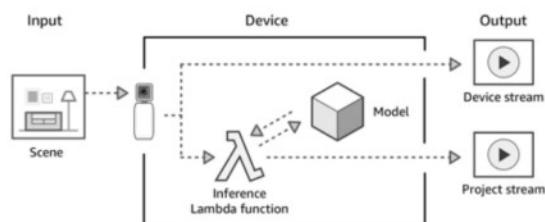
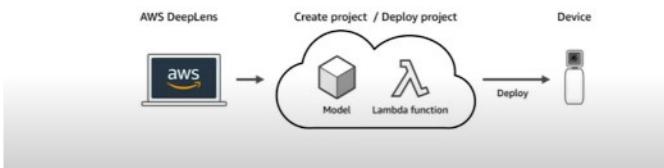


Vídeo Comunitário

AWS DeepLens

Occurs on the AWS Console

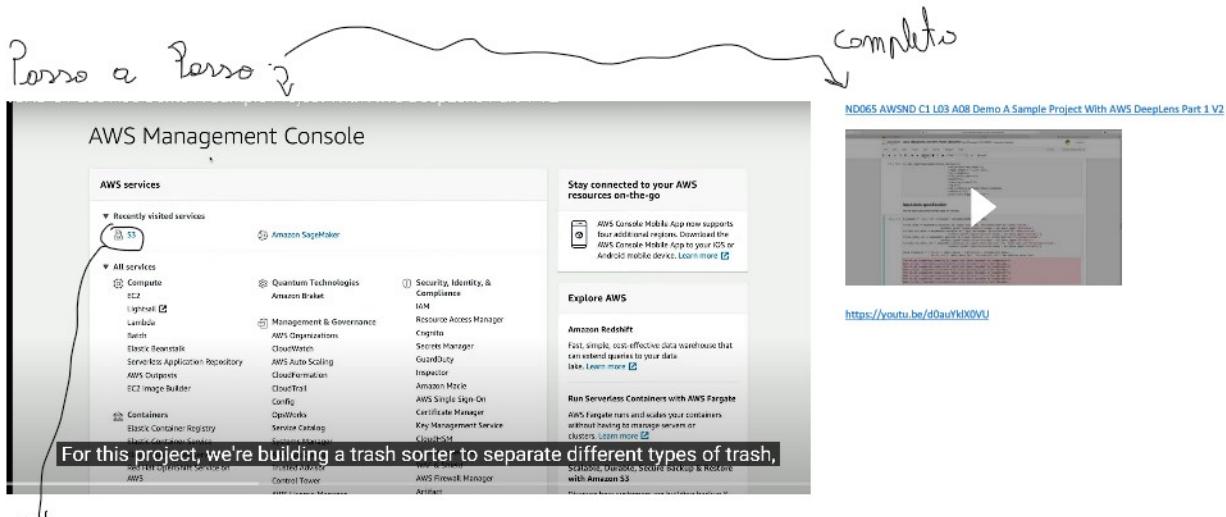


How AWS DeepLens works

- First, you use the AWS console to create your project, store your data, and train your model.
 - Then, you use your trained model on the AWS DeepLens device. On the device, the video stream from the camera is processed, inference is performed, and the output from inference is passed into two output streams:
 - **Device stream** – The video stream passed through without processing.
 - **Project stream** – The results of the model's processing of the video frames.

Four key components are required for an AWS DeepLens-based project.

- 1. Collect your data:** Collect data and store it in an Amazon S3 bucket.
 - 2. Train your model:** Use a Jupyter Notebook in Amazon SageMaker to train your model.
 - 3. Deploy your model:** Use AWS Lambda to deploy the trained model to your AWS DeepLens device.
 - 4. View model output:** Use Amazon IoT Greengrass to view your model's output after the model is deployed.



Amazon S3

Buckets

Access Points

Object Lambda Access Points

Batch Operations

Access analyzer for S3

Block Public Access settings for this account

Storage Lens

Dashboards

AWS Organizations settings

Feature spotlight

AWS Marketplace for S3

Buckets (1) → É como se fossemos para remoto

Buckets are containers for data stored in S3. Learn more

Find buckets by name:

Name	AWS Region	Access	Creation date
cloudtrail-awilog-695958022619-twdvtrt-sengard-do-not-delete	US East (N. Virginia) us-east-1	Objects can be public	September 13, 2019, 16:31:57 (UTC-07:00)
deeplearn-trash	US East (N. Virginia) us-east-1	Bucket and objects can be public	April 19, 2021, 16:45:26 (UTC-07:00)
do-not-delete-gatedgarden-audit-695958022619	US West (Oregon) us-west-2	Objects can be public	September 13, 2019, 16:49:27 (UTC-07:00)
scribe-media-prod-695958022619	US East (N. Virginia) us-east-1	Objects can be public	September 13, 2019, 17:24:09 (UTC-07:00)
scribe-results-prod-695958022619	US East (N. Virginia) us-east-1	Objects can be public	September 13, 2019, 17:24:09 (UTC-07:00)
scribe-website-prod-695958022619	US East (N. Virginia) us-east-1	Objects can be public	September 13, 2019, 17:22:59 (UTC-07:00)

Create bucket

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Actions

Upload

Find objects by prefix:

Name	Type	Last modified	Size	Storage class
deeplearn-trash/	Folder	-	-	-

Objects (5)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 Inventory](#) to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. [Learn more](#)

Actions

Upload

Find objects by prefix:

Name	Type	Last modified	Size	Storage
image/	Folder	-	-	-
output/	Folder	-	-	-
test.txt	txt	April 19, 2021, