare il modello.

Per verificare i risultati di una sperimentazione

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Nel pannello di navigazione a sinistra, scegliere Progetti.
3. Nella pagina Progetti, scegli il progetto che desideri usare. Viene visualizzata la dashboard del progetto.
4. Nel pannello di navigazione a sinistra, seleziona Rilevamenti di prova.
5. Scegli il rilevamento della prova che desideri verificare.
6. Nella pagina di rilevamento delle versioni di prova, scegli Verifica le previsioni della macchina.
7. Scegli Seleziona tutte le immagini in questa pagina.
8. Se le previsioni sono corrette, scegli Verifica come corrette. Altrimenti, scegli Verifica come errato. Il punteggio di affidabilità della previsione e della previsione è mostrato sotto ogni immagine.
9. Per modificare l'etichetta di un'immagine, procedi nel seguente modo:
    a. Scegli Corretto o Non corretto sotto l'immagine.
    b. Se non riesci a determinare l'etichetta corretta per un'immagine, ingrandisci l'immagine scegliendo l'immagine nella galleria.

> **Note**
>
> Puoi filtrare le etichette delle immagini scegliendo l'etichetta o lo stato dell'etichetta desiderata nella sezione Filtri. Puoi ordinare in base al punteggio di confidenza nella sezione Opzioni di ordinamento.

10. Se il tuo modello è un modello di segmentazione e la maschera o l'etichetta di anomalia per un'immagine è sbagliata, scegli Area anomala sotto l'immagine e apri lo strumento di annotazione. Aggiorna le informazioni sulla segmentazione eseguendoCorrezione delle etichette di segmentazione con lo strumento di annotazione.
11. Se necessario, ripeti i passaggi 7-10 su ogni pagina fino a quando tutte le immagini non sono state verificate.
12. Scegli Aggiungi immagini verificate al set di dati. Se disponi di set di dati separati, le immagini vengono aggiunte al set di dati di addestramento.
13. Riqualifica il tuo modello. Per ulteriori informazioni, consulta the section called “Addestrare il modello”.

Correzione delle etichette di segmentazione con lo strumento di annotazione

Si utilizza lo strumento di annotazione per segmentare un'immagine contrassegnando le aree anomale con una maschera.

Per correggere le etichette di segmentazione di un'immagine con lo strumento di annotazione

1. Apri lo strumento di annotazione selezionando l'area anomala sotto un'immagine nella galleria del set di dati.
2. Se l'etichetta di anomalia per una maschera non è corretta, scegli la maschera e quindi scegli l'etichetta di anomalia corretta in Etichette di anomalia. Se necessario, scegli Aggiungi etichetta di anomalia per aggiungere una nuova etichetta di anomalia.
3. Se la maschera non è corretta, scegli uno strumento di disegno nella parte inferiore della pagina e disegna maschere che coprano bene le aree anomale per l'etichetta dell'anomalia. L'immagine seguente è un esempio di maschera che copre perfettamente un'anomalia.

Di seguito è riportato un esempio di maschera scadente che non copre bene un'anomalia.

4. Se hai altre immagini da correggere, scegli Avanti e ripeti i passaggi 2 e 3.
5. Scegli Invia e chiudi per completare l'aggiornamento delle immagini.

Esecuzione del modello Amazon Lookout for Vision addestrato

Per rilevare anomalie nelle immagini con il modello, è necessario innanzitutto avviare il modello con l'StartModeloperazione. La console Amazon Lookout for Vision AWS CLI fornisce comandi che puoi usare per avviare e arrestare il modello. Questa sezione include un codice di esempio che puoi usare.

Dopo l'avvio del modello, è possibile utilizzare l'`DetectAnomalies`operazione per rilevare anomalie in un'immagine. Per ulteriori informazioni, consulta Rilevamento di anomalie in un'immagine.

Argomenti

Unità di inferenza

Quando avvii il tuo modello, Amazon Lookout for Vision fornisce almeno una risorsa di calcolo, nota come unità di inferenza. Specifichi il numero di unità di inferenza da utilizzare nel parametro di `MinInferenceUnits` input dell'API. `StartModel` L'allocazione predefinita per un modello è 1 unità di inferenza.

> **Important**
>
> Ti viene addebitato il numero di ore di funzionamento del modello e il numero di unità di inferenza utilizzate dal modello durante l'esecuzione, in base a come configuri l'esecuzione del modello. Ad esempio, se avvii il modello con due unità di inferenza e lo utilizzi per 8 ore, ti verranno addebitate 16 ore di inferenza (8 ore di esecuzione* due unità di inferenza). Per ulteriori informazioni, consulta la pagina dei prezzi di Amazon Lookout for Vision. Se non interrompi esplicitamente il modello chiamando StopModel, ti verrà addebitato un costo anche se non stai analizzando attivamente le immagini con il tuo modello.

Le transazioni al secondo (TPS) supportate da una singola unità di inferenza sono influenzate da quanto segue:

- L'algoritmo utilizzato da Lookout for Vision per addestrare il modello. Quando si addestra un modello, vengono addestrati più modelli. Lookout for Vision seleziona il modello con le migliori prestazioni in base alla dimensione del set di dati e alla sua composizione di immagini normali e anomale.
- Le immagini a risoluzione più elevata richiedono più tempo per l'analisi.
- Le immagini di dimensioni più piccole (misurate in MB) vengono analizzate più rapidamente delle immagini più grandi.

Gestione della velocità effettiva con unità di inferenza

È possibile aumentare o diminuire la velocità effettiva del modello in base alle esigenze dell'applicazione. Per aumentare la velocità effettiva, utilizzate unità di inferenza aggiuntive. Ogni unità di inferenza aggiuntiva aumenta la velocità di elaborazione di un'unità di inferenza. Per informazioni sul calcolo del numero di unità di inferenza necessarie, consulta Calcolare le unità di inferenza per i modelli Amazon Rekognition Custom Labels e Amazon Lookout for Vision. Se desideri modificare la velocità effettiva supportata dal tuo modello, hai due opzioni:

Aggiungere o rimuovere manualmente le unità di inferenza

Arrestate il modello e riavviatelo con il numero richiesto di unità di inferenza. Lo svantaggio di questo approccio è che il modello non può ricevere richieste durante il riavvio e non può essere utilizzato per gestire i picchi di domanda. Utilizza questo approccio se il tuo modello ha un throughput costante e il tuo caso d'uso può tollerare 10-20 minuti di inattività. Un esempio potrebbe essere se desideri eseguire chiamate in batch al modello utilizzando una pianificazione settimanale.

Unità di inferenza con scalabilità automatica

Se il tuo modello deve far fronte ai picchi di domanda, Amazon Lookout for Vision può ridimensionare automaticamente il numero di unità di inferenza utilizzate dal modello. All'aumentare della domanda, Amazon Lookout for Vision aggiunge unità di inferenza aggiuntive al modello e le rimuove quando la domanda diminuisce.

Per consentire a Lookout for Vision di scalare automaticamente le unità di inferenza per un modello, avvia il modello e imposta il numero massimo di unità di inferenza che può utilizzare utilizzando il parametro. `MaxInferenceUnits` L'impostazione di un numero massimo di unità di inferenza

consente di gestire i costi di esecuzione del modello limitando il numero di unità di inferenza disponibili. Se non specifichi un numero massimo di unità, Lookout for Vision non ridimensionerà automaticamente il modello, ma utilizzerà solo il numero di unità di inferenza con cui hai iniziato. Per informazioni sul numero massimo di unità di inferenza, vedere Service Quotas.

È inoltre possibile specificare un numero minimo di unità di inferenza utilizzando il parametro. `MinInferenceUnits` Ciò consente di specificare il throughput minimo per il modello, dove una singola unità di inferenza rappresenta 1 ora di tempo di elaborazione.

> **Note**
>
> Non è possibile impostare il numero massimo di unità di inferenza con la console Lookout for Vision. Specificate invece il parametro `MaxInferenceUnits` di input per l'`StartModel`operazione.

Lookout for Vision fornisce le seguenti metriche di CloudWatch Amazon Logs che puoi utilizzare per determinare lo stato corrente del ridimensionamento automatico di un modello.

Parametro	Descrizione
`DesiredInferenceUnits`	Il numero di unità di inferenza a cui Lookout for Vision viene ridimensionato verso l'alto o verso il basso.
`InServiceInferenceUnits`	Il numero di unità di inferenza utilizzate dal modello.

If `DesiredInferenceUnits` =`InServiceInferenceUnits`, Lookout for Vision non sta attualmente ridimensionando il numero di unità di inferenza.

Se `DesiredInferenceUnits` >`InServiceInferenceUnits`, Lookout for Vision sta aumentando fino al valore di. `DesiredInferenceUnits`

Se `DesiredInferenceUnits` <`InServiceInferenceUnits`, Lookout for Vision viene ridimensionato al valore di. `DesiredInferenceUnits`

Per ulteriori informazioni sulle metriche restituite da Lookout for Vision e sulle dimensioni di filtraggio, consulta Monitoring Lookout for Vision with Amazon. CloudWatch

Per scoprire il numero massimo di unità di inferenza che hai richiesto per un modello, chiama DescribeModele controlla il `MaxInferenceUnits` campo nella risposta.

Zone di disponibilità

Amazon Lookout for Vision; distribuisce unità di inferenza su più zone di disponibilità all'interno di AWS una regione per fornire una maggiore disponibilità. Per ulteriori informazioni, consulta Zone di disponibilità. Per proteggere i modelli di produzione dalle interruzioni della zona di disponibilità e dai guasti delle unità di inferenza, avvia i modelli di produzione con almeno due unità di inferenza.

Se si verifica un'interruzione della zona di disponibilità, tutte le unità di inferenza nella zona di disponibilità non sono disponibili e la capacità del modello viene ridotta. Le chiamate a DetectAnomaliesvengono ridistribuite tra le unità di inferenza rimanenti. Tali chiamate hanno esito positivo se non superano le transazioni per secondi (TPS) supportate delle unità di inferenza rimanenti. Dopo aver AWS riparato la zona di disponibilità, le unità di inferenza vengono riavviate e viene ripristinata la piena capacità.

Se una singola unità di inferenza si guasta, Amazon Lookout for Vision avvia automaticamente una nuova unità di inferenza nella stessa zona di disponibilità. La capacità del modello viene ridotta fino all'avvio della nuova unità di inferenza.

Avvio del modello Amazon Lookout for Vision

Prima di poter utilizzare un modello Amazon Lookout for Vision per rilevare anomalie, devi prima avviare il modello. Puoi avviare un modello chiamando l'StartModelAPI e passando quanto segue:

- ProjectName— Il nome del progetto che contiene il modello che desiderate avviare.
- ModelVersion— La versione del modello che si desidera avviare.
- MinInferenceUnits— Il numero minimo di unità di inferenza. Per ulteriori informazioni, consulta Unità di inferenza.
- (Facoltativo) MaxInferenceUnits: il numero massimo di unità di inferenza che Amazon Lookout for Vision può utilizzare per ridimensionare automaticamente il modello. Per ulteriori informazioni, consulta Unità di inferenza con scalabilità automatica.

La console Amazon Lookout for Vision fornisce codice di esempio che puoi usare per avviare e interrompere un modello.

> **Note**
>
> Ti viene addebitato il tempo di funzionamento del modello. Per interrompere l'esecuzione di un modello, vedereInterruzione del modello Amazon Lookout for Vision.
> Puoi utilizzare l'AWSSDK per visualizzare i modelli in esecuzione in tutte le AWS regioni in cui è disponibile Lookout for Vision. Per esempio di codice, vedi find_running_models.py.

Argomenti

Avvio del modello (console)

La console Amazon Lookout for Vision fornisce AWS CLI un comando che puoi usare per avviare un modello. Dopo l'avvio del modello, puoi iniziare a rilevare anomalie nelle immagini. Per ulteriori informazioni, consulta Rilevamento di anomalie in un'immagine.

Per avviare un modello (console)

1. Se non l'hai già fatto, installa e configura gli AWS CLI AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
3. Scegli Inizia.
4. Nel riquadro di navigazione a sinistra, scegli Progetti.
5. Nella pagina delle risorse dei progetti, scegli il progetto che contiene il modello addestrato che desideri avviare.
6. Nella sezione Modelli, scegli il modello che desideri avviare.
7. Nella pagina dei dettagli del modello, scegli Usa modello, quindi scegli Integrate API to the cloud.

> **Tip**
>
> Se desideri distribuire il tuo modello su un dispositivo edge, scegli Create model packaging job. Per ulteriori informazioni, consulta Imballaggio del modello Amazon Lookout for Vision.

8. Nei comandi CLI di AWS, copia il comando AWS CLI che chiama. `start-model`
9. Al prompt dei comandi, inserisci il `start-model` comando che hai copiato nel passaggio precedente. Se utilizzate il `lookoutvision` profilo per ottenere le credenziali, aggiungete il parametro. `--profile lookoutvision-access`
10. Nella console, scegli Modelli nella pagina di navigazione a sinistra.
11. Controlla la colonna Stato per lo stato attuale del modello. Quando lo stato è Ospitato, puoi utilizzare il modello per rilevare anomalie nelle immagini. Per ulteriori informazioni, consulta Rilevamento di anomalie in un'immagine.

Avvio del modello Amazon Lookout for Vision (SDK)

Si avvia un modello chiamando l'StartModeloperazione.

L'avvio di un modello potrebbe richiedere alcuni istanti. Puoi controllare lo stato attuale chiamando DescribeModel. Per ulteriori informazioni, consulta Visualizzazione dei modelli.

Per avviare il modello (SDK)

1. Se non l'hai già fatto, installa e configura gli AWS CLI AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il seguente codice di esempio per avviare un modello.

   CLI

   Modificate i seguenti valori:

   - `project-name`al nome del progetto che contiene il modello che desiderate avviare.
   - `model-version`alla versione del modello che si desidera avviare.
   - `--min-inference-units`sal numero di unità di inferenza che si desidera utilizzare.

- (Facoltativo) `--max-inference-units` fino al numero massimo di unità di inferenza che Amazon Lookout for Vision può utilizzare per ridimensionare automaticamente il modello.

```
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

Questo codice è tratto dall'archivio degli esempi di AWS Documentation SDK. GitHub Vedi l'esempio completo qui.

```
    @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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

```
/**
 * 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. Se l'output del codice è`Model is hosted and ready for use`, puoi utilizzare il modello per rilevare anomalie nelle immagini. Per ulteriori informazioni, consulta Rilevamento di anomalie in un'immagine.

Interruzione del modello Amazon Lookout for Vision

Per interrompere un modello in esecuzione, chiamate l'`StopModel`operazione e passate quanto segue:

- Progetto: il nome del progetto che contiene il modello che desiderate interrompere.
- ModelVersion— La versione del modello che desiderate interrompere.

La console Amazon Lookout for Vision fornisce un codice di esempio che puoi usare per interrompere un modello.

> **Note**
>
> Ti viene addebitato il tempo di funzionamento del modello.

Argomenti

Arresto del modello (console)

Eseguite i passaggi indicati nella procedura seguente per interrompere l'utilizzo della console da parte del modello.

Per arrestare il modello (console)

1. Se non l'hai già fatto, installa e configura gli AWS CLI AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
3. Scegli Inizia.
4. Nel riquadro di navigazione a sinistra, scegli Progetti.
5. Nella pagina delle risorse dei progetti, scegli il progetto che contiene il modello in esecuzione che desideri interrompere.
6. Nella sezione Modelli, scegli il modello che desideri interrompere.
7. Nella pagina dei dettagli del modello, scegli Usa modello, quindi scegli Integrate API to the cloud.
8. Nei comandi CLI di AWS, copia il comando AWS CLI che chiama. `stop-model`
9. Al prompt dei comandi, inserisci il `stop-model` comando che hai copiato nel passaggio precedente. Se utilizzate il `lookoutvision` profilo per ottenere le credenziali, aggiungete il parametro. `--profile lookoutvision-access`
10. Nella console, scegli Modelli nella pagina di navigazione a sinistra.
11. Controlla la colonna Status per lo stato attuale del modello. Il modello si è fermato quando il valore della colonna Status è Training complete.

Interruzione del modello Amazon Lookout for Vision (SDK)

Si arresta un modello chiamando l'StopModeloperazione.

L'interruzione di un modello potrebbe richiedere alcuni istanti. Per verificare lo stato attuale, usa`DescribeModel`.

Per interrompere il modello (SDK)

1. Se non l'hai già fatto, installa e configura gli AWS CLI AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il seguente codice di esempio per interrompere un modello in esecuzione.

   CLI

   Modificate i seguenti valori:

   - `project-name`al nome del progetto che contiene il modello che desiderate interrompere.
   - `model-version`alla versione del modello che desiderate interrompere.

```
aws lookoutvision stop-model --project-name "project name"\
    --model-version model version \
  --profile lookoutvision-access
```

   Python

   Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

```
    @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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

```
/**
 * 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;

}
```

Rilevamento di anomalie in un'immagine

Per rilevare anomalie in un'immagine con un modello Amazon Lookout for Vision addestrato, chiamiDetectAnomaliesoperazione. Il risultato di`DetectAnomalies`include una previsione booleana che classifica l'immagine come contenente una o più anomalie e un valore di affidabilità per la previsione. Se il modello è un modello di segmentazione dell'immagine, il risultato include anche una maschera colorata che mostra le posizioni di diversi tipi di anomalie.

Le immagini che fornisci a`DetectAnomalies`deve avere le stesse dimensioni di larghezza e altezza delle immagini utilizzate per addestrare il modello.

`DetectAnomalies`accetta immagini in formato PNG o JPG. È consigliabile che le immagini abbiano lo stesso formato di codifica e compressione utilizzato per addestrare il modello. Ad esempio, se addestrate il modello con immagini in formato PNG, chiamate`DetectAnomalies`con immagini in formato PNG.

Prima di chiamare`DetectAnomalies`, devi prima avviare il tuo modello con`StartModel`operazione. Per ulteriori informazioni, consulta Avvio del modello Amazon Lookout for Vision. Ti viene addebitato il costo in base alla durata, in minuti, di funzionamento di un modello e al numero di unità di rilevamento delle anomalie utilizzate dal modello. Se non si utilizza un modello, utilizzare il`StopModel`operazione per fermare il modello. Per ulteriori informazioni, consulta Interruzione del modello Amazon Lookout for Vision.

Argomenti

- Chiamata di DetectAnomalies
- Comprendere la risposta diDetectAnomalies
- Determinare se un'immagine è anomala
- Visualizzazione delle informazioni su classificazione e segmentazione
- Individuazione di anomalie con unAWS Lambdafunzione

Chiamata di DetectAnomalies

Chiamare`DetectAnomalies`, specificare quanto segue:

- Progetto— Il nome del progetto che contiene il modello che si desidera utilizzare.
- ModelVersion— La versione del modello che si desidera utilizzare.

- ContentType— Il tipo di immagine che desideri analizzare. I valori validi sonoimage/png(immagini in formato PNG) eimage/jpeg(immagini in formato JPG).
- Corpo— I byte binari non codificati che rappresentano l'immagine.

  L'immagine deve avere le stesse dimensioni delle immagini utilizzate per addestrare il modello.

L'esempio seguente mostra come chiamareDetectAnomalies. È possibile utilizzare la risposta della funzione degli esempi di Python e Java per chiamare le funzioni inDeterminare se un'immagine è anomala.

AWS CLI

Questo comando AWS CLI visualizza l'output JSON dell'operazione CLI DetectAnomalies. Modificate i valori dei seguenti parametri di input:

- project namecon il nome del progetto che desideri utilizzare.
- model versioncon la versione del modello che si desidera utilizzare.
- content typecon il tipo di immagine che vuoi usare. I valori validi sonoimage/png(immagini in formato PNG) eimage/jpeg(immagini in formato JPG).
- file namecon il percorso e il nome del file dell'immagine che si desidera utilizzare. Assicurati che il tipo di file corrisponda al valore dicontent-type.

```
aws lookoutvision detect-anomalies  --project-name project name\
    --model-version model version\
    --content-type content type\
    --body file name \
    --profile lookoutvision-access
```

Python

Per l'esempio di codice completo, vedereGitHub.

```
    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));
    }
```

Comprendere la risposta diDetectAnomalies

La risposta di`DetectAnomalies`varia a seconda del tipo di modello da addestrare (modello di classificazione o modello di segmentazione). In entrambi i casi la risposta è[DetectAnomalyResult](#)oggetto.

Modello di classificazione

Se il tuo modello è[Modello di classificazione delle immagini](#), la risposta di`DetectAnomalies`contiene quanto segue:

- IsAnomalous— Un indicatore booleano che indica che l'immagine contiene una o più anomalie.
- Fiducia— La fiducia che Amazon Lookout for Vision ripone nell'accuratezza della previsione delle anomalie (`IsAnomalous`).`Confidence`è un valore in virgola mobile compreso tra 0 e 1. Un valore più alto indica una maggiore confidenza.
- Fonte— Informazioni sull'immagine trasmessa a`DetectAnomalies`.

```
{
    "DetectAnomalyResult": {
        "Source": {
            "Type": "direct"
        },
        "IsAnomalous": true,
        "Confidence": 0.9996867775917053
    }
}
```

Si determina se in un'immagine è anomala controllando il`IsAnomalous`campo e confermando che il`Confidence`il valore è sufficientemente alto per le tue esigenze.

Se stai trovando i valori di confidenza restituiti da`DetectAnomalies`sono troppo bassi, considera la riqualificazione del modello. Per il codice di esempio, consulta Classificazione.

Modello di segmentazione

Se il tuo modello èModello di segmentazione delle immagini, la risposta include informazioni sulla classificazione e sulla segmentazione, come una maschera di immagine e tipi di anomalie. Le informazioni sulla classificazione vengono calcolate separatamente dalle informazioni sulla segmentazione e non si deve presumere una relazione tra di esse. Se non ricevi informazioni sulla segmentazione nella risposta, verifica di disporre della versione più recente diAWSSDK installato (AWS Command Line Interface, se si utilizzaAWS CLI). Ad esempio di codice, vediSegmentazioneeVisualizzazione delle informazioni su classificazione e segmentazione.

- IsAnomalous(classificazione) — Un indicatore booleano che classifica l'immagine come normale o anomala.
- Fiducia(classificazione) — La fiducia di Amazon Lookout for Vision nell'accuratezza della classificazione dell'immagine (`IsAnomalous`).`Confidence`è un valore in virgola mobile compreso tra 0 e 1. Un valore più alto indica una maggiore confidenza.
- Fonte— Informazioni sull'immagine trasmessa a`DetectAnomalies`.
- AnomalyMask(segmentazione) — Una maschera di pixel che copre le anomalie rilevate nell'immagine analizzata. L'immagine può presentare più anomalie. Il colore di una mappa a maschera indica il tipo di anomalia. I colori della maschera corrispondono ai colori assegnati ai tipi di anomalia nel set di dati di addestramento. Per trovare il tipo di anomalia in base al colore di una maschera, controllaColornelPixelAnomalycampo di ciascuna anomalia restituita

nel`Anomalies`elenco. Per il codice di esempio, consulta Visualizzazione delle informazioni su classificazione e segmentazione.

- Anomalie(segmentazione) — Un elenco di anomalie rilevate nell'immagine. Ogni anomalia include il tipo di anomalia (`Name`) e informazioni sui pixel (`PixelAnomaly`).`TotalPercentageArea`è l'area percentuale dell'immagine coperta dall'anomalia.`Color`è il colore della maschera per l'anomalia.

  Il primo elemento dell'elenco è sempre un tipo di anomalia che rappresenta lo sfondo dell'immagine (`BACKGROUND`) e non deve essere considerata un'anomalia. Amazon Lookout for Vision aggiunge automaticamente il tipo di anomalia di sfondo alla risposta. Non è necessario dichiarare un tipo di anomalia di fondo nel set di dati.

```
{
    "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....."
    }
}
```

Determinare se un'immagine è anomala

È possibile determinare se un'immagine è anomala in diversi modi. Il metodo scelto dipende dal tuo caso d'uso e dal tipo di modello. Di seguito sono elencate le possibili soluzioni.

Argomenti

- Classificazione
- Segmentazione

Classificazione

`IsAnomalous`classifica un'immagine come anomala, usa il`Confidence`campo per aiutare a decidere se l'immagine è effettivamente anomala. Un valore più alto indica una maggiore sicurezza. Ad esempio, potresti decidere che un prodotto è difettoso solo se la fiducia è superiore all'80%. È possibile classificare le immagini analizzate mediante modelli di classificazione o modelli di segmentazione delle immagini.

Python

Per l'esempio di codice completo, vedereGitHub.

```
   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;

    }
```

Segmentazione

Se il modello è un modello di segmentazione dell'immagine, puoi utilizzare le informazioni sulla segmentazione per determinare se un'immagine contiene anomalie. È inoltre possibile utilizzare un modello di segmentazione delle immagini per classificare le immagini. Ad esempio, codice che ottiene e visualizza maschere di immagini, vediVisualizzazione delle informazioni su classificazione e segmentazione

Area di anomalia

Usa la copertura percentuale (`TotalPercentageArea`) di un'anomalia sull'immagine. Ad esempio, potresti decidere che un prodotto è difettoso se l'area di un'anomalia è superiore all'1% dell'immagine.

Python

Per l'esempio di codice completo, vedereGitHub.

```
    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;
    }
```

Numero di tipi di anomalie

Utilizza un conteggio di diversi tipi di anomalie (Name) trovato nell'immagine. Ad esempio, potresti decidere che un prodotto è difettoso se sono presenti più di due tipi di anomalie.

Python

Per l'esempio di codice completo, vedereGitHub.

```
   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;
    }
```

Visualizzazione delle informazioni su classificazione e segmentazione

Questo esempio mostra l'immagine analizzata e sovrappone i risultati dell'analisi. Se la risposta include una maschera di anomalia, la maschera viene mostrata con i colori dei tipi di anomalia associati.

Per mostrare le informazioni sulla classificazione e la segmentazione delle immagini

1. Se non l'hai già fatto, procedi come segue:

   a. Se non l'hai già fatto, installa e configuraAWS CLIe ilAWSSDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.

   b. Addestra il tuo modello.

   c. Inizia il tuo modello.

2. Assicurati che l'utente stia chiamando`DetectAnomalies`ha accesso alla versione del modello che desideri utilizzare. Per ulteriori informazioni, consulta Impostare le autorizzazioni dell'SDK.

3. Usa il codice seguente.

Python

Il codice di esempio seguente rileva le anomalie in un'immagine fornita. L'esempio utilizza le seguenti opzioni della riga di comando:

- `project`— il nome del progetto che desideri utilizzare.
- `version`— la versione del modello, all'interno del progetto, che si desidera utilizzare.
- `image`— il percorso e il file di un file di immagine locale (formato JPEG o 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

Il codice di esempio seguente rileva le anomalie in un'immagine fornita. L'esempio utilizza le seguenti opzioni della riga di comando:

- `project`— il nome del progetto che desideri utilizzare.
- `version`— la versione del modello, all'interno del progetto, che si desidera utilizzare.
- `image`— il percorso e il file di un file di immagine locale (formato JPEG o 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. Se non hai intenzione di continuare a utilizzare il tuo modello,ferma il tuo modello.

Individuazione di anomalie con unAWS Lambdafunzione

AWS Lambda è un servizio di elaborazione che consente di eseguire del codice senza la necessità di effettuare il provisioning o la gestione dei server. Ad esempio, è possibile analizzare le immagini inviate da un'applicazione mobile senza dover creare un server per ospitare il codice dell'applicazione. Le seguenti istruzioni mostrano come creare una funzione Lambda in Python che chiamaDetectAnomalies. La funzione analizza un'immagine fornita e restituisce una classificazione per la presenza di anomalie in quell'immagine. Le istruzioni includono un esempio di codice Python che mostra come chiamare la funzione Lambda con un'immagine in un bucket Amazon S3 o un'immagine fornita da un computer locale.

Argomenti

- Fase 1: Creare unAWS Lambdafunzione (console)
- Fase 2: (Facoltativo) Crea un livello (console)
- Passaggio 3: aggiungi codice Python (console)
- Fase 4: Prova la funzione Lambda

Fase 1: Creare unAWS Lambdafunzione (console)

In questo passaggio, crei un vuotoAWSfunzione e un ruolo di esecuzione IAM che consente alla funzione di chiamare il`DetectAnomalies`operazione. Consente inoltre l'accesso al bucket Amazon S3 che archivia le immagini per l'analisi. È inoltre possibile specificare le variabili di ambiente per quanto segue:

- Il progetto Amazon Lookout for Vision e la versione del modello da utilizzare con la funzione Lambda.
- Il limite di confidenza che si desidera che il modello utilizzi.

Successivamente aggiungi il codice sorgente e facoltativamente un livello alla funzione Lambda.

Per creare unAWS Lambdafunzione (console)

1. Accedere alla AWS Management Console e aprire la console di AWS Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
2. Scegli Create function (Crea funzione). Per ulteriori informazioni, vedereCrea una funzione Lambda con la console.
3. Scegliete le seguenti opzioni.
   - Scegli Author from scratch (Crea da zero).
   - Inserisci un valore perNome della funzione.
   - PerRuntimesceglierePython 3.10.
4. ScegliCrea funzioneper creare ilAWS Lambdafunzione.
5. Nella pagina delle funzioni, scegli ilConfigurazionescheda.
6. SulVariabili d'ambienteriquadro, scegliModifica.
7. Aggiungere le seguenti variabili di ambiente. Per ogni variabile scegliAggiungi variabile di ambientequindi inserisci la chiave e il valore della variabile.

Key (Chiave)	Value (Valore)
NOME_PROGETTO	Il progetto Lookout for Vision che contiene il modello che desideri utilizzare.
VERSIONE_MODELLO	La versione del modello che si desidera utilizzare.
FIDUCIA	Il valore minimo (0-100) per la certezza del modello che la previsione è anomala. Se la confidenza è inferiore, la classificazione è considerata normale.

8. ScegliSalvaper salvare le variabili di ambiente.
9. SulAutorizzazionipannello, sottoNome del ruolo, scegli il ruolo di esecuzione per aprire il ruolo nella console IAM.
10. NelAutorizzazionischeda, scegliAggiungi autorizzazionie poiCrea una politica in linea.
11. ScegliJSONe sostituisci la politica esistente con la seguente politica.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "lookoutvision:DetectAnomalies",
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "DetectAnomaliesAccess"
        }
    ]
}
```

12. Seleziona Successivo.
13. NelDettagli della politica, inserisci un nome per la politica, ad esempioDetectAnomalies-accesso.
14. Scegli Create Policy (Crea policy).
15. Se stai archiviando immagini per l'analisi in un bucket Amazon S3, ripeti i passaggi da 10 a 14.

    a. Per il passaggio 11, utilizza la seguente politica. Sostituisci*percorso del bucket/cartella*con il percorso del bucket e della cartella Amazon S3 verso le immagini che desideri analizzare.

    ```
    {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "S3Access",
                "Effect": "Allow",
                "Action": "s3:GetObject",
                "Resource": "arn:aws:s3:::bucket/folder path/*"
            }
        ]
    }
    ```

    b. Per il passaggio 13, scegli un nome diverso per la politica, ad esempioAccesso al bucket S3.

Fase 2: (Facoltativo) Crea un livello (console)

Per eseguire questo esempio, non è necessario eseguire questo passaggio. La`DetectAnomalies`l'operazione è inclusa nell'ambiente Lambda Python predefinito come parte

diAWSSDK per Python (Boto3). Se altre parti della tua funzione Lambda richiedono una versione recenteAWSper gli aggiornamenti dei servizi che non si trovano nell'ambiente predefinito di Lambda Python, esegui questo passaggio per aggiungere l'ultima versione di Boto3 SDK come livello alla tua funzione.

Innanzitutto, crei un archivio di file.zip che contiene l'SDK Boto3. Si crea quindi un layer e si aggiunge l'archivio di file.zip al layer. Per ulteriori informazioni, vedereUtilizzo dei livelli con la funzione Lambda.

Per creare e aggiungere un livello (console)

1. Apri un prompt dei comandi e inserisci i seguenti comandi.

```
pip install boto3 --target python/.
zip boto3-layer.zip -r python/
```

2. Nota il nome del file zip (boto3-layer.zip). Ne hai bisogno nella fase 6 di questa procedura.
3. Apri la console AWS Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
4. Nel riquadro di navigazione scegli Layers (Livelli).
5. Scegli Create layer (Crea livello).
6. Immetti i valori per Nome (Nome) e Description (Descrizione).
7. ScegliCarica un file.zipe scegliCarica.
8. Nella finestra di dialogo, scegliete l'archivio con estensione zip (boto3-layer.zip) creato nel passaggio 1 di questa procedura.
9. Per tempi di esecuzione compatibili, scegliPython 3.9.
10. ScegliCreaper creare il livello.
11. Scegli l'icona del menu del pannello di navigazione.
12. Nel riquadro di navigazione, seleziona Funzioni.
13. Nell'elenco delle risorse, scegli la funzione che hai creato inFase 1: Creare unAWS Lambdafunzione (console).
14. Scegli la scheda Codice.
15. NelStratisezione, scegliAggiungi un livello.
16. ScegliLivelli personalizzati.
17. NelLivelli personalizzati, scegli il nome del livello che hai inserito nel passaggio 6.

18. NelVersionescegli la versione del livello, che dovrebbe essere 1.
19. Scegli Add (Aggiungi).

Passaggio 3: aggiungi codice Python (console)

In questo passaggio, aggiungi codice Python alla tua funzione Lambda utilizzando l'editor di codice della console Lambda. Il codice analizza un'immagine fornita con`DetectAnomaliese` restituisce una classificazione (true se l'immagine è anomala, false se l'immagine è normale). L'immagine fornita può essere posizionata in un bucket Amazon S3 o fornita come byte di immagine codificati in byte64.

Per aggiungere codice Python (console)

1. Se non utilizzi la console Lambda, procedi come segue:
   a. Apri la console AWS Lambda all'indirizzo https://console.aws.amazon.com/lambda/.
   b. Apri la funzione Lambda che hai creato inFase 1: Creare unAWS Lambdafunzione (console).
2. Scegli la scheda Codice.
3. NelCodice sorgente, sostituisci il codice inlambda_function.pycon quanto segue:

```
# 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. ScegliDistribuisciper implementare la tua funzione Lambda.

Fase 4: Prova la funzione Lambda

In questo passaggio, usi il codice Python sul tuo computer per passare un'immagine locale o un'immagine in un bucket Amazon S3 alla tua funzione Lambda. Le immagini trasmesse da un computer locale devono avere una dimensione inferiore a 6291456 byte. Se le tue immagini sono più grandi, carica le immagini in un bucket Amazon S3 e richiama lo script con il percorso Amazon S3 dell'immagine. Per informazioni sul caricamento di file di immagini in un bucket Amazon S3, consultaCaricamento di oggetti.

Assicurati di eseguire il codice nello stessoAWSRegione in cui è stata creata la funzione Lambda. È possibile visualizzareAWSRegione per la funzione Lambda nella barra di navigazione della pagina dei dettagli della funzione inConsolle Lambda.

Se ilAWS Lambdala funzione restituisce un errore di timeout, prolunga il periodo di timeout per la funzione Lambda. Per ulteriori informazioni, vedereConfigurazione del timeout della funzione (console).

Per ulteriori informazioni su come richiamare una funzione Lambda dal tuo codice, vediInvocandoAWS LambdaFunzioni.

Per provare la funzione Lambda

1. Se non l'hai già fatto, procedi come segue:

    a. Assicurati che l'utente che utilizza il codice client abbia`lambda:InvokeFunction`permesso. Puoi usare le seguenti autorizzazioni.

    ```
    {
        "Version": "2012-10-17",
        "Statement": [
    ```

```
        {
            "Sid": "LambdaPermission",
            "Effect": "Allow",
            "Action": "lambda:InvokeFunction",
            "Resource": "ARN for lambda function"
        }
    ]
}
```

È possibile ottenere l'ARN per la funzione della funzione Lambda dalla panoramica delle funzioni nelConsolle Lambda.

Per fornire l'accesso, aggiungi autorizzazioni ai tuoi utenti, gruppi o ruoli:

- Utenti e gruppi in AWS IAM Identity Center:

  Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina Create a permission set (Creazione di un set di autorizzazioni) nella Guida per l'utente di AWS IAM Identity Center.

- Utenti gestiti in IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate nella pagina Creating a role for a third-party identity provider (federation) (Creazione di un ruolo per un provider di identità di terze parti [federazione]) nella Guida per l'utente di IAM.

- Utenti IAM:
  - Crea un ruolo che l'utente possa assumere. Per istruzioni, consulta la pagina Creating a role for an IAM user (Creazione di un ruolo per un utente IAM) nella Guida per l'utente di IAM.
  - (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate nella pagina Aggiunta di autorizzazioni a un utente (console) nella Guida per l'utente di IAM.

b. Installa e configuraAWSSDK per Python. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.

c. Avvia il modelloche hai specificato nel passaggio 7 diFase 1: Creare unAWS Lambdafunzione (console).

2. Salva il seguente codice in un file denominato`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. Eseguire il codice. Come argomento della riga di comando, fornisci il nome della funzione Lambda e il percorso di un'immagine locale che desideri analizzare. Ad esempio:

```
python client.py function_name /bucket/path/image.jpg
```

   In caso di successo, l'output è una classificazione per le anomalie rilevate nell'immagine. Se non viene restituita una classificazione, valuta la possibilità di ridurre il valore di confidenza impostato nel passaggio 7 diFase 1: Creare unAWS Lambdafunzione (console).

4. Se hai finito con la funzione Lambda e il modello non è utilizzato da altre applicazioni,ferma il modello. Ricordati diavvia il modellola prossima volta che vuoi usare la funzione Lambda.

Utilizzo del modello Amazon Lookout for Vision su un dispositivo edge

Puoi usare il tuo modello Amazon Lookout for Vision su dispositivi edge gestiti AWS IoT Greengrass Version 2 da. AWS IoT Greengrass è un runtime edge e un servizio cloud open source per l'Internet of Things (IoT). Puoi usarlo per creare, implementare e gestire applicazioni IoT sui tuoi dispositivi. Per ulteriori informazioni, consulta AWS IoT Greengrass.

Implementa gli stessi modelli Amazon Lookout for Vision che hai addestrato nel cloud AWS IoT Greengrass V2 su dispositivi edge compatibili. Puoi quindi utilizzare il modello distribuito per eseguire il rilevamento delle anomalie in locale, ad esempio in fabbrica, senza lo streaming continuo dei dati sul cloud. In questo modo puoi ridurre al minimo i costi della larghezza di banda e rilevare le anomalie localmente con l'analisi delle immagini in tempo reale.

> **Tip**
>
> Prima di implementare un modello Lookout for Vision AWS IoT Greengrass con, ti consigliamo di leggere AWS IoT Greengrass Version 2la guida per gli sviluppatori. Per ulteriori informazioni, consulta Cos'è AWS IoT Greengrass? .

Per utilizzare un modello Lookout for Vision su AWS IoT Greengrass V2 un dispositivo principale, devi distribuire il modello e il software di supporto come componenti del dispositivo principale. Un componente è un modulo software, ad esempio un modello Lookout for Vision, che viene eseguito su un dispositivo core Greengrass. Esistono due tipi di componenti. Un componente personalizzato è un componente creato dall'utente ed è accessibile solo all'utente. È anche noto come componente privato. Un componente AWS fornito è un componente precostruito che AWS fornisce. È anche noto come componente pubblico. Per ulteriori informazioni, consulta https://docs.aws.amazon.com/greengrass/v2/developerguide/public-components.html.

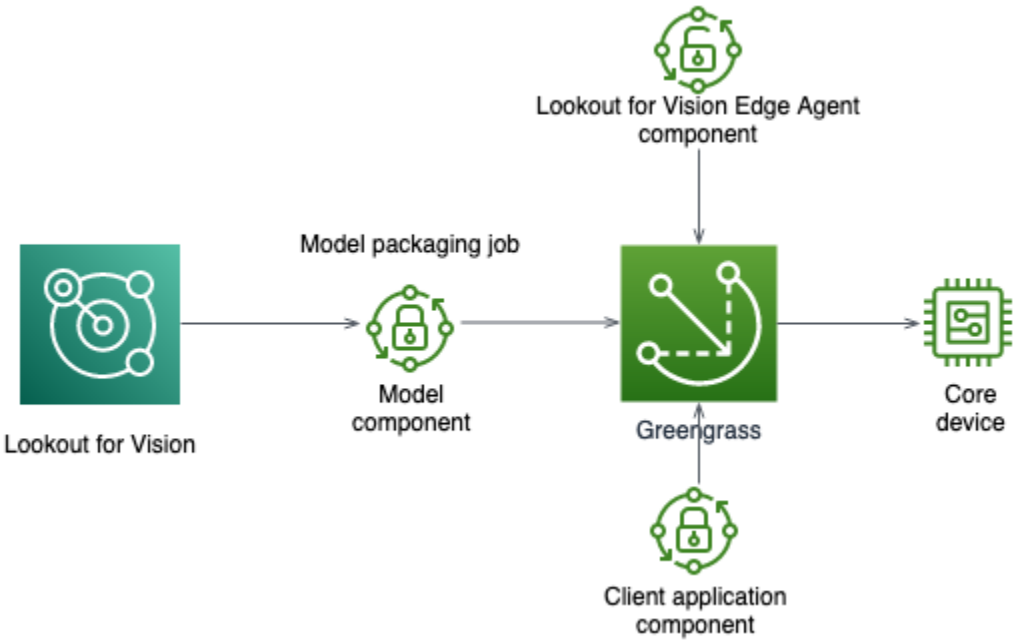
I componenti che si distribuiscono su un dispositivo principale per un modello Lookout for Vision e il software di supporto sono:

- Componente del modello. Un componente personalizzato che contiene il modello Lookout for Vision. Per creare il componente del modello, si utilizza Lookout for Vision per creare un processo di packaging del modello. Un processo di modellazione crea un componente per il modello e lo

rende disponibile come componente personalizzato all'internoAWS IoT Greengrass V2. Per ulteriori informazioni, consulta Imballaggio del modello Amazon Lookout for Vision.

- Componente dell'applicazione client. Un componente personalizzato creato dall'utente che implementa il codice per i requisiti aziendali. Ad esempio, trovare circuiti stampati anomali a partire da immagini scattate dopo l'assemblaggio. Per ulteriori informazioni, consulta Scrivere il componente dell'applicazione client.
- Componente Amazon Lookout for Vision Edge Agent. Un componente AWS fornito che fornisce un'API per l'utilizzo e la gestione del modello. Ad esempio, il codice nel componente dell'applicazione client può utilizzare l'`DetectAnomalies`API per rilevare anomalie nelle immagini. Il componente Lookout for Vision Edge Agent è una dipendenza del componente del modello. Viene installato automaticamente sul dispositivo principale quando si distribuisce il componente del modello. Per ulteriori informazioni, consulta Riferimento all'API Amazon Lookout for Vision Edge Agent.

Dopo aver creato il componente del modello e il componente dell'applicazione client, è possibile utilizzarli AWS IoT Greengrass V2 per distribuire i componenti e le dipendenze sul dispositivo principale. Per ulteriori informazioni, consulta Distribuzione dei componenti su un dispositivo.



Important

Le previsioni effettuate dal modello `DetectAnomalies` su un dispositivo principale potrebbero differire dalle previsioni effettuate utilizzando lo stesso modello ospitato nel cloud.

Ti consigliamo di testare il modello su un dispositivo principale prima di utilizzarlo in un ambiente di produzione.
Per ridurre le discrepanze di previsione tra modelli ospitati su dispositivi e modelli ospitati sul cloud, consigliamo di aumentare il numero di immagini normali e anomale nel set di dati di addestramento. Non è consigliabile riutilizzare le immagini esistenti per aumentare le dimensioni del set di dati di addestramento.

Implementazione di un modello e di un componente dell'applicazione client su un dispositivo principale AWS IoT Greengrass Version 2

La procedura per distribuire un modello Amazon Lookout for Vision e un componente dell'applicazione client su AWS IoT Greengrass Version 2 un dispositivo principale è la seguente:

1. Configura i tuoi dispositivi principali con. AWS IoT Greengrass Version 2
2. Crea un processo di creazione di modelli utilizzando Lookout for Vision. Il lavoro crea il componente del modello.
3. Scrivi un componente dell'applicazione client. Il componente implementa la tua logica aziendale.
4. Distribuisci il componente del modello e il componente dell'applicazione client sul dispositivo principale utilizzando. AWS IoT Greengrass V2

Dopo aver distribuito i componenti e le dipendenze sul dispositivo principale, è possibile utilizzare il modello sul dispositivo principale.

> **Note**
>
> È necessario utilizzare la stessa AWS regione e lo stesso AWS account per creare e distribuire il modello Lookout for Vision e il componente dell'applicazione client.

AWS IoT Greengrass Version 2requisiti di base del dispositivo

Per utilizzare un modello Amazon Lookout for Vision su AWS IoT Greengrass Version 2 un dispositivo principale, il tuo modello ha diversi requisiti del dispositivo principale.

Argomenti

Dispositivi, architetture di chip e sistemi operativi testati

Prevediamo che Amazon Lookout for Vision funzioni sul seguente hardware:

- Architetture CPU
  - X86_64 (versione a 64 bit del set di istruzioni x86)
  - Aarch64 (CPU ARMv8 a 64 bit)
- (Solo inferenza accelerata da GPU) NVIDIA GPU Accelerator con capacità di memoria sufficiente (almeno 6,0 GB per un modello funzionante).

Il team di Amazon Lookout for Vision ha testato i modelli Lookout for Vision sui seguenti dispositivi, architetture di chip e sistemi operativi.

Dispositivi

Dispositivo	Sistema operativo	Architettura	Accelerator	Opzioni del compilatore

jetson_xavier (NVIDIA® Jetson AGX Xavier)	Linux	Aarch 64	NVIDIA	`{"gpu-code": "sm_72", "trt-ver" : "7.1.3", "cuda-ver": "10.2"}`
				
				`{"gpu-code": "sm_72", "trt-ver"`

Dispositivo	Sistema operativo	Architettura	Accelerator	Opzioni del compilatore
				`: "8.2.1", "cuda-ver": "10.2"}`
g4dn.xlarge (istanze EC2 (G4) con GPU NVIDIA T4 Tensor Core)	Linux	X86_64/X86-64	NVIDIA	`{"gpu-code": "sm_75", "trt-ver" : "7.1.3", "cuda-ver": "10.2"}`
g5.xlarge (istanze EC2 (G5) con GPU NVIDIA A10G Tensor Core)	Linux	X86_64/X86-64	NVIDIA	`{"gpu-code": "sm_80", "trt-ver" : "8.2.0", "cuda-ver": "11.2"}`
c5.2xlarge (istanze Amazon EC2 C5)	Linux	X86_64/X86-64	CPU	`{"mcpu": "core-avx 2"}`

Memoria e archiviazione principali del dispositivo

Per eseguire un singolo modello e Amazon Lookout for Vision Edge Agent, il dispositivo principale ha i seguenti requisiti di memoria e storage. Potresti aver bisogno di più memoria e spazio di archiviazione per il componente dell'applicazione client.

- Archiviazione: almeno 1,5 GB.
- Memoria: almeno 6,0 GB per un modello funzionante.

Software richiesto

Un dispositivo principale richiede il seguente software.

Dispositivi Jetson

Se il dispositivo principale è un dispositivo Jetson, è necessario che sul dispositivo principale sia installato il seguente software.

Software	Versioni supportate
Jetpack SDK	da 4.4 a 4.6.1
Ambiente virtuale Python e Python per Lookout for Vision Edge Agent versione 1.x	3.8 o 3.9

Hardware X86

Se il dispositivo principale utilizza hardware x86, è necessario installare il seguente software sul dispositivo principale.

Inferenza della CPU

Software	Versioni supportate
Ambiente virtuale Python e Python per Lookout for Vision Edge Agent versione 1.x	3.8 o 3.9

Inferenza accelerata dalla GPU

Le versioni del software variano a seconda della microarchitettura della GPU NVIDIA utilizzata.

GPU NVIDIA con microarchitettura precedente ad Ampere (la capacità di elaborazione è inferiore a 8.0)

Software richiesto per una GPU NVIDIA con una microarchitettura precedente ad Ampere (capacità di elaborazione inferiore a 8,0). Deve essere inferiore a. `gpu-code sm_80`

Software	Versioni supportate
NVIDIA CUDA	10.2
NVIDIA TensorRT	Almeno 7.1.3 e meno di 8.0.0
Ambiente virtuale Python e Python per Lookout for Vision Edge Agent versione 1.x	3.8 o 3.9

GPU NVIDIA con microarchitettura Ampere (capacità di calcolo 8.0)

Software richiesto per una GPU NVIDIA con microarchitettura Ampere (la capacità di elaborazione è 8.0). Deve essere). `gpu-code sm_80`

Software	Versioni supportate
NVIDIA CUDA	11.2
NVIDIA TensorRT	8.2.0
Ambiente virtuale Python e Python per Lookout for Vision Edge Agent versione 1.x	3.8 o 3.9

Configurazione del dispositivo principale AWS IoT Greengrass Version 2

Amazon Lookout for Vision AWS IoT Greengrass Version 2 utilizza per semplificare la distribuzione del componente modello, del componente Amazon Lookout for Vision Edge Agent e del componente dell'applicazione client sul dispositivo principale. AWS IoT Greengrass V2 Per informazioni sui dispositivi e sull'hardware che puoi utilizzare, consulta. AWS IoT Greengrass Version 2requisiti di base del dispositivo

Configurazione del dispositivo principale

Usa le seguenti informazioni per configurare il tuo dispositivo principale.

Per configurare il dispositivo principale

1. Configura le tue librerie GPU. Non eseguire questo passaggio se non utilizzi l'inferenza accelerata da GPU.

   a. Verifica di disporre di una GPU che supporti CUDA. Per ulteriori informazioni, consulta Verifica di disporre di una GPU compatibile con CUDA.

   b. Configura CUDA, cuDNN e TensorRT sul tuo dispositivo effettuando una delle seguenti operazioni:

   - Se utilizzi un dispositivo Jetson, installa la versione 4.4 - 4.6.1. JetPack Per ulteriori informazioni, consulta Archivio. JetPack
   - Se utilizzi hardware basato su x86 e la microarchitettura della GPU NVIDIA è precedente ad Ampere (la capacità di elaborazione è inferiore a 8,0), procedi come segue:
     1. Configura CUDA versione 10.2 seguendo le istruzioni riportate nella Guida all'installazione di NVIDIA CUDA per Linux.
     2. Installa cuDNN seguendo le istruzioni nella documentazione di NVIDIA cuDNN.
     3. Configura TensorRT (versione 7.1.3 o successiva, ma precedente alla 8.0.0) seguendo le istruzioni nella DOCUMENTAZIONE DI NVIDIA TENSORRT.
   - Se utilizzi hardware basato su x86 e la microarchitettura della GPU NVIDIA è Ampere (la capacità di elaborazione è 8.0), procedi come segue:
     1. Configura CUDA (versione 11.2) seguendo le istruzioni riportate nella Guida all'installazione di NVIDIA CUDA per Linux.
     2. Installa cuDNN seguendo le istruzioni nella documentazione di NVIDIA cuDNN.
     3. Configura TensorRT (versione 8.2.0) seguendo le istruzioni nella DOCUMENTAZIONE NVIDIA TENSORRT.

2. Installa il software di base sul tuo dispositivo principale. AWS IoT Greengrass Version 2 Per ulteriori informazioni, consulta Installare il software AWS IoT Greengrass Core nella AWS IoT Greengrass Version 2Developer Guide.

3. Per leggere dal bucket Amazon S3 che memorizza il modello, assegna l'autorizzazione al ruolo IAM (ruolo di scambio di token) che crei durante la configurazione. AWS IoT Greengrass Version 2 Per ulteriori informazioni, consulta Consentire l'accesso ai bucket S3 per gli artefatti dei componenti.

4. Al prompt dei comandi, inserisci il seguente comando per installare Python e un ambiente virtuale Python sul dispositivo principale.

```
sudo apt install python3.8 python3-venv python3.8-venv
```

5. Utilizzate il seguente comando per aggiungere l'utente Greengrass al gruppo video. Ciò consente ai componenti distribuiti da Greengrass di accedere alla GPU:

```
sudo usermod -a -G video ggc_user
```

6. (Facoltativo) Se desideri chiamare l'API Lookout for Vision Edge Agent da un altro utente, aggiungi l'utente richiesto a`ggc_group`. Ciò consente all'utente di comunicare con l'agente Lookout for Vision Edge tramite il socket di dominio Unix:

```
sudo usermod -a -G ggc_group $(whoami)
```

Imballaggio del modello Amazon Lookout for Vision

Un processo di packaging di modelli include un modello Amazon Lookout for Vision come componente del modello.

Per creare un processo di confezionamento del modello, scegli il modello che desideri impacchettare e fornisci le impostazioni per il componente del modello creato dal lavoro. È possibile confezionare solo un modello che è stato addestrato con successo.

Puoi utilizzare la console Lookout for Vision AWS o l'SDK per creare il processo di packaging del modello. Puoi anche ottenere informazioni sui lavori di confezionamento dei modelli che crei. Per ulteriori informazioni, consulta Ottenere informazioni sui lavori di creazione di modelli. Puoi utilizzare la AWS IoT Greengrass V2 console o l'AWSSDK per distribuire i componenti sul dispositivo AWS IoT Greengrass Version 2 principale.

Argomenti

- Impostazioni del pacchetto
- Impacchettizzazione del modello (Console)
- Imballaggio del modello (SDK)
- Ottenere informazioni sui lavori di creazione di modelli

Impostazioni del pacchetto

Utilizzate le seguenti informazioni per decidere le impostazioni del pacchetto per il vostro lavoro di confezionamento del modello.

Per creare un processo di confezionamento del modello, consulta Impacchettizzazione del modello (Console) oImballaggio del modello (SDK).

Argomenti

- Hardware di destinazione
- Impostazioni dei componenti

Hardware di destinazione

Puoi scegliere un dispositivo o una piattaforma di destinazione per il tuo modello, ma non entrambi. Per ulteriori informazioni, consulta Dispositivi, architetture di chip e sistemi operativi testati.

Dispositivo di destinazione

Il dispositivo di destinazione per il modello, ad esempio NVIDIA® Jetson AGX Xavier. Non è necessario specificare le opzioni del compilatore.

Piattaforma di destinazione

Amazon Lookout for Vision supporta le seguenti configurazioni di piattaforma:

- Architetture X86_64 (versione a 64 bit del set di istruzioni x86) e Aarch64 (CPU ARMv8 a 64 bit).
- Sistema operativo Linux.
- Inferenza tramite acceleratori NVIDIA o CPU.

È necessario specificare le opzioni di compilazione corrette per la piattaforma di destinazione.

Opzioni del compilatore

Le opzioni del compilatore consentono di specificare la piattaforma di destinazione per il dispositivo AWS IoT Greengrass Version 2 principale. Attualmente è possibile specificare le seguenti opzioni del compilatore.

Acceleratore NVIDIA

- `gpu-code`— Specifica il codice gpu del dispositivo principale su cui è in esecuzione il componente del modello.
- `trt-ver`— specifica la versione di TensorRT in formato x.y.z..
- `cuda-ver`— Speciifica la versione CUDA in formato x.y.

Acceleratore CPU

- (Facoltativo)`mcpu`: specifica il set di istruzioni. Ad esempio, `core-avx2`. Se non fornisci un valore, Lookout for Vision utilizza il `core-avx2` valore.

Le opzioni vengono specificate in formato JSON. Ad esempio:

```
{"gpu-code": "sm_75", "trt-ver": "7.1.3", "cuda-ver": "10.2"}
```

Per ulteriori esempi, consulta Dispositivi, architetture di chip e sistemi operativi testati.

Impostazioni dei componenti

Il processo di creazione del modello crea un componente del modello che contiene il modello. Il job crea artefatti che AWS IoT Greengrass V2 vengono utilizzati per distribuire il componente del modello sul dispositivo principale.

Non è possibile creare un componente del modello con lo stesso nome e la stessa versione del componente di un componente esistente.

Nome componente

Un nome per il componente del modello creato da Lookout for Vision durante il packaging del modello. Il nome del componente specificato viene visualizzato nella AWS IoT Greengrass V2 console. Il nome del componente viene utilizzato nella ricetta creata per il componente dell'applicazione client. Per ulteriori informazioni, consulta Creazione del componente dell'applicazione client.

Descrizione del componente

(Facoltativo) Una descrizione del componente del modello.

Versione del componente

Un numero di versione per il componente del modello. È possibile accettare il numero di versione predefinito o sceglierne uno personalizzato. Il numero di versione deve seguire il sistema di numerazione semantico della versione: major.minor.patch. Ad esempio, la versione 1.0.0 rappresenta la prima release principale di un componente. Per ulteriori informazioni, consulta Controllo delle versioni semantico 2.0.0. Se non fornisci un valore, Lookout for Vision utilizza il numero di versione del tuo modello per generare una versione per te.

Ubicazione del componente

La sede Amazon S3 in cui desideri che il processo di confezionamento del modello salvi gli artefatti dei componenti del modello. Il bucket Amazon S3 deve trovarsi nella stessa regione AWS e nello stesso account AWS in cui lo utilizzi. AWS IoT Greengrass Version 2 Per creare un bucket Amazon S3, consulta Creazione di un bucket.

Tag

Puoi identificare, organizzare, cercare e filtrare i componenti utilizzando i tag. Ogni tag è un'etichetta composta da una chiave e da un valore definiti dall'utente. I tag vengono attaccati al componente del modello quando il processo di imballaggio del modello crea il componente del modello in Greengrass. Un componente è una risorsa AWS IoT Greengrass V2. I tag non sono allegati a nessuna delle tue risorse Lookout for Vision, come i tuoi modelli. Per ulteriori informazioni, consulta Tagging AWS resources.

Impacchettizzazione del modello (Console)

Puoi creare un processo di packaging del modello utilizzando la console Amazon Lookout for Vision.

Per informazioni sulle impostazioni dei pacchetti, consultaImpostazioni del pacchetto.

Per impacchettare un modello (console)

1. Crea un bucket Amazon S3 o riutilizza un bucket esistente che Lookout for Vision utilizza per archiviare gli elementi del processo di imballaggio (componente del modello).
2. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
3. Scegliere Inizia.
4. Nel riquadro di navigazione a sinistra, scegli Progetti.

5. Nella sezione Progetti, scegli il progetto che contiene il modello che desideri impacchettare.
6. Nel riquadro di navigazione a sinistra, sotto il nome del progetto, scegli Pacchetti modello Edge.
7. Nella sezione Model Packaging Job, scegli Create model packaging job.
8. Immettete le impostazioni per il pacchetto. Per ulteriori informazioni, consulta Impostazioni del pacchetto.
9. Scegli Crea un processo di confezionamento del modello.
10. Attendi il completamento del processo di imballaggio. Il lavoro è terminato quando lo stato del lavoro è Riuscito.
11. Scegli il lavoro di imballaggio nella sezione Lavori di imballaggio dei modelli.
12. Scegliete Continua l'implementazione in Greengrass per continuare la distribuzione del componente del modello in. AWS IoT Greengrass Version 2 Per ulteriori informazioni, consulta Distribuzione dei componenti su un dispositivo.

Imballaggio del modello (SDK)

Si impacchetta un modello come componente del modello creando un processo di confezionamento del modello. Per creare un processo di packaging del modello si chiama l'StartModelPackagingJobAPI. Il completamento del processo potrebbe richiedere del tempo. Per conoscere lo stato attuale, chiama DescribeModelPackagingJobe controlla il `Status` campo nella risposta.

Per informazioni sulle impostazioni dei pacchetti, vedereImpostazioni del pacchetto.

La procedura seguente mostra come avviare un processo di impacchettamento utilizzando la AWS CLI. È possibile impacchettare il modello per una piattaforma o un dispositivo di destinazione. Ad esempio codice Java, vedi StartModelPackagingJob.

Per impacchettare il modello (SDK)

1. Se non l'hai già fatto, installa e configura gli AWS CLI AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Assicurati di disporre delle autorizzazioni corrette per avviare un processo di creazione di modelli. Per ulteriori informazioni, consulta StartModelPackagingJob.
3. Utilizza i seguenti comandi CLI per impacchettare il tuo modello per un dispositivo di destinazione o una piattaforma di destinazione.

Target platform

Il seguente comando CLI mostra come impacchettare un modello per una piattaforma di destinazione con un acceleratore NVIDIA.

Modificare i seguenti valori:

- `project_name`al nome del progetto che contiene il modello che desiderate impacchettare.
- `model_version`alla versione del modello che desideri impacchettare.
- (Facoltativo) `description` a una descrizione del lavoro di confezionamento del modello.
- `architecture`all'architettura (`ARM64oX86_64`) del dispositivo AWS IoT Greengrass Version 2 principale su cui viene eseguito il componente del modello.
- `gpu_code`al codice gpu del dispositivo principale su cui si esegue il componente del modello.
- `trt_ver`alla versione TensorRT che hai installato sul tuo dispositivo principale.
- `cuda_ver`alla versione CUDA che hai installato sul tuo dispositivo principale.
- `component_name`al nome del componente del modello su AWS IoT Greengrass V2 cui si desidera creare.
- (Facoltativo) `component_version` a una versione per il componente del modello creato dal processo di confezionamento. Utilizzare il formato `major.minor.patch`. Ad esempio, 1.0.0 rappresenta la prima release principale di un componente.
- `bucket`nel bucket Amazon S3 in cui il processo di imballaggio memorizza gli artefatti dei componenti del modello.
- `prefix`nella posizione all'interno del bucket Amazon S3 in cui il processo di imballaggio archivia gli artefatti dei componenti del modello.
- (Facoltativo) `component_description` a una descrizione del componente del modello.
- (Facoltativo) `tag_key1` e `tag_key2` alle chiavi per i tag allegati al componente del modello.
- (Facoltativo) `tag_value1` e `tag_value2` ai valori chiave per i tag allegati al componente del modello.

```
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
```

Ad esempio:

```
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

Utilizza i seguenti comandi CLI per impacchettare un modello per un dispositivo di destinazione.

Modificate i seguenti valori:

- `project_name`al nome del progetto che contiene il modello che desiderate impacchettare.
- `model_version`alla versione del modello che desideri impacchettare.
- (Facoltativo) `description` a una descrizione del lavoro di confezionamento del modello.
- `component_name`al nome del componente del modello su cui desiderate creareAWS IoT Greengrass V2.
- (Facoltativo) `component_version` a una versione per il componente del modello creato dal processo di confezionamento. Utilizzare il formato `major.minor.patch`. Ad esempio, 1.0.0 rappresenta la prima release principale di un componente.

- `bucket`nel bucket Amazon S3 in cui il processo di imballaggio memorizza gli artefatti dei componenti del modello.
- `prefix`nella posizione all'interno del bucket Amazon S3 in cui il processo di imballaggio archivia gli artefatti dei componenti del modello.
- (Facoltativo) `component_description` a una descrizione del componente del modello.
- (Facoltativo) `tag_key1` e `tag_key2` alle chiavi per i tag allegati al componente del modello.
- (Facoltativo) `tag_value1` e `tag_value2` ai valori chiave per i tag allegati al componente del modello.

```
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
```

Ad esempio:

```
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. Nota il valore di JobName nella risposta. Questo valore servirà nella fase successiva. Ad esempio:

```
{
    "JobName": "6bcfd0ff-90c3-4463-9a89-6b4be3daf972"
}
```

5. `DescribeModelPackagingJob`Da utilizzare per visualizzare lo stato corrente del lavoro. Modificate quanto segue:

   - `project_name`al nome del progetto che stai utilizzando.
   - `job_name`al nome del lavoro che hai annotato nel passaggio precedente.

   ```
   aws lookoutvision describe-model-packaging-job \
       --project-name project_name \
       --job-name job_name \
       --profile lookoutvision-access
   ```

   Il processo di confezionamento del modello è completo se il valore di `Status` è`SUCCEEDED`. Se il valore è diverso, attendi un minuto e riprova.
6. Continua la distribuzione utilizzandoAWS IoT Greengrass V2. Per ulteriori informazioni, consulta Distribuzione dei componenti su un dispositivo.

Ottenere informazioni sui lavori di creazione di modelli

Puoi utilizzare la console AWS e l'SDK di Amazon Lookout for Vision per ottenere informazioni sui processi di packaging dei modelli che crei.

Argomenti

- Ottenere informazioni sul processo di confezionamento dei modelli (Console)
- Ottenere informazioni sul processo di confezionamento del modello (SDK)

Ottenere informazioni sul processo di confezionamento dei modelli (Console)

Per ottenere informazioni sul processo di confezionamento del modello (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegliere Inizia.
3. Nel riquadro di navigazione a sinistra, scegli Progetti.
4. Nella sezione Progetti, scegli il progetto che contiene il lavoro di creazione del modello che desideri visualizzare.

5. Nel riquadro di navigazione a sinistra, sotto il nome del progetto, scegli Pacchetti modello Edge.
6. Nella sezione Processo di imballaggio del modello, scegli il processo di confezionamento del modello che desideri visualizzare. Viene visualizzata la pagina dei dettagli del processo di confezionamento del modello.

Ottenere informazioni sul processo di confezionamento del modello (SDK)

È possibile utilizzare l'AWSSDK per elencare i processi di creazione del modello in un progetto e ottenere informazioni su uno specifico processo di creazione del modello.

Elenca i lavori di imballaggio dei modelli

È possibile elencare i lavori di imballaggio dei modelli in un progetto chiamando l'ListModelPackagingJobsAPI. La risposta include un elenco di ModelPackagingJobMetadataoggetti che fornisce informazioni su ogni processo di creazione del modello. È incluso anche un token di impaginazione che è possibile utilizzare per ottenere il set successivo di risultati, se l'elenco è incompleto.

Per elencare i lavori di confezionamento dei modelli

1. Se non l'hai già fatto, installa e configura gli AWS CLI AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il seguente comando CLI. Passa `project_name` al nome del progetto che desideri utilizzare.

```
aws lookoutvision list-model-packaging-jobs \
  --project-name project_name \
  --profile lookoutvision-access
```

Descrivi un lavoro di confezionamento del modello

Utilizza l'DescribeModelPackagingJobAPI per ottenere informazioni su un processo di confezionamento del modello. La risposta è un ModelPackagingDescriptionoggetto che include lo stato corrente del lavoro e altre informazioni.

Per descrivere un pacchetto

1. Se non l'hai già fatto, installa e configura gli AWS CLI AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il seguente comando CLI. Modificate quanto segue:
   - `project_name`al nome del progetto che stai utilizzando.
   - `job_name`al nome del lavoro. Quando chiami, ottieni il nome del lavoro (`JobName`) StartModelPackagingJob.

```
aws lookoutvision describe-model-packaging-job \
    --project-name project_name \
    --job-name job_name \
    --profile lookoutvision-access
```

Scrivere il componente dell'applicazione client

Un componente dell'applicazione client è un AWS IoT Greengrass Version 2 componente personalizzato che scrivi tu. Implementa la logica aziendale necessaria per utilizzare un modello Amazon Lookout for Vision su AWS IoT Greengrass Version 2 un dispositivo principale.

Per accedere a un modello, il componente dell'applicazione client utilizza il componente Lookout for Vision Edge Agent. Il componente Lookout for Vision Edge Agent fornisce un'API da utilizzare per analizzare le immagini con un modello e gestire i modelli su un dispositivo principale.

L'API Lookout for Vision Edge Agent è implementata utilizzando gRPC, un protocollo per effettuare chiamate di procedura remote. Per ulteriori informazioni, vedere gRPC. Per scrivere il codice, è possibile utilizzare qualsiasi linguaggio supportato da gRPC. Forniamo un esempio di codice Python. Per ulteriori informazioni, consulta Utilizzo di un modello nel componente dell'applicazione client.

> **Note**
>
> Il componente Lookout for Vision Edge Agent dipende dal componente del modello distribuito. Viene distribuito automaticamente sul dispositivo principale quando si distribuisce il componente del modello sul dispositivo principale.

Per scrivere un componente dell'applicazione client, effettuate le seguenti operazioni.

1. Configura il tuo ambiente per usare gRPC e installare librerie di terze parti.
2. Scrivi il codice per usare il modello.
3. Distribuisci il codice come componente personalizzato sul dispositivo principale.

Per un esempio di componente dell'applicazione client che mostra come eseguire il rilevamento delle anomalie in una pipeline GStreamer personalizzata, vedi https://github.com/awslabs/ -gstreamer. aws-greengrass-labs-lookoutvision

Configurazione dell'ambiente

Per scrivere codice client, il tuo ambiente di sviluppo si connette in remoto a un dispositivo AWS IoT Greengrass Version 2 principale su cui hai distribuito un componente del modello Amazon Lookout for Vision e le relative dipendenze. In alternativa, puoi scrivere codice su un dispositivo principale. Per ulteriori informazioni, consulta gli strumenti di sviluppo AWS IoT Greengrass e i componenti Develop AWS IoT Greengrass.

Il codice client deve utilizzare il client gRPC per accedere ad Amazon Lookout for Vision Edge Agent. Questa sezione mostra come configurare l'ambiente di sviluppo con gRPC e installare le dipendenze di terze parti necessarie per il `DetectAnomalies` codice di esempio.

Dopo aver finito di scrivere il codice client, crei un componente personalizzato e lo distribuisci sui tuoi dispositivi periferici. Per ulteriori informazioni, consulta Creazione del componente dell'applicazione client.

Argomenti

- Configurazione di gRPC
- Aggiungere dipendenze di terze parti

Configurazione di gRPC

Nel tuo ambiente di sviluppo, hai bisogno di un client gRPC da utilizzare nel codice per chiamare l'API Lookout for Vision Edge Agent. A tale scopo, si crea uno stub gRPC utilizzando un file di definizione del `.proto` servizio per Lookout for Vision Edge Agent.

> **Note**
>
> È inoltre possibile ottenere il file di definizione del servizio dal pacchetto di applicazioni Lookout for Vision Edge Agent. Il pacchetto di applicazioni viene installato quando il componente Lookout for Vision Edge Agent viene installato come dipendenza del componente del modello. Il pacchetto di applicazioni si trova in. `/greengrass/v2/packages/artifacts-unarchived/aws.iot.lookoutvision.EdgeAgent/`*`edge_agent_version`*`/lookoutvision_edge_agent` Sostituisci `edge_agent_version` con la versione dell'agente Lookout for Vision Edge che stai utilizzando. Per ottenere il pacchetto di applicazioni, è necessario distribuire Lookout for Vision Edge Agent su un dispositivo principale.

Per configurare gRPC

1. Scarica il file zip, proto.zip. Il file zip contiene il file di definizione del servizio.proto ()`edge-agent.proto`.
2. Decomprimi il contenuto.
3. Apri un prompt dei comandi e accedi alla cartella che contiene. `edge-agent.proto`
4. Usa i seguenti comandi per generare le interfacce client 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
   ```

   Se i comandi hanno esito positivo, gli stub `edge_agent_pb2_grpc.py` e `edge_agent_pb2.py` vengono creati nella directory di lavoro.
5. Scrivi il codice client che utilizza il tuo modello. Per ulteriori informazioni, consulta Utilizzo di un modello nel componente dell'applicazione client.

Aggiungere dipendenze di terze parti

Il codice di `DetectAnomalies` esempio utilizza la libreria Pillow per lavorare con le immagini. Per ulteriori informazioni, consulta Rilevamento delle anomalie utilizzando i byte delle immagini.

Usa il seguente comando per installare la libreria Pillow.

```
python3 -m pip install Pillow
```

Utilizzo di un modello nel componente dell'applicazione client

I passaggi per l'utilizzo di un modello da un componente dell'applicazione client sono simili all'utilizzo di un modello ospitato nel cloud.

1. Iniziate a eseguire il modello.
2. Rileva anomalie nelle immagini.
3. Arresta il modello, se non è più necessario.

Amazon Lookout for Vision Edge Agent fornisce API per avviare un modello, rilevare anomalie in un'immagine e interrompere un modello. Puoi anche utilizzare l'API per elencare i modelli su un dispositivo e ottenere informazioni su un modello distribuito. Per ulteriori informazioni, consulta Riferimento all'API Amazon Lookout for Vision Edge Agent.

È possibile ottenere informazioni sugli errori controllando i codici di stato gRPC. Per ulteriori informazioni, consulta Ottenere informazioni sugli errori.

Per scrivere il codice, è possibile utilizzare qualsiasi linguaggio supportato da gRPC. Forniamo un esempio di codice Python.

Argomenti

- Utilizzo dello stub nel componente dell'applicazione client
- Avvio del modello
- Rilevamento di anomalie
- Arresto del modello
- Elencare i modelli su un dispositivo
- Descrivere un modello
- Ottenere informazioni sugli errori

Utilizzo dello stub nel componente dell'applicazione client

Usa il codice seguente per configurare l'accesso al tuo modello tramite 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
```

Avvio del modello

Si avvia un modello chiamando l'StartModelAPI. L'avvio del modello potrebbe richiedere alcuni istanti. Puoi controllare lo stato attuale chiamandoDescribeModel. Il modello è in esecuzione se il valore del `status` campo è Running.

Esempio di codice

Sostituite *`component_name`* con il nome del componente del modello.

```
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)
```

Rilevamento di anomalie

L'DetectAnomaliesAPI viene utilizzata per rilevare anomalie in un'immagine.

L'`DetectAnomalies`operazione prevede che la bitmap dell'immagine venga passata nel formato compresso RGB888. Il primo byte rappresenta il canale rosso, il secondo byte rappresenta il canale verde e il terzo byte rappresenta il canale blu. Se fornite l'immagine in un formato diverso, ad esempio BGR, le previsioni di non sono corrette. DetectAnomalies

Per impostazione predefinita, OpenCV utilizza il formato BGR per le bitmap delle immagini. Se si utilizza OpenCV per acquisire immagini da analizzare, è necessario convertire `DetectAnomalies` l'immagine in formato RGB888 prima di passare l'immagine a. `DetectAnomalies`

Le immagini fornite `DetectAnomalies` devono avere le stesse dimensioni di larghezza e altezza delle immagini utilizzate per addestrare il modello.

Rilevamento delle anomalie utilizzando i byte delle immagini

È possibile rilevare anomalie in un'immagine fornendo l'immagine come byte di immagine. Nell'esempio seguente, i byte dell'immagine vengono recuperati da un'immagine memorizzata nel file system locale.

Sostituite *sample.jpg* con il nome del file di immagine che desiderate analizzare. Sostituite *component_name* con il nome del componente del modello.

```
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")
```

Rilevamento delle anomalie utilizzando un segmento di memoria condivisa

È possibile rilevare anomalie in un'immagine fornendo l'immagine come byte di immagine nel segmento di memoria condivisa POSIX. Per prestazioni ottimali, consigliamo di utilizzare la memoria condivisa per le richieste. DetectAnomalies Per ulteriori informazioni, consulta DetectAnomalies.

Arresto del modello

Se non utilizzi più il modello, l'StopModelAPI per interrompere l'esecuzione del modello.

```
stop_model_response = stub.StopModel(
    pb2.StopModelRequest(
         model_component=model_component_name
    )
)
print(f"New status of the model is {stop_model_response.status}")
```

Elencare i modelli su un dispositivo

Puoi utilizzare l'the section called “ListModels”API per elencare i modelli distribuiti su un dispositivo.

```
models_list_response = stub.ListModels(
    pb2.ListModelsRequest()
)
for model in models_list_response.models:
    print(f"Model Details {model}")
```

Descrivere un modello

Puoi ottenere informazioni su un modello distribuito su un dispositivo chiamando l'DescribeModelAPI. L'utilizzo `DescribeModel` è utile per ottenere lo stato corrente di un modello. Ad esempio, è necessario sapere se un modello è in esecuzione prima di poter chiamare`DetectAnomalies`. Per il codice di esempio, consulta Avvio del modello.

Ottenere informazioni sugli errori

I codici di stato gRPC vengono utilizzati per riportare i risultati delle API.

È possibile ottenere informazioni sull'errore rilevando l'`RpcError`eccezione, come mostrato nell'esempio seguente. Per informazioni sui codici di stato degli errori, consultate l'argomento di riferimento relativo a un'API.

```
# Error handling.
try:
    stub.DetectAnomalies(detect_anomalies_request)
except grpc.RpcError as e:
    print(f"Error code: {e.code()}, Status: {e.details()}")
```

Creazione del componente dell'applicazione client

È possibile creare il componente dell'applicazione client dopo aver generato gli stub gRPC e aver pronto il codice dell'applicazione client. Il componente creato è un componente personalizzato con cui viene distribuito su un dispositivo AWS IoT Greengrass Version 2 principale. AWS IoT Greengrass V2 Una ricetta creata descrive il componente personalizzato. La ricetta include tutte le dipendenze che devono essere distribuite. In questo caso, specificate il componente del modello in cui create. Imballaggio del modello Amazon Lookout for Vision Per ulteriori informazioni sulle ricette dei componenti, consultate il riferimento alla ricetta AWS IoT Greengrass Version 2 dei componenti.

Le procedure su questo argomento mostrano come creare il componente dell'applicazione client da un file di ricetta e pubblicarlo come componente AWS IoT Greengrass V2 personalizzato. È possibile utilizzare la AWS IoT Greengrass V2 console o l'AWSSDK per pubblicare il componente.

Per informazioni dettagliate sulla creazione di un componente personalizzato, consulta quanto segue nella AWS IoT Greengrass V2documentazione.

- Sviluppa e testa un componente sul tuo dispositivo
- Crea componenti AWS IoT Greengrass
- Pubblica componenti da distribuire sui tuoi dispositivi principali

Argomenti

- Autorizzazioni IAM per la pubblicazione di un componente dell'applicazione client
- Creare la ricetta
- Pubblicazione del componente dell'applicazione client (Console)
- Pubblicazione del componente dell'applicazione client (SDK)

Autorizzazioni IAM per la pubblicazione di un componente dell'applicazione client

Per creare e pubblicare il componente dell'applicazione client, sono necessarie le seguenti autorizzazioni IAM:

- `greengrass:CreateComponentVersion`
- `greengrass:DescribeComponent`
- `s3:PutObject`

Creare la ricetta

In questa procedura, si crea la ricetta per un semplice componente dell'applicazione client. Il codice incluso `lookoutvision_edge_agent_example.py` elenca i modelli distribuiti sul dispositivo e viene eseguito automaticamente dopo la distribuzione del componente sul dispositivo principale. Per visualizzare l'output, controllate il registro dei componenti dopo aver distribuito il componente. Per ulteriori informazioni, consulta Distribuzione dei componenti su un dispositivo. Quando sei pronto, usa questa procedura per creare la ricetta del codice che implementa la tua logica aziendale.

La ricetta viene creata come file in formato JSON o YAML. Puoi anche caricare il codice dell'applicazione client in un bucket Amazon S3.

Per creare la ricetta del componente dell'applicazione client

1. Se non l'hai già fatto, crea i file stub gRPC. Per ulteriori informazioni, consulta Configurazione di gRPC.
2. Salva il codice seguente in un file denominato `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. Crea un bucket Amazon S3 (o usa un bucket esistente) per archiviare i file di origine per il componente dell'applicazione client. Il bucket deve essere presente nel tuo AWS account e nella stessa AWS regione in cui utilizzi AWS IoT Greengrass Version 2 Amazon Lookout for Vision.
4. Caricalookoutvision_edge_agent_example.py, edge_agent_pb2_grpc.py and edge_agent_pb2.py nel bucket Amazon S3 che hai creato nel passaggio precedente. Annota il percorso Amazon S3 di ogni file. Hai creato edge_agent_pb2_grpc.py e inseritoedge_agent_pb2.py. Configurazione di gRPC
5. In un editor, crea il seguente file di ricette JSON o YAML.

   - model_componental nome del componente del modello. Per ulteriori informazioni, consulta Impostazioni dei componenti.
   - Cambia le voci URI nei percorsi S3 di lookoutvision_edge_agent_example.pyedge_agent_pb2_grpc.py, eedge_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. Salva il file JSON o YAML sul tuo computer.
7. Create il componente dell'applicazione client effettuando una delle seguenti operazioni:
   - Se vuoi usare la AWS IoT Greengrass console, fallo Pubblicazione del componente dell'applicazione client (Console).
   - Se vuoi usare l'AWSSDK, fallo Pubblicazione del componente dell'applicazione client (SDK).

Pubblicazione del componente dell'applicazione client (Console)

È possibile utilizzare la AWS IoT Greengrass V2 console per pubblicare il componente dell'applicazione client.

Per pubblicare il componente dell'applicazione client

1. Se non l'hai già fatto, crea la ricetta per il componente dell'applicazione client procedendo Creare la ricetta.
2. Apri la AWS IoT Greengrass console all'indirizzo https://console.aws.amazon.com/iot/
3. Nel riquadro di navigazione a sinistra, in Greengrass scegli Componenti.
4. In I miei componenti scegli Crea componente.
5. Nella pagina Crea componente scegli Inserisci ricetta come JSON se desideri utilizzare una ricetta in formato JSON. Scegli Inserisci ricetta come YAML se desideri utilizzare una ricetta in formato YAML.
6. In Ricetta sostituisci la ricetta esistente con la ricetta JSON o YAML in cui hai creato. Creare la ricetta
7. Scegli Crea componente.
8. Successivamente, distribuisci il componente dell'applicazione client.

Pubblicazione del componente dell'applicazione client (SDK)

È possibile pubblicare il componente dell'applicazione client utilizzando l'CreateComponentVersionAPI.

Per pubblicare il componente dell'applicazione client (SDK)

1. Se non l'hai già fatto, crea la ricetta per il componente dell'applicazione client facendo. Creare la ricetta
2. Al prompt dei comandi, immettete il seguente comando per creare il componente dell'applicazione client. Sostituisci `recipe-file` con il nome del file di ricette in Creare la ricetta cui hai creato.

   ```
   aws greengrassv2 create-component-version --inline-recipe fileb://recipe-file
   ```

   Annota l'ARN del componente nella risposta. Questo valore servirà nella fase successiva.
3. Utilizzate il comando seguente per ottenere lo stato del componente dell'applicazione client. Sostituisci `component-arn` con l'ARN che hai annotato nel passaggio precedente. Il componente dell'applicazione client è pronto se il valore di `componentState` è`DEPLOYABLE`.

   ```
   aws greengrassv2 describe-component --arn component-arn
   ```
4. Successivamente, distribuisci il componente dell'applicazione client.

Distribuzione dei componenti su un dispositivo

Per distribuire il componente del modello e il componente dell'applicazione client su un dispositivo AWS IoT Greengrass Version 2 principale, si utilizza la AWS IoT Greengrass V2 console o l'CreateDeploymentAPI. Per ulteriori informazioni, consulta Create deployments o nella Developer Guide. AWS IoT Greengrass Version 2 Per informazioni sull'aggiornamento di un componente distribuito su un dispositivo principale, consulta Revise deployments.

Argomenti

- Autorizzazioni IAM per la distribuzione dei componenti
- Implementazione dei componenti (console)
- Distribuzione dei componenti (SDK)

Autorizzazioni IAM per la distribuzione dei componenti

Per distribuire un componente con AWS IoT Greengrass V2 te sono necessarie le seguenti autorizzazioni:

- `greengrass:ListComponents`
- `greengrass:ListComponentVersions`
- `greengrass:ListCoreDevices`
- `greengrass:CreateDeployment`
- `greengrass:GetDeployment`
- `greengrass:ListDeployments`

`CreateDeployment`e `GetDeployment` hanno azioni dipendenti. Per ulteriori informazioni, consulta Azioni definite da AWS IoT Greengrass V2.

Per informazioni sulla modifica delle autorizzazioni IAM, consulta Modifica delle autorizzazioni per un utente.

Implementazione dei componenti (console)

Utilizzare la procedura seguente per distribuire il componente dell'applicazione client su un dispositivo principale. L'applicazione client dipende dal componente del modello (che a sua volta dipende dall'agente Lookout for Vision Edge). L'implementazione del componente dell'applicazione client avvia anche la distribuzione del componente modello e dell'agente Lookout for Vision Edge.

> **Note**
>
> È possibile aggiungere i componenti a una distribuzione esistente. È inoltre possibile distribuire componenti in un gruppo di oggetti.

Per eseguire questa procedura, è necessario disporre di un dispositivo AWS IoT Greengrass V2 principale configurato. Per ulteriori informazioni, consulta Configurazione del dispositivo principale AWS IoT Greengrass Version 2.

Per distribuire i componenti su un dispositivo

1. Apri la AWS IoT Greengrass console all'indirizzo https://console.aws.amazon.com/iot/.
2. Nel riquadro di navigazione a sinistra, in Greengrass, scegli Deployments.
3. In Distribuzioni scegli Crea.
4. Nella pagina Specificare la destinazione, procedi come segue:

1. In Informazioni sulla distribuzione, inserisci o modifica il nome descrittivo per la tua distribuzione.
2. In Destinazione di distribuzione, seleziona Dispositivo principale e inserisci un nome di destinazione.
3. Seleziona Successivo.

5. Nella pagina Seleziona componenti, procedi come segue:
   1. In I miei componenti, scegliete il nome del componente dell'applicazione client (`com.lookoutvison.EdgeAgentPythonExample`).
   2. Seleziona Next (Successivo).
6. Nella pagina Configura componenti, mantieni la configurazione corrente e scegli Avanti.
7. Nella pagina Configura impostazioni avanzate, mantieni le impostazioni correnti e scegli Avanti.
8. Nella pagina Revisione, scegli Distribuisci per iniziare a distribuire il componente.

Verifica dello stato di implementazione (console)

È possibile controllare lo stato della distribuzione dalla AWS IoT Greengrass V2 console. Se il componente dell'applicazione client utilizza la ricetta e il codice di esempio dithe section called “Creazione del componente dell'applicazione client”, visualizza il registro del componente dell'applicazione client al termine della distribuzione. In caso di successo, il registro include un elenco dei modelli Lookout for Vision distribuiti sul componente.

Per informazioni sull'utilizzo dell'AWSSDK per verificare lo stato della distribuzione, consulta Verifica dello stato della distribuzione.

Per verificare lo stato della distribuzione

1. Apri la AWS IoT Greengrass console all'indirizzo https://console.aws.amazon.com/iot/
2. Nel riquadro di navigazione a sinistra, scegli Dispositivi principali.
3. In Greengrass core devices scegli il tuo dispositivo principale.
4. Scegli la scheda Distribuzioni per visualizzare lo stato attuale della distribuzione.
5. Una volta completata la distribuzione (lo stato è Completato), apri una finestra di terminale sul dispositivo principale e visualizza il registro dei componenti dell'applicazione client all'indirizzo. `/greengrass/v2/logs/com.lookoutvison.EdgeAgentPythonExample.log`

Se la distribuzione utilizza la ricetta e il codice di esempio, il registro include l'output di. `lookoutvision_edge_agent_example.py` Ad esempio:

```
Model Details model_component:"ModelComponent"
```

Distribuzione dei componenti (SDK)

Utilizza la seguente procedura per distribuire il componente dell'applicazione client, il componente modello e Amazon Lookout for Vision Edge Agent sul tuo dispositivo principale.

1. Crea un `deployment.json` file per definire la configurazione di distribuzione per i tuoi componenti. Questo file dovrebbe essere simile all'esempio seguente.

```
{
  "targetArn":"targetArn",
  "components": {
    "com.lookoutvison.EdgeAgentPythonExample": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
      }
    }
  }
}
```

   - Nel `targetArn` campo, sostituisci *`targetArn`*con l'Amazon Resource Name (ARN) dell'oggetto o del gruppo di oggetti a cui destinare la distribuzione, nel seguente formato:
     - Cosa: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Gruppo di cose: `arn:aws:iot:`*`region`*`:`*`account-id:`*`thinggroup/`*`thingGroupName`*
2. Controlla se il target di distribuzione ha una distribuzione esistente che desideri modificare. Esegui questa operazione:

   a. Esegui il comando seguente per elencare le distribuzioni per l'obiettivo di distribuzione. Sostituisci `targetArn` con l'Amazon Resource Name (ARN) dell'oggetto o del gruppo di AWS oggetti IoT di destinazione. Per ottenere gli ARN degli oggetti nell'attuale regione AWS, usa il comando`aws iot list-things`.

```
aws greengrassv2 list-deployments --target-arn targetArn
```

La risposta contiene un elenco con la distribuzione più recente per l'obiettivo. Se la risposta è vuota, la destinazione non dispone di una distribuzione esistente e puoi saltare al passaggio 3. Altrimenti, copia il `deploymentId` codice dalla risposta da utilizzare nel passaggio successivo.

b. Esegui il comando seguente per ottenere i dettagli della distribuzione. Questi dettagli includono metadati, componenti e configurazione del processo. Sostituisci `deploymentId` con l'ID del passaggio precedente.

```
aws greengrassv2 get-deployment --deployment-id deploymentId
```

c. Copia una delle seguenti coppie chiave-valore dalla risposta del comando precedente in deployment.json. È possibile modificare questi valori per la nuova distribuzione.

- `deploymentName`— Il nome della distribuzione.
- `components`— I componenti della distribuzione. Per disinstallare un componente, rimuovilo da questo oggetto.
- `deploymentPolicies`— Le politiche della distribuzione.
- `tags`— I tag della distribuzione.

3. Esegui il comando seguente per distribuire i componenti sul dispositivo. Annota il valore di `deploymentId` nella risposta.

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

4. Esegui il comando seguente per ottenere lo stato della distribuzione. Passa `deployment-id` al valore che hai annotato nel passaggio precedente. La distribuzione è stata completata correttamente se il valore di `deploymentStatus` è. `COMPLETED`

```
aws greengrassv2 get-deployment --deployment-id deployment-id
```

5. Una volta completata la distribuzione, aprite una finestra di terminale sul dispositivo principale e visualizzate il registro dei componenti dell'applicazione client all'indirizzo. `/greengrass/v2/logs/com.lookoutvison.EdgeAgentPythonExample.log` Se la distribuzione utilizza la ricetta e il codice di esempio, il registro include l'output di. `lookoutvision_edge_agent_example.py` Ad esempio:

```
Model Details model_component:"ModelComponent"
```

Riferimento all'API Amazon Lookout for Vision Edge Agent

Questa sezione è il riferimento API per Amazon Lookout for Vision Edge Agent.

Rilevamento di anomalie con un modello

L'DetectAnomaliesAPI viene utilizzata per rilevare anomalie nelle immagini utilizzando un modello in esecuzione su un dispositivo principale. AWS IoT Greengrass Version 2

Ottenere informazioni sul modello

API che ottengono informazioni sui modelli distribuiti su un dispositivo principale.

- ListModels
- DescribeModel

Esecuzione di un modello

API per avviare e arrestare un modello Amazon Lookout for Vision distribuito su un dispositivo principale.

- StartModel
- StopModel

DetectAnomalies

Rileva anomalie nell'immagine fornita.

La risposta di `DetectAnomalies` include una previsione booleana secondo cui l'immagine contiene una o più anomalie e un valore di confidenza per la previsione. Se il modello è un modello di segmentazione, la risposta include quanto segue:

- Un'immagine di maschera che copre ogni tipo di anomalia con un colore unico. È possibile `DetectAnomalies` archiviare l'immagine della maschera nella memoria condivisa o restituirla come byte di immagine.

- L'area percentuale dell'immagine coperta da un tipo di anomalia.
- Il colore esadecimale per un tipo di anomalia sull'immagine della maschera.

> **Note**
>
> Il modello con cui si utilizza `DetectAnomalies` deve essere in esecuzione. È possibile ottenere lo stato attuale chiamandoDescribeModel. Per iniziare a eseguire un modello, vedereStartModel.

`DetectAnomalies`supporta bitmap (immagini) compresse in formato RGB888 interlacciato. Il primo byte rappresenta il canale rosso, il secondo byte rappresenta il canale verde e il terzo byte rappresenta il canale blu. Se fornite l'immagine in un formato diverso, ad esempio BGR, le previsioni di non sono corrette. DetectAnomalies

Per impostazione predefinita, OpenCV utilizza il formato BGR per le bitmap delle immagini. Se si utilizza OpenCV per acquisire immagini da analizzare, è necessario convertire `DetectAnomalies` l'immagine in formato RGB888 prima di passare l'immagine a. `DetectAnomalies`

La dimensione minima supportata dell'immagine è 64x64 pixel. La dimensione massima supportata per l'immagine è 4096x4096 pixel.

È possibile inviare l'immagine nel messaggio protobuf o tramite un segmento di memoria condivisa. La serializzazione di immagini di grandi dimensioni nel messaggio protobuf può aumentare significativamente la latenza delle chiamate a. `DetectAnomalies` Per una latenza minima, si consiglia di utilizzare la memoria condivisa.

```
rpc DetectAnomalies(DetectAnomaliesRequest) returns (DetectAnomaliesResponse);
```

DetectAnomaliesRequest

I parametri di input per`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

L'immagine con `DetectAnomalies` cui vuoi analizzare.

width

La larghezza dell'immagine in pixel.

height

L'altezza dell'immagine in pixel.

byte_data

Byte di immagine passati nel messaggio protobuf.

shared_memory_handle

Byte di immagine passati nel segmento di memoria condivisa.

SharedMemoryHandle

Rappresenta un segmento di memoria condivisa POSIX.

name

Il nome del segmento di memoria POSIX. Per informazioni sulla creazione di memoria condivisa, vedete shm_open.

formato

La dimensione del buffer di immagine in byte a partire dall'offset.

offset

L'offset, in byte, rispetto all'inizio del buffer di immagini dall'inizio del segmento di memoria condivisa.

AnomalyMaskParams

Parametri per l'emissione di una maschera di anomalia. (Modello di segmentazione).

shared_memory_handle

Contiene i byte dell'immagine per la maschera, se non è stato fornito. `shared_memory_handle`

DetectAnomaliesRequest

model_component

Il nome del AWS IoT Greengrass V2 componente che contiene il modello che desideri utilizzare.

bitmap

L'immagine con `DetectAnomalies` cui vuoi analizzare.

anomaly_mask_params

Parametri opzionali per l'output della maschera. (Modello di segmentazione).

DetectAnomaliesResponse

La risposta di`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;
}
```

Anomalia

Rappresenta un'anomalia rilevata su un'immagine. (Modello di segmentazione).

name

Il nome di un tipo di anomalia trovato in un'immagine. `name`corrisponde a un tipo di anomalia nel set di dati di addestramento. Il servizio inserisce automaticamente il tipo di anomalia in background nella risposta da. DetectAnomalies

pixel_anomaly

Informazioni sulla maschera di pixel che copre un tipo di anomalia.

PixelAnomaly

Informazioni sulla maschera di pixel che copre un tipo di anomalia. (Modello di segmentazione).

area_percentuale totale

L'area percentuale dell'immagine coperta dal tipo di anomalia.

colore esadecimale

Un valore di colore esadecimale che rappresenta il tipo di anomalia nell'immagine. Il colore corrisponde al colore del tipo di anomalia utilizzato nel set di dati di addestramento.

DetectAnomalyResult

is_anomalo

Indica se l'immagine contiene un'anomalia. `true`se l'immagine contiene un'anomalia. `false`se l'immagine è normale.

confidence

La fiducia che `DetectAnomalies` si ha nell'accuratezza della previsione. `confidence`è un valore in virgola mobile compreso tra 0 e 1.

anomaly_mask

se shared_memory_handle non è stato fornito, contiene i byte dell'immagine per la maschera. (Modello di segmentazione).

anomalie

Un elenco di 0 o più anomalie rilevate nell'immagine di input. (Modello di segmentazione).

anomaly_score

Un numero che quantifica in che misura le anomalie previste per un'immagine si discostano da un'immagine senza anomalie. `anomaly_score`è un valore float compreso tra `0.0` a (deviazione minima da un'immagine normale) a 1,0 (deviazione massima da un'immagine normale). Amazon Lookout for Vision restituisce un valore `anomaly_score` per, anche se la previsione per un'immagine è normale.

anomaly_threshold

Un numero (float) che determina quando la classificazione prevista per un'immagine è normale o anomala. Le immagini con un `anomaly_score` valore uguale o superiore al valore di `anomaly_threshold` sono considerate anomale. Un `anomaly_score` valore inferiore

`anomaly_threshold` indica un'immagine normale. Il valore utilizzato da un modello viene calcolato da Amazon Lookout for Vision quando si addestra il modello. `anomaly_threshold` Non puoi impostare o modificare il valore di `anomaly_threshold`

Codici di stato

Codice	Numero	Descrizione
OK	0	`DetectAnomalies` ha fatto una previsione con successo
UNKNOWN	2	Si è verificato un errore sconosciuto.
INVALID_ARGUMENT	3	Uno o più parametri di input non sono validi. Controlla il messaggio di errore per maggiori dettagli.
NOT_FOUND	5	Non è stato trovato un modello con il nome specificato.
RESOURCE_EXHAUSTED	8	Non ci sono risorse sufficienti per eseguire questa operazione. Ad esempio, The Lookout for Vision Edge Agent non riesce a tenere il passo con la frequenza delle chiamate versoDetectAnomalies . Controlla il messaggio di errore per maggiori dettagli.
FAILED_PRECONDITION	9	`DetectAnomalies` è stato chiamato per un modello che non si trova nello stato RUNNING.
INTERNO	13	Si è verificato un errore interno.

DescribeModel

Descrive un modello Amazon Lookout for Vision distribuito su AWS IoT Greengrass Version 2 un dispositivo principale.

```
rpc DescribeModel(DescribeModelRequest) returns (DescribeModelResponse);
```

DescribeModelRequest

```
message DescribeModelRequest {
  string model_component = 1;
}
```

model_component

Il nome del AWS IoT Greengrass V2 componente che contiene il modello che desideri descrivere.

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

Il nome del AWS IoT Greengrass Version 2 componente che contiene il modello Amazon Lookout for Vision.

lookout_vision_model_arn

L'Amazon Resource Name ARN del modello Amazon Lookout for Vision utilizzato per generare il componente. AWS IoT Greengrass V2

status

Lo stato attuale del modello. Per ulteriori informazioni, consulta ModelStatus.

status_message

Il messaggio di stato del modello.

Codici di stato

Codice	Numero	Descrizione
OK	0	La chiamata ha avuto successo.
UNKNOWN	2	Si è verificato un errore sconosciuto.
INVALID_ARGUMENT	3	Uno o più parametri di input non sono validi. Controlla il messaggio di errore per maggiori dettagli.
NOT_FOUND	5	Non è stato trovato un modello con il nome fornito.
INTERNO	13	Si è verificato un errore interno.

ListModels

Elenca i modelli distribuiti su un dispositivo AWS IoT Greengrass Version 2 principale.

```
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

Il nome del AWS IoT Greengrass Version 2 componente che contiene un modello Amazon Lookout for Vision.

lookout_vision_model_arn

L'Amazon Resource Name (ARN) del modello Amazon Lookout for Vision utilizzato per generare il componente. AWS IoT Greengrass V2

status

Lo stato attuale del modello. Per ulteriori informazioni, consulta ModelStatus.

status_message

Il messaggio di stato del modello.

Codici di stato

Codice	Numero	Descrizione
OK	0	La chiamata ha avuto successo.
UNKNOWN	2	Si è verificato un errore sconosciuto.
INTERNO	13	Si è verificato un errore interno.

StartModel

Avvia un modello in esecuzione su un dispositivo AWS IoT Greengrass Version 2 principale. L'avvio del modello potrebbe richiedere alcuni istanti. Per verificare lo stato attuale chiamaDescribeModel. Il modello è in esecuzione se il `Status` campo lo è`RUNNING`.

Il numero di modelli che è possibile eseguire contemporaneamente dipende dalle specifiche hardware del dispositivo principale.

```
rpc StartModel(StartModelRequest) returns (StartModelResponse);
```

StartModelRequest

```
message StartModelRequest {
  string model_component = 1;
}
```

model_component

Il nome del AWS IoT Greengrass Version 2 componente che contiene il modello da avviare.

StartModelResponse

```
message StartModelResponse {
  ModelStatus status = 1;
```

```
}
```

status

Lo stato attuale del modello. La risposta è `STARTING` se la chiamata ha esito positivo. Per ulteriori informazioni, consulta ModelStatus.

Codici di stato

Codice	Numero	Descrizione
OK	0	Il modello sta iniziando
UNKNOWN	2	Si è verificato un errore sconosciuto.
INVALID_ARGUMENT	3	Uno o più parametri di input non sono validi. Controlla il messaggio di errore per maggiori dettagli.
NOT_FOUND	5	Non è stato trovato un modello con il nome fornito.
RESOURCE_EXHAUSTED	8	Non ci sono risorse sufficienti per eseguire questa operazione. Ad esempio, non c'è abbastanza memoria per caricare il modello. Controlla il messaggio di errore per maggiori dettagli.
FAILED_PRECONDITION	9	Il metodo è stato chiamato per un modello che non si trova nello stato STOPPED o FAILED.
INTERNO	13	Si è verificato un errore interno.

StopModel

Interrompe l'esecuzione di un modello su un dispositivo AWS IoT Greengrass Version 2 principale. `StopModel`ritorna dopo l'arresto del modello. Il modello si è fermato correttamente se il `Status` campo nella risposta è`STOPPED`.

```
  rpc StopModel(StopModelRequest) returns (StopModelResponse);
```

StopModelRequest

```
message StopModelRequest {
  string model_component = 1;
}
```

model_component

Il nome del AWS IoT Greengrass Version 2 componente che contiene il modello che vuoi fermare.

StopModelResponse

```
message StopModelResponse {
  ModelStatus status = 1;
}
```

status

Lo stato attuale del modello. La risposta è `STOPPED` se la chiamata ha esito positivo. Per ulteriori informazioni, consulta ModelStatus.

Codici di stato

Codice	Numero	Descrizione
OK	0	Il modello si sta arrestando.
UNKNOWN	2	Si è verificato un errore sconosciuto.

Codice	Numero	Descrizione
INVALID_ARGUMENT	3	Uno o più parametri di input non sono validi. Controlla il messaggio di errore per maggiori dettagli.
NOT_FOUND	5	Non è stato trovato un modello con il nome fornito.
FAILED_PRECONDITION	9	Il metodo è stato chiamato per un modello che non si trova nello stato RUNNING.
INTERNO	13	Si è verificato un errore interno.

ModelStatus

Lo stato di un modello distribuito su un dispositivo AWS IoT Greengrass Version 2 principale. Per conoscere lo stato attuale, chiamaDescribeModel.

```
enum ModelStatus {
  STOPPED = 0;
  STARTING = 1;
  RUNNING = 2;
  FAILED = 3;
  STOPPING = 4;
}
```

Utilizzo Lookout for Vision di

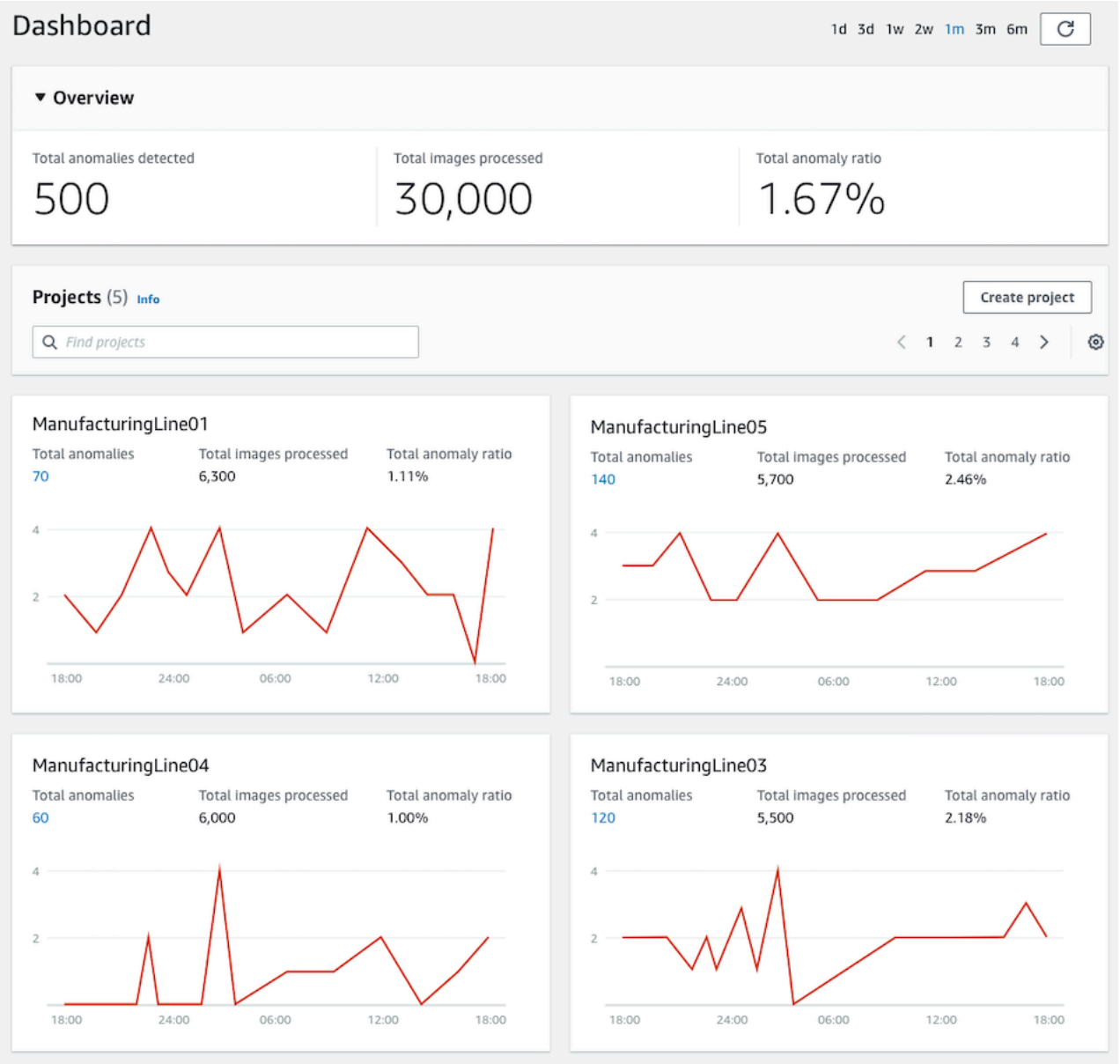
La dashboard fornisce una panoramica delle metriche per i tuoi progetti Amazon Lookout for Vision, come il numero totale di anomalie rilevate nell'ultima settimana. Con la dashboard ottieni una panoramica di tutti i tuoi progetti e una panoramica per ogni singolo progetto. Puoi scegliere la sequenza temporale in base alla quale mostrare le metriche. Puoi anche utilizzare il pannello di

La sezione Panoramica mostra il numero totale di progetti, il numero totale di immagini e il numero totale di immagini rilevate da tutti i tuoi progetti.

La sezione Progetti mostra le seguenti informazioni generali per i singoli progetti:

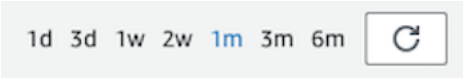
- Il numero totale di anomalie rilevate.
- Il numero totale di immagini elaborate.
- Il rapporto di anomalia totale (ovvero la percentuale di immagini rilevate con un'anomalia).
- Un grafico mostra i rilevamenti delle anomalie nel periodo di tempo selezionato.

Puoi anche ottenere ulteriori informazioni su un progetto.

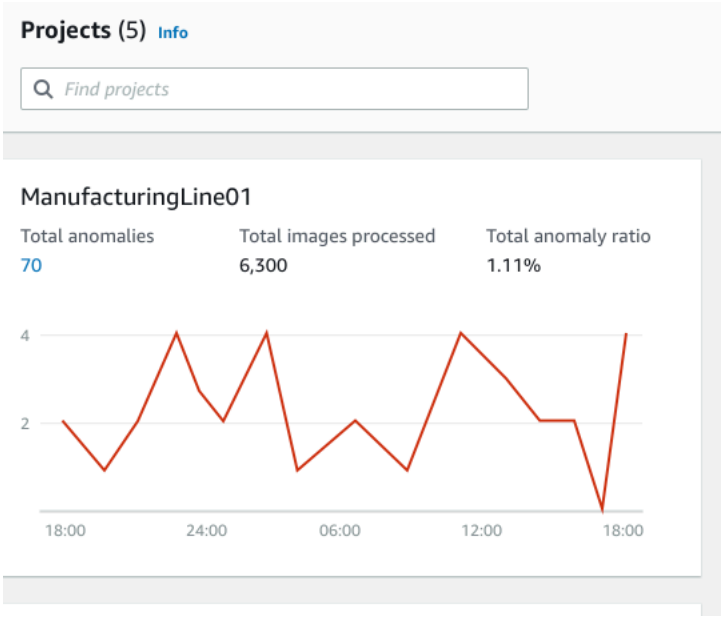


Per utilizzare il pannello

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegliere Inizia.
3. Nel pannello di navigazione sinistro scegli Dashboard.
4. Per visualizzare le metriche in un intervallo di tempo specifico, procedi come indicato di seguito:
   a. Scegli l'intervallo di tempo nella parte superiore destra della dashboard.
   b. Scegliere il pulsante di aggiornamento per mostrare il pannello di



5. Per ottenere ulteriori dettagli su un progetto, scegli il nome del progetto nella sezione Progetti (ad esempio`ManufacturingLine01`).



6. Per creare un progetto, scegli Crea progetto nella sezione Progetti.

Gestione delle risorse Amazon Lookout for Vision

Puoi gestire le tue risorse Amazon Lookout for Vision utilizzando la console o AWS l'SDK. Amazon Lookout for Vision dispone delle seguenti risorse:

- Progetti
- Set di dati
- Modelli
- Rilevamenti di prova

> **Note**
>
> Non è possibile eliminare un'attività di rilevamento delle versioni di prova. Inoltre, non puoi gestire i rilevamenti delle versioni di prova utilizzando l'AWSSDK.

Argomenti

Visualizzazione dei progetti

Puoi ottenere un elenco di progetti Amazon Lookout for Vision e informazioni sui singoli progetti dalla console o utilizzando AWS l'SDK.

> **Note**
>
> L'elenco dei progetti alla fine è coerente. Se si crea o si elimina un progetto, potrebbe essere necessario attendere qualche istante prima che l'elenco dei progetti sia aggiornato.

Visualizzazione dei progetti (console)

Esegui i passaggi della procedura seguente per visualizzare i tuoi progetti nella console.

Per visualizzare i tuoi progetti

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegli Inizia.
3. Nel riquadro di navigazione a sinistra, scegli Progetti. Viene mostrata la vista dei progetti.
4. Scegli il nome del progetto per visualizzarne i dettagli.

Visualizzazione dei progetti (SDK)

Un progetto gestisce i set di dati e i modelli per un singolo caso d'uso. Ad esempio, il rilevamento di anomalie nelle parti di macchine. L'esempio seguente chiama `ListProjects` per ottenere un elenco dei tuoi progetti.

Per visualizzare i tuoi progetti (SDK)

1. Se non l'hai già fatto, installa e configura gli AWS CLI AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il seguente codice di esempio per visualizzare i tuoi progetti.

   CLI

   Usa il `list-projects` comando per elencare i progetti nel tuo account.

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

Usa il `describe-project` comando per ottenere informazioni su un progetto.

Cambia il valore `project-name` di con il nome del progetto che vuoi descrivere.

```
aws lookoutvision describe-project --project-name project_name \
  --profile lookoutvision-access
```

Python

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

```
    @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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

```
/**
 * 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;

}
```

Eliminazione di un progetto

È possibile eliminare un progetto dalla pagina di visualizzazione dei progetti nella console o utilizzando l'`DeleteProject`operazione.

Le immagini a cui fanno riferimento i set di dati di un progetto non vengono eliminate.

Eliminazione di un progetto (console)

Per eliminare un progetto, utilizzare la procedura seguente. Se si utilizza la procedura della console, le versioni dei modelli e i set di dati associati vengono eliminati automaticamente.

Per eliminare un progetto

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegli Inizia.
3. Nel riquadro di navigazione a sinistra, scegli Progetti.
4. Nella pagina Progetti, seleziona il progetto che desideri eliminare.
5. Nella parte superiore della pagina, seleziona Delete (Elimina).
6. Nella finestra di dialogo Elimina, inserite delete per confermare che desiderate eliminare il progetto.
7. Se necessario, scegliete di eliminare tutti i set di dati e i modelli associati.

8. Seleziona Delete project (Elimina progetto).

Eliminazione di un progetto (SDK)

Puoi eliminare un progetto Amazon Lookout for Vision DeleteProjectchiamando e fornendo il nome del progetto che desideri eliminare.

Prima di poter eliminare un progetto, devi eliminare tutti i modelli del progetto. Per ulteriori informazioni, consulta Eliminazione di un modello (SDK). È inoltre necessario eliminare i set di dati associati al modello. Per ulteriori informazioni, consulta Eliminazione di un set di dati.

L'eliminazione del progetto potrebbe richiedere alcuni minuti. Durante questo periodo, lo stato del progetto è`DELETING`. Il progetto viene eliminato se una chiamata successiva a `DeleteProject` non include il progetto eliminato.

Per eliminare un progetto (SDK)

1. Se non l'hai già fatto, installa e configura gli AWS CLI AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il codice seguente per eliminare un progetto.

   AWS CLI

   Modificate `project-name` il valore di con il nome del progetto che desiderate eliminare.

   ```
   aws lookoutvision delete-project --project-name project_name \
     --profile lookoutvision-access
   ```

   Python

   Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

   ```
       @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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

```
/**
 * 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();
```

```
}
```

Visualizzazione dei set di dati

Un progetto può avere un singolo set di dati utilizzato per addestrare e testare il modello. In alternativa, è possibile disporre di set di dati di addestramento e test separati. Puoi usare la console per visualizzare i tuoi set di dati. È inoltre possibile utilizzare l'`DescribeDataset`operazione per ottenere informazioni su un set di dati (formazione o test).

Visualizzazione dei set di dati in un progetto (console)

Esegui i passaggi della procedura seguente per visualizzare i set di dati del progetto nella console.

Per visualizzare i set di dati (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegli Inizia.
3. Nel riquadro di navigazione a sinistra, scegli Progetti.
4. Nella pagina Progetti, seleziona il progetto che contiene i set di dati che desideri visualizzare.
5. Nel riquadro di navigazione a sinistra, scegli Dataset per visualizzare i dettagli del set di dati. Se disponi di un set di dati di addestramento e di test, viene mostrata una scheda per ogni set di dati.

Visualizzazione dei set di dati in un progetto (SDK)

È possibile utilizzare l'`DescribeDataset`operazione per ottenere informazioni sul set di dati di addestramento o test associato a un progetto.

Per visualizzare i set di dati (SDK)

1. Se non l'hai già fatto, installa e configura gli AWS CLI SDK. AWS Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il codice di esempio seguente per visualizzare un set di dati.

CLI

Modificate i seguenti valori:

- `project-name`al nome del progetto che contiene il modello che desiderate visualizzare.
- `dataset-type`al tipo di set di dati che si desidera visualizzare (`trainotest`).

```
aws lookoutvision describe-dataset --project-name project name\
  --dataset-type train or test \
  --profile lookoutvision-access
```

Python

Questo codice è tratto dal repository degli esempi GitHub di AWS Documentation SDK. Vedi l'esempio completo qui.

```
    @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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

```
/**
 * 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;

}
```

Aggiungere immagini al set di dati

Dopo aver creato un set di dati, potresti voler aggiungere altre immagini al set di dati. Ad esempio, se la valutazione del modello indica che il modello è scadente, è possibile migliorarne la qualità aggiungendo altre immagini. Se avete creato un set di dati di test, l'aggiunta di altre immagini può aumentare la precisione delle metriche prestazionali del modello.

Riaddestrate il modello dopo aver aggiornato i set di dati.

Argomenti

- Aggiungere altre immagini
- Aggiungere altre immagini (SDK)

Aggiungere altre immagini

Puoi aggiungere altre immagini ai tuoi set di dati caricando immagini dal tuo computer locale. Per aggiungere altre immagini etichettate con l'SDK, utilizzate l'operazione. UpdateDatasetEntries

Per aggiungere altre immagini al tuo set di dati (console)

1. Scegli Azioni e seleziona il set di dati a cui desideri aggiungere immagini.
2. Scegli le immagini che desideri caricare nel set di dati. Puoi trascinare le immagini o scegliere le immagini che desideri caricare dal tuo computer locale. Puoi caricare fino a 30 immagini alla volta.
3. Scegli Carica immagini.
4. Sceglie Save changes (Salva modifiche).

Quando hai finito di aggiungere altre immagini, devi etichettarle in modo che possano essere utilizzate per addestrare il modello. Per ulteriori informazioni, consulta Classificazione delle immagini (console).

Aggiungere altre immagini (SDK)

Per aggiungere altre immagini etichettate con l'SDK, utilizzate l'operazione. UpdateDatasetEntries Fornite un file manifesto che contiene le immagini che desiderate aggiungere. È inoltre possibile aggiornare le immagini esistenti specificando l'immagine nel `source-ref` campo della riga JSON nel file manifest. Per ulteriori informazioni, consulta Creazione di un file manifest.

Per aggiungere altre immagini a un set di dati (SDK)

1. Se non l'hai già fatto, installa e configura gli AWS CLI e gli AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il codice di esempio seguente per aggiungere altre immagini a un set di dati.

   CLI

   Modificate i seguenti valori:

   - `project-name`al nome del progetto che contiene il set di dati che desideri aggiornare.
   - `dataset-type`al tipo di set di dati che desideri aggiornare (`trainotest`).
   - `changes`alla posizione del file manifesto che contiene gli aggiornamenti dei set di dati.

```
aws lookoutvision update-dataset-entries\
    --project-name project\
    --dataset-type train or test\
    --changes fileb://manifest file \
    --profile lookoutvision-access
```

Python

Questo codice è tratto dal repository degli esempi GitHub di AWS Documentation SDK. Vedi l'esempio completo qui.

```
    @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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

```
/**
 * 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. Ripeti il passaggio precedente e fornisci i valori per l'altro tipo di set di dati.

Rimuovere immagini dal set di dati

Non puoi eliminare le immagini direttamente da un set di dati. È invece necessario eliminare il set di dati esistente e creare un nuovo set di dati senza le immagini che si desidera rimuovere. Il modo in cui rimuovi le immagini dipende da come le hai importate nel set di dati esistente (file manifest, bucket Amazon S3 o computer locale).

Puoi anche utilizzare l'AWSSDK per rimuovere le immagini. Ciò è utile quando si crea un modello di segmentazione delle immagini senza un file manifesto di segmentazione delle immagini, in quanto non è necessario ridisegnare le maschere di immagine con la console Amazon Lookout for Vision.

Argomenti

- Rimozione di immagini da un set di dati (Console)
- Rimozione di immagini da un set di dati (SDK)

Rimozione di immagini da un set di dati (Console)

Utilizza la seguente procedura per rimuovere immagini da un set di dati con la console Amazon Lookout for Vision.

Per rimuovere immagini da un set di dati (console)

1. Apri la galleria dei set di dati del progetto.
2. Annota il nome di ogni immagine che desideri rimuovere.
3. Eliminare il set di dati esistente.
4. Completa una delle seguenti operazioni:
   - Se hai creato il set di dati con un file manifesto, fai:
     a. In un editor di testo, apri il file manifesto che hai usato per creare il set di dati.
     b. Rimuovi la riga JSON per ogni immagine che hai annotato nel passaggio 2. È possibile identificare la riga JSON di un'immagine controllando il `source-ref` campo.

c. Salva il file manifest.

d. Crea un nuovo set di dati con il file manifest aggiornato.

- Se hai creato il set di dati da immagini importate da un bucket Amazon S3, esegui:

  a. Elimina le immagini che hai annotato nel passaggio 2 dal bucket Amazon S3.

  b. Crea un nuovo set di dati con le immagini rimanenti nel bucket Amazon S3. Se classifichi le immagini in base al nome della cartella, non è necessario classificare le immagini nel passaggio successivo.

  c. Completa una delle seguenti operazioni:

     - Se state creando un modello di classificazione delle immagini, classificate ogni immagine senza etichetta.
     - Se state creando un modello di segmentazione delle immagini, classificate e segmentate ogni immagine senza etichetta.

- Se hai creato il set di dati da immagini importate da un computer locale, esegui:

  a. Sul tuo computer, crea una cartella con le immagini che desideri utilizzare. Non includere le immagini che desideri rimuovere dal set di dati. Per ulteriori informazioni, consulta Creazione di un set di dati utilizzando immagini archiviate nel computer locale.

  b. Crea il set di dati con le immagini nella cartella creata nel passaggio 4.a.

  c. Completa una delle seguenti operazioni:

     - Se state creando un modello di classificazione delle immagini, classificate ogni immagine senza etichetta.
     - Se state creando un modello di segmentazione delle immagini, classificate e segmentate ogni immagine senza etichetta.

5. Addestra il modello.

Rimozione di immagini da un set di dati (SDK)

Puoi utilizzare l'AWSSDK per rimuovere immagini da un set di dati.

Per rimuovere immagini da un set di dati (SDK)

1. Apri la galleria dei set di dati del progetto.
2. Annota il nome di ogni immagine che desideri rimuovere.
3. Esporta le righe JSON per il set di dati utilizzando l'ListDatasetEntriesoperazione.

4. Crea un file manifest con le righe JSON esportate.
5. In un editor di testo, aprite il file manifest.
6. Rimuovi la riga JSON per ogni immagine che hai annotato nel passaggio 2. È possibile identificare la riga JSON di un'immagine controllando il `source-ref` campo.
7. Salva il file manifest.
8. Eliminare il set di dati esistente.
9. Crea un nuovo set di dati con il file manifest aggiornato.
10. Addestra il modello.

Eliminazione di un set di dati

È possibile eliminare un set di dati da un progetto utilizzando la console o l'operazione. `DeleteDataset` Le immagini a cui fa riferimento un set di dati non vengono eliminate. Se elimini il set di dati di test da un progetto che include un set di dati di addestramento e uno di test, il progetto torna a essere un singolo progetto di set di dati: il set di dati rimanente viene suddiviso durante l'addestramento per creare un set di dati di addestramento e test. Se elimini il set di dati di addestramento, non puoi addestrare un modello nel progetto finché non crei un nuovo set di dati di addestramento.

Eliminazione di un set di dati (console)

Eseguite i passaggi della procedura seguente per eliminare un set di dati. Se si eliminano tutti i set di dati in un progetto, viene visualizzata la pagina Crea set di dati.

Per eliminare un set di dati (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegli Inizia.
3. Nel riquadro di navigazione a sinistra, scegli Progetti.
4. Nella pagina Progetti, seleziona il progetto che contiene il set di dati che desideri eliminare.
5. Nel riquadro di navigazione a sinistra, scegli Dataset.
6. Scegli Azioni, quindi seleziona il set di dati che desideri eliminare.
7. Nella finestra di dialogo Elimina, inserisci delete per confermare che desideri eliminare il set di dati.

8. Scegliete Elimina set di dati di allenamento o Elimina set di dati di test per eliminare il set di dati.

Eliminazione di un set di dati (SDK)

Usa l'`DeleteDataset`operazione per eliminare un set di dati.

Per eliminare un set di dati (SDK)

1. Se non l'hai già fatto, installa e configura gli AWS CLI SDK. AWS Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il seguente codice di esempio per eliminare un modello.

   CLI

   Modificate il valore di quanto segue

   - `project-name`al nome del progetto che contiene il modello che desiderate eliminare.
   - `dataset-type`a uno dei due `train` o`test`, a seconda del set di dati che si desidera eliminare. Se hai un singolo progetto di set di dati, specifica di `train` eliminare il set di dati.

   ```
   aws lookoutvision delete-dataset --project-name project name\
     --dataset-type dataset type \
     --profile lookoutvision-access
   ```

   Python

   Questo codice è tratto dal repository degli esempi di AWS Documentation SDK. GitHub Vedi l'esempio completo qui.

   ```
       @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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

```
/**
 * 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 });

}
```

Esportazione di set di dati da un progetto (SDK)

Puoi utilizzare l'AWSSDK per esportare set di dati da un progetto Amazon Lookout for Vision verso una posizione bucket Amazon S3.

Esportando un set di dati, puoi eseguire attività come la creazione di un progetto Lookout for Vision con una copia dei set di dati di un progetto di origine. È inoltre possibile creare un'istantanea dei set di dati utilizzati per una versione specifica di un modello.

Il codice Python in questa procedura esporta il set di dati di addestramento (immagini del manifesto e del set di dati) per un progetto in una posizione di destinazione Amazon S3 specificata. Se presente nel progetto, il codice esporta anche il manifesto del set di dati di test e le immagini del set di dati. La destinazione può trovarsi nello stesso bucket Amazon S3 del progetto di origine o in un bucket Amazon S3 diverso. Il codice utilizza l'ListDatasetEntriesoperazione per ottenere i file manifest del set di dati. Le operazioni di Amazon S3 copiano le immagini del set di dati e i file manifest aggiornati nella posizione di destinazione di Amazon S3.

Questa procedura mostra come esportare i set di dati di un progetto. Mostra anche come creare un nuovo progetto con i set di dati esportati.

Per esportare i set di dati da un progetto (SDK)

1. Se non l'hai già fatto, installa e configura gli AWS CLI SDK. AWS Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Determina il percorso Amazon S3 di destinazione per l'esportazione del set di dati. Assicurati che la destinazione si trovi in una AWSregione supportata da Amazon Lookout for Vision. Per creare un nuovo bucket Amazon S3, consulta Creazione di un bucket.
3. Assicurati che l'utente disponga delle autorizzazioni di accesso al percorso Amazon S3 di destinazione per l'esportazione del set di dati e alle posizioni S3 per i file di immagine nei set di dati del progetto di origine. Puoi utilizzare la seguente politica, che presuppone che i file di

immagini possano trovarsi in qualsiasi posizione. Sostituisci *bucket/path* con il bucket e il percorso di destinazione per l'esportazione del set di dati.

```
{
    "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": "*"
        }
    ]
}
```

Per fornire l'accesso, aggiungi autorizzazioni ai tuoi utenti, gruppi o ruoli:

- Utenti e gruppi in AWS IAM Identity Center:

  Crea un set di autorizzazioni. Segui le istruzioni riportate nella pagina Create a permission set (Creazione di un set di autorizzazioni) nella Guida per l'utente di AWS IAM Identity Center.

- Utenti gestiti in IAM tramite un provider di identità:

  Crea un ruolo per la federazione delle identità. Segui le istruzioni riportate nella pagina Creating a role for a third-party identity provider (federation) (Creazione di un ruolo per un provider di identità di terze parti [federazione]) nella Guida per l'utente di IAM.

- Utenti IAM:

- Crea un ruolo che l'utente possa assumere. Per istruzioni, consulta la pagina Creating a role for an IAM user (Creazione di un ruolo per un utente IAM) nella Guida per l'utente di IAM.
- (Non consigliato) Collega una policy direttamente a un utente o aggiungi un utente a un gruppo di utenti. Segui le istruzioni riportate nella pagina Aggiunta di autorizzazioni a un utente (console) nella Guida per l'utente di IAM.

4. Salva il codice seguente in un file denominato. `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. Eseguire il codice. Fornite i seguenti argomenti della riga di comando:

   - progetto: il nome del progetto di origine che contiene i set di dati da esportare.
   - destination: il percorso Amazon S3 di destinazione per i set di dati.

   Ad esempio, `python dataset_export.py myproject s3://bucket/path/`
6. Annota le posizioni dei file manifest visualizzati nel codice. Ne hai bisogno nel passaggio 8.
7. Crea un nuovo progetto Lookout for Vision con set di dati esportato seguendo le istruzioni disponibili all'indirizzo. Creare il tuo progetto
8. Completa una delle seguenti operazioni:
   - Usa la console Lookout for Vision per creare set di dati per il tuo nuovo progetto seguendo le istruzioni riportate all'indirizzo. Creazione di un set di dati con un file manifest (console) Non è necessario eseguire i passaggi da 1 a 6.

     Per il passaggio 12, procedi come segue:

     a. Se il progetto di origine ha un set di dati di test, scegli Set di dati di addestramento e test separati, altrimenti scegli un set di dati singolo.

     b. Per la posizione del file.manifest, inserisci la posizione del file manifest appropriato (train o test) che hai annotato nel passaggio 6.

- Utilizzate l'CreateDatasetoperazione per creare set di dati per il nuovo progetto utilizzando il codice in. Creazione di un set di dati con un file manifest (SDK) Per il `manifest_file` parametro, utilizzate la posizione del file manifesto indicata nel passaggio 6. Se il progetto di origine ha un set di dati di test, usa nuovamente il codice per creare il set di dati di test.

9. Se sei pronto, addestra il modello seguendo le istruzioni riportate all'indirizzo. Addestrare il modello

Visualizzazione dei modelli

Un progetto può avere più versioni di un modello. È possibile utilizzare la console per visualizzare i modelli di un progetto. È inoltre possibile utilizzare l'`ListModels`operazione.

> **Note**
>
> L'elenco dei modelli alla fine è coerente. Se si crea un modello, potrebbe essere necessario attendere qualche istante prima che l'elenco dei modelli sia aggiornato.

Visualizzazione dei modelli (console)

Eseguite i passaggi della procedura seguente per visualizzare i modelli del progetto nella console.

Per visualizzare i modelli (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegli Inizia.
3. Nel riquadro di navigazione a sinistra, scegli Progetti.
4. Nella pagina Progetti, seleziona il progetto che contiene i modelli che desideri visualizzare.
5. Nel riquadro di navigazione a sinistra, scegli Modelli, quindi visualizza i dettagli del modello.

Visualizzazione dei modelli (SDK)

Per visualizzare le versioni di un modello si utilizza l'`ListModels`operazione. Per ottenere informazioni su una versione specifica del modello, utilizzate l'`DescribeModel`operazione.

L'esempio seguente elenca tutte le versioni del modello in un progetto, quindi visualizza le informazioni sulle prestazioni e sulla configurazione di output per le singole versioni del modello.

Per visualizzare i modelli (SDK)

1. Se non l'hai già fatto, installa e configura gli AWS CLI AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il seguente codice di esempio per elencare i tuoi modelli e ottenere informazioni su un modello.

   CLI

   Utilizzate il `list-models` comando per elencare i modelli in un progetto.

   Modificate il seguente valore:

   - `project-name`al nome del progetto che contiene il modello che desiderate visualizzare.

   ```
   aws lookoutvision list-models --project-name project name \
     --profile lookoutvision-access
   ```

   Utilizzate il `describe-model` comando per ottenere informazioni su un modello. Modificate i seguenti valori:

   - `project-name`al nome del progetto che contiene il modello che desiderate visualizzare.
   - `model-version`alla versione del modello che desideri descrivere.

   ```
   aws lookoutvision describe-model --project-name project name\
       --model-version model version \
      --profile lookoutvision-access
   ```

   Python

   Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

   ```
       @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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

```
/**
 * 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();

}
```

Eliminazione di un modello

È possibile eliminare una versione di un modello utilizzando la console o utilizzando l'`DeleteModel`operazione. Non è possibile eliminare la versione del modello in esecuzione o in fase di addestramento.

Se la versione del modello è in esecuzione, utilizzate innanzitutto l'`StopModel`operazione per arrestare la versione del modello. Per ulteriori informazioni, consulta Interruzione del modello Amazon Lookout for Vision. Se un modello è in fase di addestramento, attendi che finisca prima di eliminare il modello.

L'eliminazione di un modello potrebbe richiedere alcuni secondi. Per determinare se un modello è stato eliminato, chiama ListProjectse controlla se la versione del modello (`ModelVersion`) è nell'`Models`array.

Eliminazione di un modello (console)

Effettuare le seguenti operazioni per eliminare un modello dalla console.

Per eliminare un modello (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegli Inizia.
3. Nel riquadro di navigazione a sinistra, scegli Progetti.

4. Nella pagina Progetti, seleziona il progetto che contiene il modello che desideri eliminare.
5. Nel riquadro di navigazione a sinistra scegliere Models (Modelli).
6. Nella vista dei modelli, selezionate il pulsante di opzione relativo al modello che desiderate eliminare.
7. Nella parte superiore della pagina, seleziona Delete (Elimina).
8. Nella finestra di dialogo Elimina, inserite delete per confermare che desiderate eliminare il modello.
9. Scegliete Elimina modello per eliminare il modello.

Eliminazione di un modello (SDK)

Utilizzare la seguente procedura per eliminare il modello con l'`DeleteModel`operazione.

Per eliminare un modello (SDK)

1. Se non l'hai già fatto, installa e configura gli AWS CLI AWS SDK. Per ulteriori informazioni, consulta Passaggio 4: Configurazione di AWS CLI e SDK AWS.
2. Usa il seguente codice di esempio per eliminare un modello.

   CLI

   Modificate i seguenti valori:

   - `project-name`al nome del progetto che contiene il modello che desiderate eliminare.
   - `model-version`alla versione del modello che si desidera eliminare.

   ```
   aws lookoutvision delete-model --project-name project name\
     --model-version model version \
     --profile lookoutvision-access
   ```

   Python

   Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

   ```
       @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

Questo codice è tratto dal GitHub repository degli esempi di AWS Documentation SDK. Vedi l'esempio completo qui.

```
/**
* 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);

}
```

Modelli di etichettatura

Puoi identificare, organizzare, cercare e filtrare i tuoi modelli Amazon Lookout for Vision utilizzando i tag. Ogni tag è un'etichetta composta da una chiave e da un valore definiti dall'utente. Ad esempio, per determinare la fatturazione dei tuoi modelli, puoi etichettare i modelli con una `Cost center` chiave e aggiungere il numero del centro di costo appropriato come valore. Per ulteriori informazioni, consulta Tagging AWS resources.

Usa i tag per:

- Tieni traccia della fatturazione per un modello utilizzando i tag di allocazione dei costi. Per ulteriori informazioni, consulta Utilizzo dei tag per l'allocazione dei costi.
- Controlla l'accesso a un modello utilizzando Identity and Access Management (IAM). Per ulteriori informazioni, consulta Controllare l'accesso alle risorse AWS utilizzando i tag delle risorse.
- Automatizza la gestione dei modelli. Ad esempio, puoi eseguire script automatici di avvio o arresto che disattivano i modelli di sviluppo durante le ore non lavorative per ridurre i costi. Per ulteriori informazioni, consulta Esecuzione del modello Amazon Lookout for Vision addestrato.

Puoi taggare i modelli utilizzando la console Amazon Lookout for Vision o utilizzando AWS gli SDK.

Argomenti

- Etichettatura dei modelli (console)

- Modelli di etichettatura (SDK)

Etichettatura dei modelli (console)

Puoi utilizzare la console Amazon Lookout for Vision per aggiungere tag ai modelli, visualizzare i tag associati a un modello e rimuovere tag.

Aggiungere o rimuovere tag (console)

Questa procedura spiega come aggiungere o rimuovere tag da un modello esistente. È inoltre possibile aggiungere tag a un nuovo modello quando viene addestrato. Per ulteriori informazioni, consulta Addestrare il modello.

Per aggiungere o rimuovere tag da un modello esistente (console)

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegli Inizia.
3. Nel riquadro di navigazione selezionare Projects (Progetti).
4. Nella pagina delle risorse dei progetti, scegli il progetto che contiene il modello a cui desideri taggare.
5. Nel riquadro di navigazione, sotto il progetto che hai scelto in precedenza, scegli Modelli.
6. Nella sezione Modelli, scegli il modello a cui vuoi aggiungere un tag.
7. Nella pagina dei dettagli del modello, scegli la scheda Tag.
8. Nella sezione Tags scegliere Manage tags (Gestisci tag).
9. Nella pagina Gestisci tag, scegli Aggiungi nuovo tag.
10. Inserisci una chiave e un valore.
    a. Per Chiave, inserisci un nome per la chiave.
    b. Per Value (Valore), immetti un valore.
11. Per aggiungere altri tag, ripeti i passaggi 9 e 10.
12. (Facoltativo) Per rimuovere un tag, scegli Rimuovi accanto al tag che desideri rimuovere. Se state rimuovendo un tag salvato in precedenza, questo viene rimosso quando salvate le modifiche.
13. Per salvare le modifiche, scegliere Salva modifiche.

Visualizzazione dei tag del modello (console)

Puoi utilizzare la console Amazon Lookout for Vision per visualizzare i tag allegati a un modello.

Per visualizzare i tag associati a tutti i modelli all'interno di un progetto, devi utilizzare l'SDK AWS. Per ulteriori informazioni, consulta Elenco dei tag del modello (SDK).

Per visualizzare i tag allegati a un modello

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegli Inizia.
3. Nel riquadro di navigazione selezionare Projects (Progetti).
4. Nella pagina delle risorse dei progetti, scegli il progetto che contiene il modello di cui desideri visualizzare il tag.
5. Nel riquadro di navigazione, sotto il progetto che hai scelto in precedenza, scegli Modelli.
6. Nella sezione Modelli, scegli il modello di cui desideri visualizzare il tag.
7. Nella pagina dei dettagli del modello, scegli la scheda Tag. I tag sono mostrati nella sezione Tag.

Modelli di etichettatura (SDK)

Puoi usare l'AWSSDK per:

- Aggiungere tag a un nuovo modello
- Aggiungi tag a un modello esistente
- Elenca i tag associati a un modello
- Rimuovere i tag da un modello

Questa sezione include AWS CLI esempi. Se non hai installato ilAWS CLI, consultaPassaggio 4: Configurazione di AWS CLI e SDK AWS.

Aggiungere tag a un nuovo modello (SDK)

È possibile aggiungere tag a un modello quando lo si crea utilizzando l'CreateModeloperazione. Specificate uno o più tag nel parametro di input dell'`Tags`array.

```
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
```

Per informazioni sulla creazione e l'addestramento di un modello, vedereAddestramento di un modello (SDK).

Aggiungere tag a un modello esistente (SDK)

Per aggiungere uno o più tag a un modello esistente, utilizzate l'TagResourceoperazione. Specificate l'Amazon Resource Name (ARN) (`ResourceArn`) del modello e i tag (`Tags`) che desiderate aggiungere.

```
aws lookoutvision tag-resource --resource-arn "resource-arn"\
  --tags '[{"Key":"Key","Value":"Value"}]' \
  --profile lookoutvision-access
```

Ad esempio codice Java, vedi TagModel.

Elenco dei tag del modello (SDK)

Per elencare i tag associati a un modello, usa l'ListTagsForResourceoperazione e specifica l'Amazon Resource Name (ARN) del modello, il (`ResourceArn`). La risposta è una mappa delle chiavi e dei valori dei tag allegati al modello specificato.

```
aws lookoutvision list-tags-for-resource --resource-arn resource-arn \
  --profile lookoutvision-access
```

Per vedere quali modelli di un progetto hanno un tag specifico, chiama `ListModels` per ottenere un elenco di modelli. Quindi chiama `ListTagsForResource` ogni modello nella risposta da`ListModels`. Controlla la risposta da `ListTagsForResource` per vedere se è presente il tag richiesto.

Ad esempio codice Java, vedi ListModelTags. Ad esempio, codice Python che cerca un valore di tag in tutti i progetti, vedi find_tag.py.

Rimozione di tag da un modello (SDK)

Per rimuovere uno o più tag da un modello, utilizzate l'UntagResourceoperazione. Specificate l'Amazon Resource Name (ARN) (`ResourceArn`) del modello e le chiavi del tag (`Tag-Keys`) che desiderate rimuovere.

```
aws lookoutvision untag-resource --resource-arn resource-arn\
  --tag-keys '["Key"]' \
  --profile lookoutvision-access
```

Ad esempio codice Java, vedi UntagModel.

Visualizzazione delle attività di rilevamento delle versioni di prova

È possibile visualizzare i rilevamenti della versione di prova utilizzando la console. Non puoi utilizzare l'AWSSDK per visualizzare le attività di rilevamento delle versioni di prova.

> **Note**
>
> L'elenco dei rilevamenti delle sperimentazioni alla fine è coerente. Se si crea un rilevamento di prova, potrebbe essere necessario attendere qualche istante prima che l'elenco dei rilevamenti dello studio sia aggiornato.

Visualizzazione delle attività di rilevamento delle versioni di prova (console)

Utilizza le seguenti procedure per visualizzare i rilevamenti della versione di prova.

Per visualizzare le attività di rilevamento delle versioni di prova

1. Apri la console Amazon Lookout for Vision all'indirizzo https://console.aws.amazon.com/lookoutvision/.
2. Scegli Inizia.
3. Nel riquadro di navigazione a sinistra, scegli Trial detections.
4. Nella pagina Rilevamenti della versione di prova, scegli un'attività di rilevamento della versione di prova per visualizzarne i dettagli.

Codice e set di dati di esempio

Di seguito sono riportati esempi di codice e set di dati che puoi utilizzare con Amazon Lookout for Vision.

Argomenti

Esempio di codice

Sono disponibili i seguenti esempi di codice per Amazon Lookout for Vision.

Esempio	Descrizione
GitHub	Esempio di codice Python che addestra e ospita un modello Amazon Lookout for Vision.
Amazon Lookout for Vision	Un taccuino Python che puoi usare per creare un modello con le immagini di esempio del circuito.
Codice di esempio in Python	Esempi di Python utilizzati nella documenta zione Amazon Lookout for Vision.
Codice di esempio Java	Esempi Java utilizzati nella documentazione di Amazon Lookout for Vision.

Set di dati di esempio

I seguenti sono set di dati di esempio che puoi utilizzare con Amazon Lookout for Vision.

Argomenti

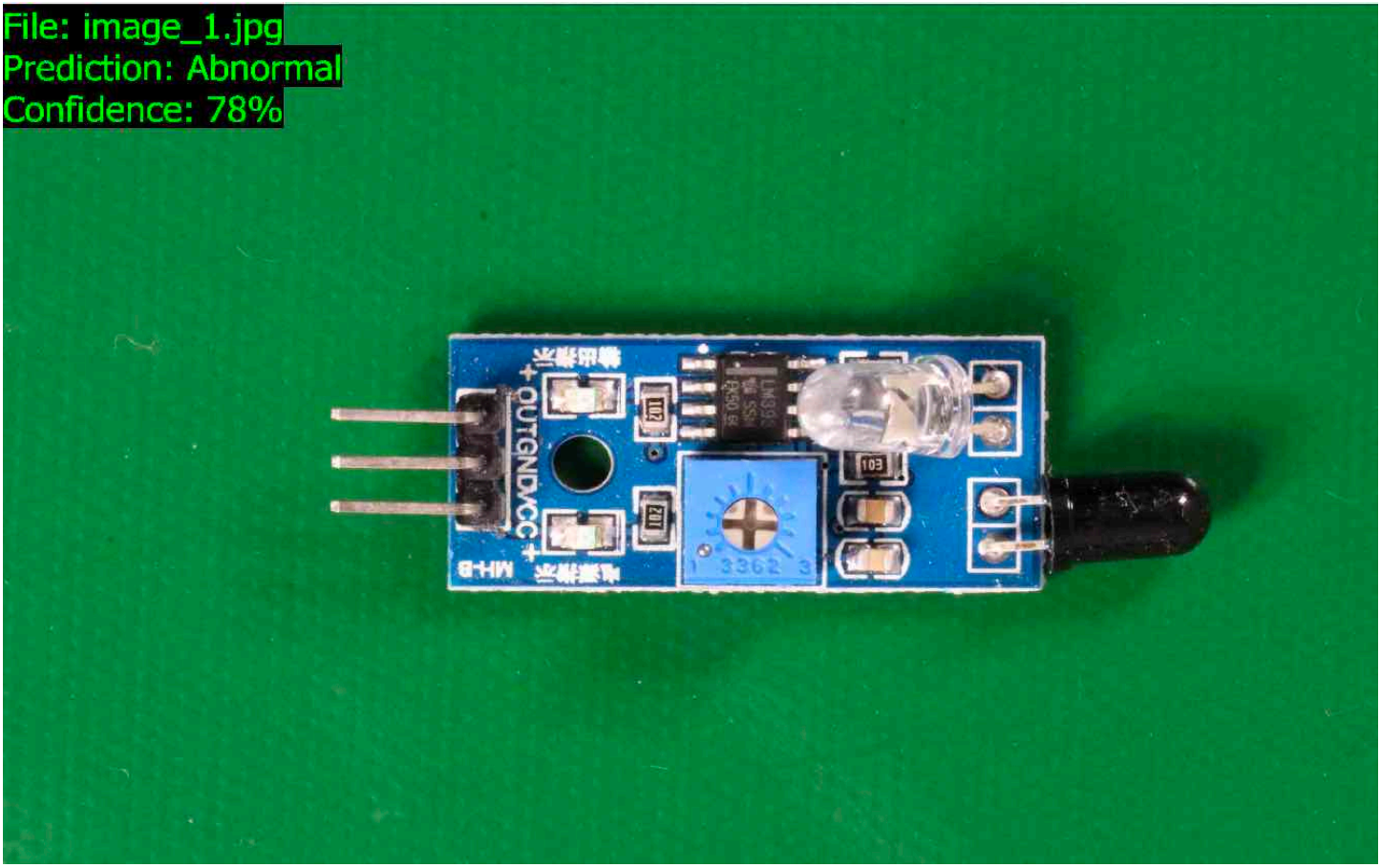
set di dati di segmentazione delle immagini

Nozioni di base su Amazon Lookout for Visionfornisce un set di dati di cookie non funzionanti che è possibile utilizzare per creare un modello di segmentazione delle immagini.

Per un altro set di dati che crea un modello di segmentazione delle immagini, consulta Identificare la posizione delle anomalie utilizzando Amazon Lookout for Vision sull'edge senza utilizzare una GPU.

set di dati di classificazione delle immagini

Amazon Lookout for Vision fornisce immagini di esempio di circuiti stampati che puoi utilizzare per creare un modello di classificazione delle immagini.



È possibile copiare le immagini dalamazon-lookout-for-vision GitHub repository https://github.com/aws-samples/. Le immagini si trovano nella`circuitboard` cartella.

La`circuitboard` cartella contiene le seguenti cartelle.

- `train`— Immagini da utilizzare in un set di dati di training.

- `test`— Immagini da utilizzare in un set di dati di test.
- `extra_images`— Immagini che puoi utilizzare per eseguire un rilevamento di prova o per provare il tuo modello addestrato DetectAnomaliesdurante l'operazione.

Letest cartelletrain e hanno ciascuna una sottocartella denominatanormal (contiene immagini normali) e una sottocartella denominataanomaly (contiene immagini con anomalie).

> **Note**
>
> Successivamente, quando crei un set di dati con la console, Amazon Lookout for Vision può utilizzare i nomi delle cartelle (normaleanomaly) per etichettare automaticamente le immagini. Per ulteriori informazioni, consulta the section called “Bucket Amazon S3”.

Per preparare le immagini del set di dati

1. Clona ilamazon-lookout-for-vision repository https://github.com/aws-samples/ sul tuo computer. Per ulteriori informazioni, consulta Clonazione di un repository.
2. Crea un bucket Amazon S3. Per ulteriori informazioni, consulta Come creare un bucket S3? .
3. Al prompt dei comandi, digita il comando seguente per copiare le immagini dei set di dati dal computer nel bucket Amazon S3.

```
aws s3 cp --recursive your-repository-folder/circuitboard s3://your-bucket/
circuitboard
```

Dopo aver caricato le immagini, puoi creare un modello. Puoi classificare automaticamente le immagini aggiungendo le immagini dalla posizione Amazon S3 in cui hai precedentemente caricato le immagini della scheda di circuito. Ricorda che ti viene addebitato un costo per ogni addestramento riuscito di un modello e per la quantità di tempo in cui il modello è in esecuzione (ospitato).

Per creare un modello di classificazione

1. FareCreazione di un progetto (console).
2. FareCreazione di un set di dati utilizzando immagini archiviate in un bucket Amazon S3.
   - Per il passaggio 6, scegli la scheda Set di dati di addestramento e test separati.

- Per il passaggio 8a, inserisci l'URI S3 per le immagini di addestramento che hai caricato in Per preparare le immagini del set di dati. Ad esempio, `s3://your-bucket/circuitboard/train`. Per il passaggio 8b, inserisci l'URI S3 per il set di dati di test. Ad esempio, `s3://your-bucket/circuitboard/test`.
- Assicurati di eseguire il passaggio 9.

3. FareAddestramento di un modello (console).
4. FareAvvio del modello (console).
5. FareRilevamento di anomalie in un'immagine. È possibile utilizzare immagini dalla`test_images` cartella.
6. Al termine del modello, fateloArresto del modello (console).

Sicurezza in Amazon Lookout for Vision

Per AWS, la sicurezza del cloud ha la massima priorità. In quanto cliente AWS, puoi trarre vantaggio da un'architettura di data center e di rete progettata per soddisfare i requisiti delle aziende più esigenti a livello di sicurezza.

La sicurezza è una responsabilità condivisa tra AWS e l'utente. Il modello di responsabilità condivisa descrive questo modello come sicurezza del cloud e sicurezza nel cloud:

- La sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che esegue AWS i servizi nel cloud AWS. AWS fornisce, inoltre, servizi utilizzabili in modo sicuro. I revisori di terze parti testano regolarmente e verificano l'efficacia della nostra sicurezza nell'ambito dei Programmi di conformità AWS. https://aws.amazon.com/compliance/services-in-scope/
- Sicurezza nel cloud: la tua responsabilità è determinata dal servizio AWS che utilizzi. L'utente è anche responsabile per altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda, le leggi e le normative applicabili.

Questa documentazione consente di comprendere come comprendere come applicare il modello di responsabilità condivisa quando si usa usa Lookout for Vision Gli argomenti illustrano come configurare Lookout for Vision È inoltre illustrato come utilizzare altri servizi AWS che ti aiutano a monitorare e proteggere le risorse Lookout for Vision

Argomenti

Protezione dei dati in Amazon Lookout for Vision

Il modello di responsabilità AWS condivisa Modello si applica alla protezione dei dati in Amazon Lookout for Vision. Come descritto in questo modello, AWSè responsabile della protezione dell'infrastruttura globale che esegue tutto l'Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. Inoltre, sei responsabile della configurazione della

protezione e delle attività di gestione per i Servizi AWS che utilizzi. Per ulteriori informazioni sulla privacy dei dati, vedi le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al Modello di responsabilità condivisa AWS e GDPR nel Blog sulla sicurezza AWS.

Per garantire la protezione dei dati, ti suggeriamo di proteggere le credenziali Account AWS e di configurare singoli utenti con AWS IAM Identity Center o AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Utilizza SSL/TLS per comunicare con le risorse AWS. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail.
- Utilizza le soluzioni di crittografia AWS, insieme a tutti i controlli di sicurezza predefiniti in Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se necessiti di moduli crittografici convalidati FIPS 140-2 quando accedi ad AWSattraverso un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Ti consigliamo vivamente di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Ciò include quando lavori con Lookout for Vision o Servizi AWS altro utilizzando la console, l'API AWS o gli AWS CLI SDK. I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

Crittografia dei dati

Le seguenti informazioni spiegano dove Amazon Lookout for Vision utilizza la crittografia dei dati per proteggere i dati.

Crittografia a riposo

Immagini

Per addestrare il tuo modello, Amazon Lookout for Vision crea una copia delle immagini di formazione e test di origine. Le immagini copiate vengono crittografate a riposo in Amazon Simple Storage Service (S3) utilizzando la crittografia lato server con una chiave o una Chiave di proprietà di AWS chiave fornita da te. Le chiavi vengono archiviate utilizzando AWS Key Management Service (SSE-KMS). Le tue immagini di origine non vengono modificate. Per ulteriori informazioni, consulta Addestrare il modello.

Modelli Amazon Lookout for Vision

Per impostazione predefinita, i modelli addestrati e i file manifest vengono crittografati in Amazon S3 utilizzando la crittografia lato server con chiavi KMS archiviate in AWS Key Management Service (SSE-KMS). Lookout for Vision utilizza Chiave di proprietà di AWS un. Per ulteriori informazioni, consulta Protezione dei dati con la crittografia lato server. I risultati della formazione vengono scritti nel bucket specificato nel parametro di `output_bucket` input to. `CreateModel` I risultati dell'addestramento vengono crittografati utilizzando le impostazioni di crittografia configurate per il bucket (`output_bucket`).

Bucket per console Amazon Lookout for Vision

La console Amazon Lookout for Vision crea un bucket Amazon S3 (bucket console) che puoi utilizzare per gestire i tuoi progetti. Il bucket della console è crittografato utilizzando la crittografia Amazon S3 predefinita. Per ulteriori informazioni, consulta Crittografia predefinita di Amazon Simple Storage Service per i bucket S3. Se utilizzi la tua chiave KMS, configura il bucket della console dopo averlo creato. Per ulteriori informazioni, consulta Protezione dei dati con la crittografia lato server. Amazon Lookout for Vision blocca l'accesso pubblico al bucket della console.

Crittografia in transito

Gli endpoint dell'API Amazon Lookout for Vision supportano solo connessioni sicure tramite HTTPS. Tutte le comunicazioni sono crittografate con Transport Layer Security (TLS).

Gestione delle chiavi

Puoi utilizzare AWS Key Management Service (KMS) per gestire la crittografia delle immagini di input archiviate nei bucket Amazon S3. Per ulteriori informazioni, consulta Fase 5: (Facoltativo) Utilizzo della propria chiave AWS Key Management Service.

Per impostazione predefinita, le immagini sono crittografate con una chiave di proprietà e gestione di AWS. Puoi anche scegliere di utilizzare la tua chiave AWS Key Management Service (KMS) personale. Per ulteriori informazioni, consulta Concetti di AWS Key Management Service.

Riservatezza del traffico Internet

Un endpoint Amazon Virtual Private Cloud (Amazon VPC) per Amazon Lookout for Vision è un'entità logica all'interno di un VPC che consente la connettività solo ad Amazon Lookout for Vision. Amazon VPC indirizza le richieste ad Amazon Lookout for Vision e reindirizza le risposte al VPC. Per ulteriori informazioni, consulta Endpoint VPC nella Guida per l'utente di Amazon VPC. Per informazioni sull'utilizzo degli endpoint Amazon VPC con Amazon Lookout for Vision, consulta. Accedi ad Amazon Lookout for Vision utilizzando un endpoint di interfaccia (AWS PrivateLink)

Gestione delle identità e degli accessi per Amazon Lookout for Vision

AWS Identity and Access Management (IAM) è un Servizio AWSche consente agli amministratori di controllare in modo sicuro l'accesso alle risorse AWS. Gli amministratori IAM controllano chi può essere autenticato (effettuato l'accesso) e autorizzato (dispone delle autorizzazioni) a utilizzare le risorse Lookout for Vision. IAM è un Servizio AWS il cui uso non comporta costi aggiuntivi.

Argomenti

- Destinatari
- Autenticazione con identità
- Gestione dell'accesso con policy
- In che modo Amazon Lookout for Vision funziona con IAM
- Esempi di policy basate sull'identità di Amazon Lookout for Vision
- AWSpolitiche gestite per Amazon Lookout for Vision
- Risoluzione dei problemi relativi all'identità e all'accesso ad Amazon Lookout for Vision

Destinatari

Il modo in cui utilizzi AWS Identity and Access Management (IAM) varia a seconda del lavoro svolto in Lookout for Vision.

Utente del servizio: se utilizzi il servizio Lookout for Vision per svolgere il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. Man mano che utilizzi più funzionalità di Lookout for Vision per svolgere il tuo lavoro, potresti aver bisogno di autorizzazioni aggiuntive. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità di Lookout for Vision, Risoluzione dei problemi relativi all'identità e all'accesso ad Amazon Lookout for Vision consulta.

Amministratore del servizio: se sei responsabile delle risorse di Lookout for Vision presso la tua azienda, probabilmente hai pieno accesso a Lookout for Vision. È tuo compito determinare a quali funzionalità e risorse di Lookout for Vision devono accedere gli utenti del servizio. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM con Lookout for Vision, In che modo Amazon Lookout for Vision funziona con IAM consulta.

Amministratore IAM: se sei un amministratore IAM, potresti voler conoscere i dettagli su come scrivere policy per gestire l'accesso a Lookout for Vision. Per visualizzare esempi di policy basate sull'identità di Lookout for Vision che puoi utilizzare in IAM, consulta. Esempi di policy basate sull'identità di Amazon Lookout for Vision

Autenticazione con identità

L'autenticazione è la procedura di accesso ad AWScon le credenziali di identità. Devi essere autenticato (connesso a AWS) come utente root Utente root dell'account AWS, come utente IAM o assumere un ruolo IAM.

Puoi accedere ad AWScome identità federata utilizzando le credenziali fornite attraverso un'origine di identità. AWS IAM Identity Center Gli esempi di identità federate comprendono gli utenti del centro identità IAM, l'autenticazione Single Sign-On (SSO) dell'azienda e le credenziali di Google o Facebook. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Se accedi ad AWStramite la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere alla AWS Management Consoleo al portale di accesso AWS. Per ulteriori informazioni sull'accesso ad AWS, consulta la sezione Come accedere al tuo Account AWSnella Guida per l'utente di Accedi ad AWS.

Se accedi ad AWS in modo programmatico, AWS fornisce un Software Development Kit (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le richieste utilizzando le tue

credenziali. Se non utilizzi gli strumenti AWS, devi firmare le richieste personalmente. Per ulteriori informazioni sulla firma delle richieste, consulta Firma delle richieste AWS nella Guida per l'utente IAM.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. AWS consiglia ad esempio di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza dell'account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA) in AWS nella Guida per l'utente di IAM.

Utente root di un Account AWS

Quando crei un Account AWS, inizi con una singola identità di accesso che ha accesso completo a tutti i Servizi AWSe le risorse nell'account. Tale identità è detta utente root Account AWSed è possibile accedervi con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conservare le credenziali dell'utente root e utilizzarle per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione Attività che richiedono le credenziali dell'utente root nella Guida per l'utente di IAM.

Identità federata

Come best practice, richiedere agli utenti umani, compresi quelli che richiedono l'accesso di amministratore, di utilizzare la federazione con un provider di identità per accedere a Servizi AWSutilizzando credenziali temporanee.

Un'identità federata è un utente della directory degli utenti aziendali, un provider di identità Web, AWS Directory Service, la directory Identity Center o qualsiasi utente che accede a Servizi AWS utilizzando le credenziali fornite tramite un'origine di identità. Quando le identità federate accedono a Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. È possibile creare utenti e gruppi in IAM Identity Center oppure connettersi e sincronizzarsi con un gruppo di utenti e gruppi nell'origine di identità per utilizzarli in tutte le applicazioni e gli Account AWS. Per ulteriori informazioni su IAM Identity Center, consulta Cos'è IAM Identity Center? nella Guida per l'utente di AWS IAM Identity Center.

Utenti e gruppi IAM

Un utente IAM è una identità all'interno del tuo Account AWSche dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, se si hanno casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente di IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato Amministratori IAM e concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Quando creare un utente IAM (invece di un ruolo) nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità all'interno di Account AWSche dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. È possibile assumere temporaneamente un ruolo IAM nella AWS Management Consolemediante lo scambio di ruoli. È possibile assumere un ruolo chiamando un'operazione AWS CLIo API AWSoppure utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente di IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Creazione di un ruolo per un provider di identità di terza parte nella Guida per l'utente di IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un

ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per ulteriori informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center.

- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, per alcuni dei Servizi AWS, è possibile collegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.
- Accesso multi-servizio: alcuni Servizi AWS utilizzano funzionalità in altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
  - Inoltro delle sessioni di accesso (FAS): quando si utilizza un utente o un ruolo IAM per eseguire operazioni in AWS, tale utente o ruolo viene considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con il Servizio AWS richiedente, per effettuare richieste a servizi a valle. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che necessita di interazioni con altri Servizi AWS o risorse per essere portata a termine. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le operazioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.
  - Ruolo di servizio: un ruolo di servizio è un ruolo IAM assunto da un servizio per eseguire operazioni per conto dell'utente. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.
  - Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati ai servizi sono visualizzati nell'account Account AWSe sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

- Applicazioni in esecuzione su Amazon EC2: è possibile utilizzare un ruolo IAM per gestire credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 che eseguono richieste di AWS CLIo dell'API AWS. Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un ruolo AWSa un'istanza EC2, affinché sia disponibile per tutte le relative applicazioni, puoi creare un profilo dell'istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2 nella Guida per l'utente di IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta Quando creare un ruolo IAM (invece di un utente) nella Guida per l'utente di IAM.

Gestione dell'accesso con policy

Per controllare l'accesso a AWS è possibile creare policy e collegarle a identità o risorse AWS. Una policy è un oggetto in AWSche, quando associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWSvaluta queste policy quando un principale IAM (utente, utente root o sessione ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle policy viene archiviata in AWSsotto forma di documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWSJSON per specificare l'accesso ai diversi elementi. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. Successivamente l'amministratore può aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dalla AWS Management Console, la AWS CLIo l'API AWS.

Policy basate su identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono incorporate direttamente in un singolo utente, gruppo o ruolo. Le policy gestite sono policy autonome che possono essere collegate a più utenti, gruppi e ruoli in Account AWS. Le policy gestite includono le policy gestite da AWSe le policy gestite dal cliente. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente di IAM.

Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile allegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarle per controllare l'accesso a una risorsa specifica. Quando è allegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o Servizi AWS.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy gestite da AWSda IAM in una policy basata su risorse.

Liste di controllo degli accessi (ACL)

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3, AWS WAFe Amazon VPC sono esempi di servizi che supportano le ACL. Per maggiori informazioni sulle ACL, consulta Panoramica delle liste di controllo degli accessi (ACL) nella Guida per gli sviluppatori di Amazon Simple Storage Service.

Altri tipi di policy

AWS supporta altri tipi di policy meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzione avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.
- Policy di controllo dei servizi (SCP): le SCP sono policy JSON che specificano il numero massimo di autorizzazioni per un'organizzazione o unità organizzativa (OU) in AWS Organizations. AWS Organizationsè un servizio per il raggruppamento e la gestione centralizzata degli Account AWSmultipli di proprietà dell'azienda. Se abiliti tutte le funzionalità in un'organizzazione, puoi applicare le policy di controllo dei servizi (SCP) a uno o tutti i tuoi account. La SCP limita le autorizzazioni per le entità negli account membri, compreso ogni Utente root dell'account AWS. Per ulteriori informazioni su organizzazioni e policy SCP, consulta la pagina sulle Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations.
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente di IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per informazioni su come AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella Guida per l'utente di IAM.

In che modo Amazon Lookout for Vision funziona con IAM

Prima di utilizzare IAM per gestire l'accesso a Lookout for Vision, scopri quali funzionalità IAM sono disponibili per l'uso con Lookout for Vision.

Funzionalità IAM che puoi utilizzare con Amazon Lookout for Vision

Funzionalità IAM	Supporto per Lookout for Vision
Policy basate su identità	Sì
Policy basate su risorse	No
Operazioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione della policy (specifica del servizio)	Sì
Liste di controllo degli accessi	No
ABAC (tag nelle policy)	Parziale
Credenziali temporanee	Sì
Sessioni di accesso diretto (FAS)	Sì
Ruoli di servizio	No
Ruoli collegati al servizio	No

Per avere una visione di alto livello di come Lookout for Vision e AWS altri servizi funzionano con la maggior parte delle funzionalità IAM, AWSconsulta i servizi che funzionano con IAM nella IAM User Guide.

Politiche basate sull'identità per Lookout for Vision

Supporta le policy basate su identità	Sì

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Con le policy basate su identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Non è possibile specificare l'entità principale in una policy basata sull'identità perché si applica all'utente o al ruolo a cui è associato. Per informazioni su tutti gli elementi utilizzabili in una policy JSON, consulta Guida di riferimento agli elementi delle policy JSON IAM nella Guida per l'utente di IAM.

Esempi di policy basate sull'identità per Lookout for Vision

Per visualizzare esempi di policy basate sull'identità di Lookout for Vision, vedere. Esempi di policy basate sull'identità di Amazon Lookout for Vision

Politiche basate sulle risorse all'interno di Lookout for Vision

Supporta le policy basate su risorse	No

Le policy basate su risorse sono documenti di policy JSON che è possibile allegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarle per controllare l'accesso a una risorsa specifica. Quando è allegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o Servizi AWS.

Per consentire l'accesso multi-account, puoi specificare un intero account o entità IAM in un altro account come principale in una policy basata sulle risorse. L'aggiunta di un principale multi-account a una policy basata sulle risorse rappresenta solo una parte della relazione di trust. Quando l'entità principale e la risorsa si trovano in diversi Account AWS, un amministratore IAM nell'account attendibile deve concedere all'entità principale (utente o ruolo) anche l'autorizzazione per accedere alla risorsa. L'autorizzazione viene concessa collegando all'entità una policy basata sull'identità. Tuttavia, se una policy basata su risorse concede l'accesso a un principale nello stesso account, non

sono richieste ulteriori policy basate su identità. Per ulteriori informazioni, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.

Azioni politiche per Lookout for Vision

Supporta le operazioni di policy	Sì

Gli amministratori possono utilizzare le policy JSON AWSper specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso a un criterio. Le operazioni di policy hanno spesso lo stesso nome dell'operazione API AWS. Ci sono alcune eccezioni, ad esempio le azioni di sola autorizzazione che non hanno un'operazione API corrispondente. Esistono anche alcune operazioni che richiedono più operazioni in una policy. Queste operazioni aggiuntive sono denominate operazioni dipendenti.

Includi le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Per visualizzare un elenco di azioni Lookout for Vision, consulta Azioni definite da Amazon Lookout for Vision nel Service Authorization Reference.

Le azioni politiche in Lookout for Vision utilizzano il seguente prefisso prima dell'azione:

```
lookoutvision
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [
      "lookoutvision:action1",
      "lookoutvision:action2"
         ]
```

Per visualizzare esempi di policy basate sull'identità di Lookout for Vision, vedere. Esempi di policy basate sull'identità di Amazon Lookout for Vision

Risorse politiche per Lookout for Vision

Supporta le risorse di policy	Sì

Gli amministratori possono utilizzare le policy JSON AWSper specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento JSON `Resource`della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Le istruzioni devono includere un elemento `Resource`o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il suo nome della risorsa Amazon (ARN). Puoi eseguire questa operazione per azioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le azioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Per visualizzare un elenco dei tipi di risorse Lookout for Vision e dei relativi ARN, consulta Risorse definite da Amazon Lookout for Vision nel Service Authorization Reference. Per sapere con quali azioni puoi specificare l'ARN di ogni risorsa, consulta Azioni definite da Amazon Lookout for Vision.

Per visualizzare esempi di policy basate sull'identità di Lookout for Vision, vedere. Esempi di policy basate sull'identità di Amazon Lookout for Vision

Chiavi relative alle condizioni delle policy per Lookout for Vision

Supporta le chiavi di condizione delle policy specifiche del servizio	Sì

Gli amministratori possono utilizzare le policy JSON AWSper specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Condition` (o blocco `Condition`) consente di specificare le condizioni in cui un'istruzione è in vigore. L'elemento `Condition` è facoltativo. Puoi compilare espressioni

condizionali che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta.

Se specifichi più elementi `Condition` in un'istruzione o più chiavi in un singolo elemento `Condition`, questi vengono valutati da AWS utilizzando un'operazione `AND` logica. Se specifichi più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione `OR` logica. Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

Puoi anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, puoi autorizzare un utente IAM ad accedere a una risorsa solo se è stata taggata con il relativo nome utente IAM. Per ulteriori informazioni, consulta Elementi delle policy IAM: variabili e tag nella Guida per l'utente di IAM.

AWS supporta chiavi di condizione globali e chiavi di condizione specifiche per il servizio. Per visualizzare tutte le chiavi di condizione globali di AWS, consulta Chiavi di contesto delle condizioni globali di AWS nella Guida per l'utente di IAM.

Per visualizzare un elenco dei codici di condizione di Lookout for Vision, consulta Chiavi di condizione per Amazon Lookout for Vision nel Service Authorization Reference. Per sapere con quali azioni e risorse puoi utilizzare una chiave di condizione, consulta Azioni definite da Amazon Lookout for Vision.

Per visualizzare esempi di policy basate sull'identità di Lookout for Vision, vedere. Esempi di policy basate sull'identità di Amazon Lookout for Vision

ACL in Lookout for Vision

Supporta le ACL	No

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni ad accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino il formato del documento di policy JSON.

ABAC con Lookout for Vision

Supporta ABAC (tag nelle policy)	Parziale

Il controllo dell'accesso basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi. In AWS, tali attributi sono denominati tag. È possibile collegare dei tag alle entità IAM (utenti o ruoli) e a numerose risorse AWS. L'assegnazione di tag alle entità e alle risorse è il primo passaggio di ABAC. In seguito, vengono progettate policy ABAC per consentire operazioni quando il tag dell'entità principale corrisponde al tag sulla risorsa a cui si sta provando ad accedere.

La strategia ABAC è utile in ambienti soggetti a una rapida crescita e aiuta in situazioni in cui la gestione delle policy diventa impegnativa.

Per controllare l'accesso basato su tag, fornisci informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Yes (Sì). Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per ulteriori informazioni su ABAC, consulta Che cos'è ABAC? nella Guida per l'utente di IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta Utilizzo del controllo degli accessi basato su attributi (ABAC) nella Guida per l'utente di IAM.

Utilizzo di credenziali temporanee con Lookout for Vision

Supporta le credenziali temporanee	Sì

Alcuni Servizi AWSnon funzionano quando si accede utilizzando credenziali temporanee. Per ulteriori informazioni, inclusi i Servizi AWS che funzionano con le credenziali temporanee, consulta Servizi AWS supportati da IAM nella Guida per l'utente IAM.

Le credenziali temporanee sono utilizzate se si accede alla AWS Management Consoleutilizzando qualsiasi metodo che non sia la combinazione di nome utente e password. Ad esempio, quando accedi ad AWS utilizzando il collegamento Single Sign-On (SSO) della tua azienda, tale processo crea in automatico credenziali temporanee. Le credenziali temporanee vengono create in automatico anche quando accedi alla console come utente e poi cambi ruolo. Per ulteriori informazioni sullo scambio dei ruoli, consulta Cambio di un ruolo (console) nella Guida per l'utente di IAM.

È possibile creare manualmente credenziali temporanee utilizzando la AWS CLIo l'API AWS. È quindi possibile utilizzare tali credenziali temporanee per accedere ad AWS. AWSconsiglia di generare le

credenziali temporanee dinamicamente anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta Credenziali di sicurezza provvisorie in IAM.

Sessioni di accesso diretto per Lookout for Vision

Supporta sessioni di accesso diretto (FAS)	Sì

Quando si utilizza un utente o un ruolo IAM per eseguire operazioni in AWS, si viene considerati un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'azione che attiva un'altra azione in un servizio diverso. FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con il Servizio AWS richiedente, per effettuare richieste a servizi a valle. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che necessita di interazioni con altri Servizi AWS o risorse per essere portata a termine. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le operazioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.

Ruoli di servizio per Lookout for Vision

Supporta i ruoli di servizio	No

Un ruolo di servizio è un ruolo IAM che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.

> **Warning**
>
> La modifica delle autorizzazioni per un ruolo di servizio potrebbe interrompere la funzionalità Lookout for Vision. Modifica i ruoli di servizio solo quando Lookout for Vision fornisce indicazioni in tal senso.

Ruoli collegati ai servizi per Lookout for Vision

Supporta i ruoli collegati ai servizi	No

Un ruolo collegato ai servizi è un tipo di ruolo di servizio che è collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'operazione per tuo conto. I ruoli collegati ai servizi sono visualizzati nell'account Account AWSe sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

Per ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta Servizi AWS supportati da IAM. Trova un servizio nella tabella che include un `Yes` nella colonna Service-linked role (Ruolo collegato ai servizi). Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Esempi di policy basate sull'identità di Amazon Lookout for Vision

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare le risorse Lookout for Vision. Inoltre, non sono in grado di eseguire attività utilizzando la AWS Management Console, l'AWS Command Line Interface (AWS CLI) o l'API AWS. Per concedere agli utenti l'autorizzazione per eseguire operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Per dettagli sulle azioni e sui tipi di risorse definiti da Lookout for Vision, incluso il formato degli ARN per ciascun tipo di risorsa, consulta Azioni, risorse e chiavi di condizione per Amazon Lookout for Vision nel Service Authorization Reference.

Argomenti

- Best practice per le policy
- Accesso a un singolo progetto Amazon Lookout for Vision
- Esempi di policy basate su tag

Best practice per le policy

I criteri basati sull'identità determinano se qualcuno può creare, accedere o eliminare le risorse Lookout for Vision nel tuo account. Queste operazioni possono comportare costi aggiuntivi per l'Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e raccomandazioni:

- Nozioni di base sulle policy gestite da AWSe passaggio alle autorizzazioni con privilegio minimo: per le informazioni di base su come concedere autorizzazioni a utenti e carichi di lavoro, utilizza le policy gestite da AWSche concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo Account AWS. Ti consigliamo pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente di AWSspecifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni dei processi nella Guida per l'utente IAM.
- Applica le autorizzazioni con privilegi minimi: quando imposti le autorizzazioni con le policy IAM, concedi solo le autorizzazioni richieste per eseguire un'attività. Puoi farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegi minimi. Per ulteriori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso: per limitare l'accesso a operazioni e risorse puoi aggiungere una condizione alle tue policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi inoltre utilizzare le condizioni per concedere l'accesso alle operazioni di servizio, ma solo se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per ulteriori informazioni, consulta la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente di IAM.
- Utilizzo di IAM Access Analyzer per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio della policy IAM (JSON) e alle best practice di IAM. IAM Access Analyzer offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per ulteriori informazioni, consulta Convalida delle policy per IAM Access Analyzer nella Guida per l'utente di IAM.
- Richiesta dell'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o utenti root nel tuo Account AWS, attiva MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungi le condizioni MFA alle policy. Per ulteriori informazioni, consulta Configurazione dell'accesso alle API protetto con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Accesso a un singolo progetto Amazon Lookout for Vision

In questo esempio, vuoi concedere a un utente del tuo AWS account l'accesso a uno dei tuoi progetti Amazon Lookout for Vision.

```
{
    "Sid": "SpecificProjectOnly",
    "Effect": "Allow",
    "Action": [
        "lookoutvision:DetectAnomalies"
    ],
    "Resource": "arn:aws:lookoutvision:us-east-1:123456789012:model/myproject/*"
}
```

Esempi di policy basate su tag

Le politiche basate su tag sono documenti di policy JSON che specificano le azioni che un principale può eseguire sulle risorse con tag.

Usa un tag per accedere a una risorsa

Questa policy di esempio concede a un utente o a un ruolo nel tuo account AWS l'autorizzazione a utilizzare l'`DetectAnomalies`operazione con qualsiasi modello etichettato con la chiave `stage` e il valore`production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "LookoutVision:DetectAnomalies"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "production"
                }
            }
        }
   ]
}
```

Usa un tag per negare l'accesso a specifiche operazioni di Amazon Lookout for Vision

Questa policy di esempio nega l'autorizzazione a un utente o a un ruolo nel tuo account AWS di chiamare le `StopModel` operazioni `DeleteModel` or con qualsiasi modello contrassegnato con la chiave `stage` e il valore`production`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "LookoutVision:DeleteModel",
                "LookoutVision:StopModel"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/stage": "production"
                }
            }
        }
   ]
}
```

AWSpolitiche gestite per Amazon Lookout for Vision

Una policy gestita da AWS è una policy autonoma creata e amministrata da AWS. Le policy gestite da AWS sono progettate per fornire autorizzazioni per molti casi d'uso comuni in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Ricorda che le policy gestite da AWS potrebbero non concedere autorizzazioni con privilegi minimi per i tuoi casi d'uso specifici perché possono essere utilizzate da tutti i clienti AWS. Consigliamo pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i tuoi casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle policy gestite da AWS. Se AWS aggiorna le autorizzazioni definite in una policy gestita da AWS, l'aggiornamento riguarda tutte le identità principali (utenti, gruppi e ruoli) a cui è collegata la policy. È molto probabile che AWS aggiorni

una policy gestita da AWS quando viene lanciato un nuovo Servizio AWS o nuove operazioni API diventano disponibili per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWS nella Guida per l'utente di IAM.

Policy gestita da AWS: AmazonLookoutVisionReadOnlyAccess

Usa il`AmazonLookoutVisionReadOnlyAccess`politica per consentire agli utenti l'accesso in sola lettura ad Amazon Lookout for Vision (e alle sue dipendenze) con le seguenti azioni di Amazon Lookout for Vision (operazioni SDK). Ad esempio, puoi usare`DescribeModel`per ottenere informazioni su un modello esistente.

- DescribeDataset
- DescribeModel
- DescribeModelPackagingJob
- DescribeProject
- ListDatasetEntries
- ListModelPackagingJobs
- ListModels
- ListProjects
- ListTagsForResource

Per richiamare azioni di sola lettura, gli utenti non hanno bisogno delle autorizzazioni del bucket Amazon S3. Tuttavia, le risposte operative potrebbero includere riferimenti ai bucket Amazon S3. Ad esempio,`source-ref`inserisci la risposta da`ListDatasetEntries`è un riferimento a un'immagine in un bucket Amazon S3. Aggiungi le autorizzazioni dei bucket Amazon S3 se i tuoi utenti devono accedere ai bucket di riferimento. Ad esempio, un utente potrebbe voler scaricare un'immagine a cui fa riferimento un`source-ref`campo. Per ulteriori informazioni, consulta Concessione delle autorizzazioni per Amazon S3 Bucket.

È possibile allegare la policy `AmazonLookoutVisionReadOnlyAccess` alle identità IAM.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

```
{
    "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": "*"
        }
    ]
}
```

Policy gestita da AWS:AmazonLookoutVisionFullAccess

Usa il`AmazonLookoutVisionFullAccess`politica per consentire agli utenti l'accesso completo ad Amazon Lookout for Vision (e alle sue dipendenze) con le azioni di Amazon Lookout for Vision (operazioni SDK). Ad esempio, puoi addestrare un modello senza dover utilizzare la console Amazon Lookout for Vision. Per ulteriori informazioni, consulta Operazioni.

Per creare un set di dati (`CreateDataset`) o crea un modello (`CreateModel`), i tuoi utenti devono disporre delle autorizzazioni di accesso complete al bucket Amazon S3 che archivia le immagini dei set di dati, AmazonSageMakerFile del manifesto di Ground Truth e risultati della formazione. Per ulteriori informazioni, consulta Fase 2: Configurare le autorizzazioni.

Puoi anche concedere l'autorizzazione alle azioni SDK di Amazon Lookout for Vision utilizzando`AmazonLookoutVisionConsoleFullAccess`politica.

È possibile allegare la policy `AmazonLookoutVisionFullAccess` alle identità IAM.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LookoutVisionFullAccess",
            "Effect": "Allow",
            "Action": [
                "lookoutvision:*"
            ],
            "Resource": "*"
        }
    ]
}
```

Policy gestita da AWS: AmazonLookoutVisionConsoleFullAccess

Usa il`AmazonLookoutVisionFullAccess`politica per consentire agli utenti l'accesso completo alla console Amazon Lookout for Vision, alle azioni (operazioni SDK) e a qualsiasi dipendenza del servizio. Per ulteriori informazioni, consulta Nozioni di base su Amazon Lookout for Vision.

La`LookoutVisionConsoleFullAccess`la politica include le autorizzazioni per il bucket della tua console Amazon Lookout for Vision. Per informazioni sul bucket della console, vediPassaggio 3: Creare il bucket della console. Per archiviare set di dati, immagini e AmazonSageMakerGround Truth manifesta i file in un bucket Amazon S3 diverso, i tuoi utenti necessitano di autorizzazioni aggiuntive. Per ulteriori informazioni, consulta the section called “Impostazione delle autorizzazioni per i bucket Amazon S3”.

È possibile allegare la policy `AmazonLookoutVisionConsoleFullAccess` alle identità IAM.

Raggruppamenti di autorizzazioni

Questa politica è raggruppata in dichiarazioni basate sul set di autorizzazioni fornite:

- `LookoutVisionFullAccess`— Consente l'accesso per eseguire tutte le azioni di Lookout for Vision.
- `LookoutVisionConsoleS3BucketSearchAccess`— Consente di elencare tutti i bucket Amazon S3 di proprietà del chiamante. Lookout for Vision utilizza questa azione per identificare il bucket della console Lookout for Vision specifico per regione AWS, se presente nell'account del chiamante.
- `LookoutVisionConsoleS3BucketFirstUseSetupAccessPermissions`— Consente di creare e configurare bucket Amazon S3 che corrispondono allo schema dei nomi dei bucket della console Lookout for Vision. Lookout for Vision utilizza queste azioni per creare e configurare un bucket della console Lookout for Vision specifico per regione quando non riesce a trovarne uno.
- `LookoutVisionConsoleS3BucketAccess`— Consente azioni Amazon S3 dipendenti su bucket che corrispondono allo schema dei nomi dei bucket della console Lookout for Vision. Usi di Lookout for Visions`3:ListBucket`per cercare oggetti di immagine durante la creazione di un set di dati da un bucket Amazon S3 e quando si avvia un'attività di rilevamento di prova. Usi di Lookout for Visions`3:GetBucketLocation es3:GetBucketVersioning`per convalidare i bucketAWSRegione, proprietario e configurazione come parte di quanto segue:
  - Creazione di un set di dati
  - Addestrare un modello
  - Avvio di un'attività di rilevamento di prova
  - Esecuzione del feedback sul rilevamento delle prove

  `LookoutVisionConsoleS3ObjectAccess`— Consente la lettura e la scrittura di oggetti Amazon S3 all'interno di bucket che corrispondono allo schema dei nomi dei bucket della console Lookout for Vision. Lookout for Vision utilizza queste azioni per visualizzare immagini nelle visualizzazioni della galleria della console e caricare nuove immagini da utilizzare nei set di dati. Inoltre, queste autorizzazioni consentono a Lookout for Vision di scrivere metadati durante la creazione di un set di dati, l'addestramento di un modello, l'avvio di un'attività di rilevamento della prova e l'esecuzione di feedback sul rilevamento delle prove.
- `LookoutVisionConsoleDatasetLabelingToolsAccess`— Consente Amazon dipendenteSageMaker GroundTruthazioni di etichettatura. Lookout for Vision utilizza queste azioni per scansionare i bucket S3 alla ricerca di immagini, creareGroundTruthfile manifest e annotare i risultati delle attività di rilevamento delle prove con etichette di convalida.
- `LookoutVisionConsoleDashboardAccess`- Consente la lettura di AmazonCloudWatchmetriche. Lookout for Vision utilizza queste azioni per compilare i grafici del pannello di controllo e le statistiche rilevate sulle anomalie.

- `LookoutVisionConsoleTagSelectorAccess`— Consente di leggere suggerimenti relativi alla chiave e al valore dei tag specifici dell'account. Lookout for Vision utilizza queste autorizzazioni per fornire consigli sulle chiavi dei tag e sui valori dei tag all'interno diGestisci i tagpagine della console.
- `LookoutVisionConsoleKmsKeySelectorAccess`— Consente la quotazioneAWS Key Management ServiceChiavi e alias (KMS). Amazon Lookout for Vision utilizza questa autorizzazione per inserire le chiavi KMS nei campi suggeritiEtichetteselezione su determinate azioni di Lookout for Vision che supportano le chiavi KMS gestite dal cliente per la crittografia.

```
{
    "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": "*"
        }
    ]
}
```

Policy gestita da AWS: AmazonLookoutVisionConsoleReadOnlyAccesso

Usa il`AmazonLookoutVisionConsoleReadOnlyAccess`politica per consentire agli utenti l'accesso in sola lettura alla console Amazon Lookout for Vision, alle azioni (operazioni SDK) e a qualsiasi dipendenza del servizio.

La`AmazonLookoutVisionConsoleReadOnlyAccess`la policy include le autorizzazioni Amazon S3 per il bucket della console Amazon Lookout for Vision. Se il tuo set di dati, immagini o AmazonSageMakerI file manifest di Ground Truth si trovano in un bucket Amazon S3 diverso, i tuoi utenti necessitano di autorizzazioni aggiuntive. Per ulteriori informazioni, consulta the section called “Impostazione delle autorizzazioni per i bucket Amazon S3”.

È possibile allegare la policy `AmazonLookoutVisionConsoleReadOnlyAccess` alle identità IAM.

Raggruppamenti di autorizzazioni

Questa politica è raggruppata in dichiarazioni basate sul set di autorizzazioni fornite:

- `LookoutVisionReadOnlyAccess`— Consente l'accesso per eseguire azioni di Lookout for Vision di sola lettura.
- `LookoutVisionConsoleS3BucketSearchAccess`— Consente l'elenco di tutti i bucket S3 di proprietà del chiamante. Lookout for Vision utilizza questa azione per identificare il bucket della console Lookout for Vision specifico per regione AWS, se presente nell'account del chiamante.

- `LookoutVisionConsoleS3ObjectReadAccess`— Consente di leggere le versioni di oggetti Amazon S3 e Amazon S3 nei bucket della console Lookout for Vision. Lookout for Vision utilizza queste azioni per visualizzare le immagini nei set di dati, nei modelli e nei rilevamenti di prova.
- `LookoutVisionConsoleDashboardAccess`— Consente di leggere AmazonCloudWatchmetriche. Lookout for Vision utilizza queste azioni per compilare le statistiche relative ai grafici del dashboard e alle anomalie rilevate.

```
{
    "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": "*"
        }
    ]
}
```

Aggiornamenti di Lookout for Vision aAWSpolitiche gestite

Visualizza i dettagli sugli aggiornamenti diAWSha gestito le politiche per Lookout for Vision da quando questo servizio ha iniziato a tenere traccia di queste modifiche. Per ricevere avvisi automatici sulle modifiche a questa pagina, iscriviti al feed RSS nella pagina della cronologia dei documenti di Lookout for Vision.

Aggiunte le operazioni di confezionamento del modello	Amazon Lookout for Vision ha aggiunto i seguenti modelli di operazioni di imballaggio alAmazonLookoutVisionFullAccesseAmazonLookoutVisionConsoleFullAccess politiche: • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob Amazon Lookout for Vision ha aggiunto i seguenti modelli di operazioni di imballaggio alAmazonLookoutVisionReadOnlyAccesseAmazonLookoutVisionConsoleReadOnlyAccesspolitiche: • DescribeModelPackagingJob • ListModelPackagingJobs	7 dicembre 2021
Nuove politiche aggiunte	Amazon Lookout for Vision ha aggiunto le seguenti politiche. • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess • AmazonLookoutVisionConsoleReadOnlyAccess	11 maggio 2021
Lookout for Vision ha iniziato a tracciare le modifiche	Amazon Lookout for Vision ha iniziato a monitorare le	1 marzo 2021

Modifica	Descrizione	Data
Aggiunte le operazioni di confezionamento del modello	Amazon Lookout for Vision ha aggiunto i seguenti modelli di operazioni di imballaggio alAmazonLookoutVisionFullAcesseAmazonLookoutVisionConsoleFullAccess politiche: • DescribeModelPackagingJob • ListModelPackagingJobs • StartModelPackagingJob Amazon Lookout for Vision ha aggiunto i seguenti modelli di operazioni di imballaggio alAmazonLookoutVisionReadOnlyAcesseAmazonLookoutVisionConsoleReadOnlyAccesspolitiche: • DescribeModelPackagingJob • ListModelPackagingJobs	7 dicembre 2021
Nuove politiche aggiunte	Amazon Lookout for Vision ha aggiunto le seguenti politiche. • AmazonLookoutVisionReadOnlyAccess • AmazonLookoutVisionFullAccess • AmazonLookoutVisionConsoleFullAccess • AmazonLookoutVisionConsoleReadOnlyAccess modifiche per il suoAWSpolitiche gestite.	11 maggio 2021

Risoluzione dei problemi relativi all'identità e all'accesso ad Amazon Lookout for Vision

Utilizza le seguenti informazioni per aiutarti a diagnosticare e risolvere i problemi più comuni che potresti riscontrare quando lavori con Lookout for Vision e IAM.

Argomenti

Non sono autorizzato a eseguire un'azione in Lookout for Vision

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, le tue policy devono essere aggiornate per poter eseguire l'operazione.

L'errore di esempio seguente si verifica quando l'utente IAM `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa *`my-example-widget`* fittizia ma non dispone di autorizzazioni `lookoutvision:`*`GetWidget`* fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lookoutvision:GetWidget on resource: my-example-widget
```

In questo caso, la policy per l'utente `mateojackson` deve essere aggiornata per consentire l'accesso alla risorsa *`my-example-widget`* utilizzando l'azione `lookoutvision:`*`GetWidget`*.

Per ulteriore assistenza con l'accesso, contatta l'amministratore AWS. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Non sono autorizzato a eseguire iam: PassRole

Se ricevi un messaggio di errore indicante che non sei autorizzato a eseguire l'`iam:PassRole`azione, le tue politiche devono essere aggiornate per consentirti di trasferire un ruolo a Lookout for Vision.

Alcuni Servizi AWSconsentono di trasmettere un ruolo esistente a tale servizio, invece di creare un nuovo ruolo di servizio o un ruolo collegato ai servizi. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` tenta di utilizzare la console per eseguire un'azione in Lookout for Vision. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Per ulteriore assistenza con l'accesso, contatta l'amministratore AWS. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire a persone esterne a me di accedere Account AWS alle mie risorse Lookout for Vision

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo accessi (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per sapere se Lookout for Vision supporta queste funzionalità, In che modo Amazon Lookout for Vision funziona con IAM consulta.
- Per informazioni su come garantire l'accesso alle risorse negli Account AWSche possiedi, consulta Fornire l'accesso a un utente IAM in un altro Account AWS in tuo possesso nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso alle risorse ad Account AWS di terze parti, consulta Fornire l'accesso agli Account AWS di proprietà di terze parti nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta Fornire l'accesso a utenti autenticati esternamente (Federazione delle identità) nella Guida per l'utente di IAM.
- Per informazioni sulle differenze tra l'utilizzo di ruoli e policy basate su risorse per l'accesso multi-account, consultare Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.

Convalida della conformità per Amazon Lookout for Vision

Revisori di terze parti valutano la sicurezza e la conformità di Amazon Lookout for Vision nell'ambito di AWS diversi programmi di conformità. Amazon Lookout for Vision è conforme al Regolamento generale sulla protezione dei dati (GDPR).

Per sapere se il Servizio AWSè coperto da programmi di conformità specifici, consulta i Servizi AWScoperti dal programma di conformità e scegli il programma di conformità desiderato. Per informazioni generali, consulta Programmi per la conformità di AWS.

È possibile scaricare i report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Download di report in AWS Artifact.

La responsabilità di conformità durante l'utilizzo dei Servizi AWSè determinata dalla riservatezza dei dati, dagli obiettivi di conformità dell'azienda e dalle normative vigenti. Per semplificare il rispetto della conformità, AWSmette a disposizione le seguenti risorse:

- Guide Quick Start per la sicurezza e conformità: queste guide all'implementazione illustrano considerazioni relative all'architettura e forniscono la procedura per l'implementazione di ambienti di base su AWSincentrati sulla sicurezza e sulla conformità.
- Architetture per la sicurezza e la conformità HIPAA su Amazon Web Services: questo whitepaper descrive come le aziende possono utilizzare AWSper creare applicazioni conformi alla normativa HIPAA.

  > **Note**
  >
  > Non tutti i Servizi AWSsono conformi ai requisiti HIPAA. Per ulteriori informazioni, consulta la sezione Riferimenti sui servizi conformi ai requisiti HIPAA.

- Risorse per la conformità AWS: una raccolta di cartelle di lavoro e guide suddivise per settore e area geografica.
- AWSGuide alla conformità dei clienti: comprendi il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione Servizi AWS e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization (ISO)).

- Valutazione delle risorse con le regole nella Guida per gli sviluppatori di AWS Config: il servizio AWS Configvaluta il livello di conformità delle configurazioni delle risorse con pratiche interne, linee guida e regolamenti.
- AWS Security Hub: questo Servizio AWSfornisce una visione completa dello stato di sicurezza all'interno di AWS. La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse AWSe verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina Documentazione di riferimento sui controlli della Centrale di sicurezza.
- AWS Audit Manager: questo Servizio AWSaiuta a verificare continuamente l'utilizzo di AWSper semplificare la gestione dei rischi e della conformità alle normative e agli standard di settore.

La resilienza di Amazon Lookout for Vision

L'infrastruttura globale di AWS è basata su Regioni e zone di disponibilità AWS. AWS Le Regioni forniscono più zone di disponibilità fisicamente separate e isolate che sono connesse tramite reti altamente ridondanti, a bassa latenza e velocità effettiva elevata. Con le Zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le Zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili, rispetto alle infrastrutture a data center singolo o multiplo.

Per ulteriori informazioni sulle Regioni AWS e sulle zone di disponibilit, consulta Infrastruttura globale di AWS.

Sicurezza dell'infrastruttura in Amazon Lookout for Vision

In quanto servizio gestito, Amazon Lookout for Vision è protetto dalla sicurezza di rete AWS globale. Per informazioni sui servizi di sicurezza AWS e su come AWS protegge l'infrastruttura, consulta la pagina Sicurezza del cloud AWS. Per progettare l'ambiente AWS utilizzando le best practice per la sicurezza dell'infrastruttura, consulta la pagina Protezione dell'infrastruttura nel Pilastro della sicurezza di AWS Well-Architected Framework.

Utilizzi le chiamate API AWS pubblicate per accedere a Lookout for Vision tramite la rete. I clienti devono supportare quanto segue:

- Transport Layer Security (TLS). È richiesto TLS 1.2 ed è consigliato TLS 1.3.

- Suite di cifratura con Perfect Forward Secrecy (PFS), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. In alternativa, è possibile utilizzare AWS Security Token Service (AWS STS) per generare le credenziali di sicurezza temporanee per sottoscrivere le richieste.

Monitoraggio Amazon Lookout for Vision

Il monitoraggio è importante per garantire l'affidabilità, la disponibilità e le prestazioni di Amazon Lookout for Vision AWS fornisce i seguenti strumenti di monitoraggio per tenere Lookkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk

- Amazon CloudWatch monitora leAWS risorse e le applicazioni che esegui suAWS in tempo reale. Puoi raccogliere i parametri e tenerne traccia, creare pannelli di controllo personalizzati e impostare allarmi per inviare una notifica o intraprendere azioni quando un parametro specificato raggiunge una determinata soglia. Ad esempio, puoi impostare perché CloudWatch tenga traccia dell'uso della CPU o di altri parametri delle tue istanze Amazon EC2 e avviare automaticamente nuove istanze quando necessario. Per ulteriori informazioni, consulla la Guida per CloudWatch l'utente di Amazon.
- Amazon CloudWatch Logs consente di monitorare, archiviare e accedere ai file di log dalle istanze Amazon EC2 e da altre origini. CloudTrail CloudWatch I log possono monitorare le informazioni nei file di log e notificare quando vengono raggiunte determinate soglie. Puoi inoltre archiviare i dati del log in storage estremamente durevole. Per ulteriori informazioni, consulla la Guida per l'utente CloudWatch di Amazon Logs.
- Amazon EventBridge può essere utilizzato per automatizzare iAWS servizi e rispondere automaticamente a eventi di sistema, come i problemi relativi alla disponibilità delle applicazioni o le modifiche delle risorse. Gli eventi deiAWS servizi vengono recapitati EventBridge a quasi in tempo reale. Puoi compilare regole semplici che indichino quali eventi sono considerati di interesse per te e quali operazioni automatizzate intraprendere quando un evento corrisponde a una regola. Per ulteriori informazioni, consulla la Guida per EventBridge l'utente di Amazon.
- AWS CloudTrail acquisisce le chiamate API e gli eventi correlati effettuati da o per conto del tuo account AWS e fornisce i file di log a un bucket Amazon S3 specificato. Puoi identificare quali utenti e account hanno richiamato AWS, l'indirizzo IP di origine da cui sono state effettuate le chiamate e quando sono avvenute. Per ulteriori informazioni, consultare la Guida per l'utente AWS CloudTrail.

Monitoraggio Lookout for Vision CloudWatch

Puoi monitorare Lookout for Vision CloudWatch Queste statistiche vengono conservate per un periodo di 15 mesi, per permettere l'accesso alle informazioni storiche e offrire una prospettiva migliore sulle prestazioni del servizio o dell'applicazione Web. È anche possibile impostare allarmi

che controllano determinate soglie e inviare notifiche o intraprendere azioni quando queste soglie vengono raggiunte. Per ulteriori informazioni, consulla la Guida per CloudWatch l'utente di Amazon.

Il servizio Lookout for VisionAWS/LookoutVision

Parametro	Descrizione
DetectedAnomalyCount	Il numero di anomalie rilevate in un progetto Statistiche valide:Sum,Average Unità: numero
ProcessedImageCount	Il numero totale di immagini sottoposte al rilevamento delle anomalie Statistiche valide:Sum,Average Unità: numero
InvalidImageCount	Il numero di immagini che non erano valide e non potevano restituire un risultato Statistiche valide:Sum,Average Unità: numero
SuccessfulRequestCount	Il numero di chiamate API riuscite correttamente. Statistiche valide:Sum,Average Unità: numero
ErrorCount	Il numero di errori delle API. Statistiche valide:Sum,Average Unità: numero
ThrottledCount	Il numero di errori API dovuti alla limitazione Statistiche valide:Sum,Average

Parametro	Descrizione
	Unità: numero
`Time`	Il tempo in millisecondi impiegato da Lookout for Vision per calcolare il rilevamento delle anomalie Statistiche valide:`Data Samples,Average` Unità: Millisecondi per le`Average` statistiche e Conteggio per`Data Samples` le statistiche
`MinInferenceUnits`	Il numero minimo di unità di inferenza specificato durante la`StartModel` richiesta. Statistiche valide: `Average` Unità: numero
`MaxInferenceUnits`	Il numero massimo di unità di inferenza massimo per l'inferenza specificato durante la`StartModel` richiesta. Statistiche valide: `Average` Unità: numero
`DesiredInferenceUnits`	Il numero di unità di inferenza a cui Lookout for Vision sta aumentando o diminuendo. Statistiche valide: `Average` Unità: numero

Parametro	Descrizione
`InServiceInferenceUnits`	Il numero di unità di inferenza utilizzate dal modello. Statistiche valide: `Average` Si consiglia di utilizzare la statistica Average per ottenere la media di 1 minuto del numero di istanze utilizzate. Unità: numero

Le seguenti dimensioni sono supportate per i parametri Lookout for Vision

Dimensione	Descrizione
`ProjectName`	Puoi suddividere le metriche per progetto per vedere quali progetti presentano problemi o devono essere aggiornati.
`ModelVersion`	Puoi suddividere le metriche per versione del modello per vedere quali modelli presentano problemi o devono essere aggiornati.

Registrazione delle chiamate API Lookout for Vision conAWS CloudTrail

Amazon Lookout for Vision Lookout for Vision.AWS CloudTrailAWS CloudTrail acquisisce tutte le chiamate API per Lookout for Vision. Le chiamate acquisite includono le chiamate dalla console Lookout for Vision e le chiamate di codice alle operazioni API Lookout for Vision. Se si crea un percorso, è possibile abilitare la distribuzione continua di CloudTrail eventi in un bucket Amazon S3, inclusi gli eventi per Lookout for Vision. Se non configuri un trail, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console in Cronologia degli eventi. Le informazioni raccolte da consentono CloudTrail di determinare la richiesta effettuata a Lookout for Vision, l'indirizzo IP da cui è partita la richiesta, l'autore della richiesta, il momento in cui è stata eseguita e altri dettagli.

Per ulteriori informazioni CloudTrail, consulta la Guida perAWS CloudTrail l'utente.

Lookout for Vision in CloudTrail

CloudTrail è abilitato sull'AWSaccount al momento della sua creazione. Quando si verifica un'attività in Lookout for Vision in un CloudTrail evento insieme ad altri eventi diAWS servizio. È possibile visualizzare, cercare e scaricare gli eventi recenti nell'account AWS. Per ulteriori informazioni, consulta Visualizzazione di eventi mediante la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi nell'AWSaccount, inclusi gli eventi relativi a Lookout for for for for for for for for for for for. Un trail consente di CloudTrail distribuire i file di log Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le regioni AWS. Il trail registra gli eventi di tutte le Regioni nella partizione AWS e distribuisce i file di registro nel bucket Amazon S3 specificato. Inoltre, puoi configurare altriAWS servizi per analizzare con maggiore dettaglio e usare i dati raccolti nei CloudTrail log. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Panoramica della creazione di un percorso
- CloudTrail Servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail log da più regioni e Ricezione di file di CloudTrail log da più account

Tutte le azioni di Lookout for Vision vengono registrate CloudTrail e documentate nella documentazione di riferimento dell'API Lookout for Vision. Ad esempio, le chiamate a`DetectAnomalies` e`StartModel` le azioni generano voci nei file di CloudTrail registro.`CreateProject`

Se si monitorano le chiamate API Amazon Lookout for.

- visione d'occhio:StartTriallDetection
- visione d'occhio:ListTriallDetection
- visione d'occhio:DescribeTrialDetection

Queste chiamate speciali vengono utilizzate da Amazon Lookout for Vision per supportare varie operazioni relative al rilevamento delle versioni di prova. Per ulteriori informazioni, consulta Verifica del modello con un'attività di rilevamento delle versioni di prova.

Ogni evento o voce di registro contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente AWS Identity and Access Management o root.
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro servizio AWS.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

Comprensione delle voci dei file di registro di Lookout for Vision

Un trail è una configurazione che consente la distribuzione di eventi come i file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di log possono contenere una o più voci di log. Un evento rappresenta una singola richiesta da un'origine e include informazioni sull'operazione richiesta, data e ora dell'operazione, parametri della richiesta e così via. CloudTrail i file di log non sono una traccia stack ordinata delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail log di che illustra l'`CreateDataset`operazione.


```
{
  "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"
}
```

Creare le risorse Amazon Amazon Lookout for Vision Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon AmazonAWS CloudFormation

Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon Amazon LooAmazon LooLoodi Amazon LooAmazon Loodi Amazon Loodi Amazon LooAmazon LooAmazon Amazon LooLooLooLooLooLooLooLooLooLooLooLooLooLooAWS CloudFormationAWS É possibile creare un modello che descriva tutte leAWS risorse che desideri eAWS CloudFormation si occuperà di allestire e configurare le risorse per tuo conto.

Puoi utilizzarloAWS CloudFormation per effettuare il provisioning e configurare progetti Amazon Lookout for Vision.

Quando utilizziAWS CloudFormation, puoi riutilizzare il modello per configurare i progetti Lookout for Vision Descrivere i i progetti una volta e quindi allestisci gli stessi progetti più volte in più regioni e account AWS.

Lookout for VisionAWS CloudFormation

É necessario conoscere i AWS CloudFormationmodelli, prima di poter effettuare il provisioning e la configurazione dei progetti per LooLooLooLooLooLooLooLooLoo I modelli sono file di testo formattati in JSON o YAML. Questi modelli descrivono le risorse di cui intendi effettuare il provisioning negli stack AWS CloudFormation. Se non hai familiarità con JSON o YAML, puoi usare AWS CloudFormation Designer per iniziare a utilizzare i modelli AWS CloudFormation. Per ulteriori informazioni, consulta Che cos'è AWS CloudFormation Designer? nella Guida per l'utente di AWS CloudFormation.

Per informazioni di riferimento dei progetti di LookoutVision LooLooLooLooLooLooLooLooLooLooLooLooLooLooLooLooLooLooLooLooLoo

Ulteriori informazioni su AWS CloudFormation

Per ulteriori informazioni su AWS CloudFormation, consulta le seguenti risorse:

- AWS CloudFormation
- Guida per l'utente di AWS CloudFormation
- Documentazione di riferimento dell'API AWS CloudFormation
- Guida per l'utente dell'interfaccia a riga di comando di AWS CloudFormation

Accedi ad Amazon Lookout for Vision utilizzando un endpoint di interfaccia (AWS PrivateLink)

Puoi utilizzareAWS PrivateLink per creare una connessione privata tra il tuo VPC e Amazon Lookout for Vision. Puoi accedere a Lookout for Vision come se fosse nel VPC, senza l'uso di un gateway Internet, un dispositivo NAT, una connessione VPN oAWS Direct Connect una connessione. Le istanze presenti nel VPC non richiedono indirizzi IP pubblici per accedere a Lookout for Vision.

Stabilisci questa connessione privata creando un endpoint di interfaccia attivato da AWS PrivateLink. In ciascuna sottorete viene creata un'interfaccia di rete endpoint da abilitare per l'endpoint di interfaccia. Queste sono interfacce di rete gestite dal richiedente che fungono da punto di ingresso per il traffico destinato a Lookout for Vision.

Per ulteriori informazioni, consulta la sezione Accesso a Servizi AWS tramite AWS PrivateLink nella Guida di AWS PrivateLink.

Considerazioni sugli endpoint Lookout for Vision VPC

Prima di configurare un endpoint di interfaccia per Lookout for Vision, consulta le considerazioni nella AWS PrivateLinkGuida.

Lookout for Vision supporta l'esecuzione di chiamate a tutte le sue operazioni API tramite l'endpoint di interfaccia.

Le policy di endpoint VPC non sono supportate per Lookout for Vision. Per impostazione predefinita, l'accesso completo a Lookout for Vision è consentito tramite l'endpoint di interfaccia. In alternativa, è possibile associare un gruppo di sicurezza alle interfacce di rete degli endpoint per controllare il traffico verso Lookout for Vision tramite l'endpoint di interfaccia.

Creazione di un endpoint VPC di interfaccia per Lookout for Vision

Puoi creare un endpoint di interfaccia per Lookout for Vision utilizzando la console Amazon VPC oAWS Command Line Interface (AWS CLI). Per ulteriori informazioni, consulta la sezione Creazione di un endpoint di interfaccia nella Guida per l'utente di AWS PrivateLink.

Crea un endpoint di interfaccia per Lookout for Vision utilizzando uno dei seguenti nomi di servizi:

```
com.amazonaws.region.lookoutvision
```

Se si abilita il DNS privato per l'endpoint di interfaccia, è possibile effettuare richieste API verso Lookout for Vision utilizzando il nome DNS regionale predefinito. Ad esempio, `lookoutvision.us-east-1.amazonaws.com`.

Creazione di una policy di endpoint VPC per Lookout for Vision

Una policy di endpoint è una risorsa IAM che è possibile allegare all'endpoint di interfaccia. La policy di endpoint di default consente l'accesso completo a Lookout for Vision tramite l'endpoint di interfaccia. Per controllare l'accesso consentito a Lookout for Vision dal VPC, allegare una policy di endpoint personalizzata all'endpoint di interfaccia.

Una policy di endpoint specifica le informazioni riportate di seguito:

- I responsabili che possono eseguire azioni (Account AWSutenti IAM e ruoli IAM).
- Le operazioni che possono essere eseguite.
- Le risorse in cui è possibile eseguire le operazioni.

Per ulteriori informazioni, consulta la sezione Controllo dell'accesso ai servizi con policy di endpoint nella Guida di AWS PrivateLink.

Ad esempio, policy di endpoint VPC per le operazioni Lookout for Vision

Di seguito è riportato un esempio di una policy di endpoint personalizzata per Lookout for Vision. Quando si collega questa policy all'endpoint dell'interfaccia, specifica che tutti gli utenti che hanno accesso all'endpoint dell'interfaccia VPC possono chiamare l'operazione`DetectAnomalies` API per il modello Lookout for Vision`myModel` associato al progetto`myProject`.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "lookoutvision:DetectAnomalies"
         ],
         "Resource": "arn:aws:lookoutvision:us-west-2:123456789012:model/myProject/
myModel"
```

```
        }
    ]
}
```

Quote in Amazon Lookout for Vision

Le tabelle seguenti descrivono le quote correnti in Amazon Lookout for Vision. Per informazioni sulle quote che possono essere modificate, vedereQuote dei servizi AWS.

Quote modello

Le quote seguenti si applicano al test, alla formazione e alla funzionalità di un modello.

Risorsa	Quota
Formato di file supportato	Formati di immagine PNG e JPEG
Dimensione minima dell'immagine del file di immagine in un bucket Amazon S3	64 pixel x 64 pixel
Dimensione massima dell'immagine del file di immagine in un bucket Amazon S3	4096 pixel X 4096 pixel è il massimo. Le dimensioni più piccole possono essere caricate più velocemente.
Dimensioni diverse delle immagini dei file di immagine utilizzati in un progetto	Tutte le immagini del set di dati devono avere le stesse dimensioni
Dimensione massima del file per un'immagine in un bucket Amazon S3	8 MB
Mancanza di etichette	Le immagini devono essere etichettate come normali o anomale prima dell'allenamento. Le immagini senza etichette vengono ignorate durante l'allenamento.
Numero minimo di immagini etichettate come normali nel set di dati di addestramento	10 per un progetto con set di dati di training e test separati. 20 per un progetto con un singolo set di dati.
Numero minimo di immagini etichettate come anomalie in un set di dati di addestramento	0 per un progetto con set di dati di training e test separati. 10 per un progetto con un singolo set di dati.

Risorsa	Quota
Numero massimo di immagini nel set di dati di addestramento alla classificazione	16,000
Numero massimo di immagini in un set di dati di test di classificazione	4.000
Numero minimo di immagini etichettate come normali nel set di dati di test	10
Numero minimo di immagini etichettate come anomalie nel set di dati del test	10
Numero massimo di immagini in un set di dati di addestramento alla localizzazione delle anomalie	8000
Numero massimo di immagini in un set di dati del test di localizzazione delle anomalie	800
Numero massimo di immagini nel set di dati di rilevamento della versione di prova	2.000
Dimensione massima del file del manifesto del set di dati	1 GB
Numero massimo di set di dati di addestram ento in un modello	1
Tempo massimo di allenamento	24 ore
Tempo massimo di test	24 ore
Numero massimo di etichette di anomalia in un progetto	100
Numero massimo di etichette di anomalia su un'immagine della maschera	20

Risorsa	Quota
Numero minimo di immagini per un'etichetta di anomalia. Per contare, l'immagine deve contenere un solo tipo di etichetta di anomalia.	20 per un singolo progetto di set di dati. 10 per ogni set di dati in un progetto con set di dati di addestramento e test separati.

Cronologia dei documenti per Amazon Lookout for Vision

La tabella seguente descrive le modifiche importanti introdotte in ogni versione della Guida per gli sviluppatori di Amazon Lookout for Vision. Per ricevere notifiche sugli aggiornamenti di questa documentazione, è possibile abbonarti a un feed RSS.

- Ultimo aggiornamento della documentazione: 20 febbraio 2023

Modifica	Descrizione	Data
È stato aggiunto un esempio di funzione Lambda	Esempio che mostra come trovare anomalie con unaAWS Lambda funzione. Per ulteriori informazioni, consulta Nozioni di base su una funzione AWS Lambda	20 febbraio 2023
Aggiornata la guida IAM per la visioneAWS WAF	Guida aggiornata per allinears i alle best practice IAM. Per ulteriori informazioni, consulta la sezione Best practice per la sicurezza in IAM	8 febbraio 2023
È stato aggiunto un esempio di esportazione di set di dati	È stato aggiunto un esempio in Python che mostra come utilizzare l'`ListDatasetEntries` operazione per esportare i set di dati da un progetto Amazon Lookout for Vision. Per ulteriori informazioni, consulta Esportazione di set di dati da un progetto (SDK).	2 dicembre 2022
Argomento introduttivo aggiornato	Guida introduttiva aggiornata per mostrare la creazione di un modello di segmentaz	20 ottobre 2022

	ione delle immagini con un set di dati di esempio. Per ulteriori informazioni, consultare Nozioni di base su Amazon Lookout for Vision.	
Aggiunto argomento di risoluzione dei problemi	È stato aggiunto l'argomento relativo alla risoluzione dei problemi relativi alla formazione e Per ulteriori informazioni, consulta Nozioni di base su i modelli.	17 ottobre 2022
È stato aggiunto un argomento sull'utilizzo dei lavori di Amazon SageMaker Ground Truth	Invece di etichettare personalmente le immagini, puoi utilizzare i job di Amazon SageMaker Ground Truth per etichettare le immagini per i modelli di classificazione e segmentazione delle immagini. Per ulteriori informazioni, consulta Utilizzare un job Amazon SageMaker Ground Truth.	19 agosto 2022
Amazon Lookout for Vision ora fornisce la localizzazione delle anomalie.	Puoi creare un modello di segmentazione che trovi le posizioni su un'immagine in cui sono presenti diversi tipi di anomalie (come graffi, ammaccature o lacerazioni), l'etichetta dell'anomalia e la dimensione dell'anomalia. Per ulteriori informazioni, consulta Esecuzione di un modello Amazon Lookout for Vision addestrato.	16 agosto 2022

Amazon Lookout for Vision ora fornisce l'inferenza della CPU sui dispositivi periferici.	I modelli Amazon Lookout for Vision possono ora essere distribuiti per eseguire inferenze localmente su una piattaforma di elaborazione x86 che esegue Linux con una sola CPU, senza bisogno di un acceleratore GPU. Per ulteriori informazioni, consulta CPU Accelerator.	16 agosto 2022
Amazon Lookout for Vision ora può scalare automaticamente le unità di inferenza.	Per far fronte ai picchi di domanda, Amazon Lookout for Vision è ora in grado di scalare il numero di unità di inferenza utilizzate dal modello. Per ulteriori informazioni, consulta Nozioni di base su Amazon Lookout for Vision.	16 agosto 2022
Esempi Java aggiunti	La guida per sviluppatori di Amazon Lookout for Vision ora include esempi Java. Per ulteriori informazioni, consulta Nozioni di base su l'AWSSDK.	2 maggio 2022
Disponibilità generale dell'implementazione del modello su un dispositivo periferico	L'implementazione del modello su un dispositivo periferico gestito daAWS IoT Greengrass Version 2 è ora generalmente disponibile. Per ulteriori informazioni, consultare Utilizzo del modello Amazon Lookout for Vision.	14 marzo 2022

Informazioni aggiornate sul bucket della console	Informazioni aggiornate sul contenuto del bucket della console e approcci alternativi alla creazione del bucket della console. Per ulteriori informazioni, vedi Fase 4: Creazione del bucket della console.	7 marzo 2022
Crea un file manifest da un file CSV	È ora possibile semplificare la creazione di un file manifest utilizzando uno script che legge le informazioni di classificazione da un file CSV. Per ulteriori informazioni, consulta Creazione di un file manifest da un file CSV.	10 febbraio 2022
Versione in anteprima della distribuzione del modello su un dispositivo periferico	La versione di anteprima della distribuzione del modello su un dispositivo edge gestito daAWS IoT Greengrass Version 2 è ora disponibile. Per ulteriori informazioni, consultare Utilizzo del modello Amazon Lookout for Vision.	7 dicembre 2021
Aggiunti nuovi esempi in Python e Java 2	Sono stati aggiunti esempi in Python e Java 2 per l'analisi delle immagini con`DetectAnomalies` . Per ulteriori informazioni, consulta Nozioni di base su un'immagine.	7 settembre 2021

Sono state aggiunte nuove politicheAWS gestite.	Amazon Lookout for Vision aggiunge il supporto per le politicheAWS gestite. Per ulteriori informazioni, consultare le politicheAWS gestite per Amazon Lookout for Vision.	11 maggio 2021
Informazioni aggiornate sull'unità di inferenza.	Sono state aggiunte informazioni che descrivono le unità di inferenza e come vengono caricate. Per ulteriori informazioni, consulta Nozioni di base su Amazon Lookout for Vision.	15 marzo 2021
Disponibilità generale per Amazon Lookout for Vision.	Amazon Lookout for Vision è ora disponibile al pubblico. Esempi di codice Python aggiornati per gestire attività asincrone come l'addestramento di un modello.	17 febbraio 2021
Etichettatura eAWS CloudFormation supporto aggiunti.	Ora puoi taggare i modelli Amazon Lookout for Vision e creare progetti conAWS CloudFormation. Per ulteriori informazioni, consulta Etichettatura dei modelli e creazione di progetti Amazon Lookout for Vision con AWS CloudFormation.	31 gennaio 2021
Nuova funzionalità e guida	Questa è la versione iniziale del servizio Amazon Lookout for Vision Amazon Lookout for Vision Developer Guide.	1° dicembre 2020

Glossario per AWS

Per la terminologia AWS più recente, consultare il glossario AWS nella documentazione di riferimento per Glossario AWS.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.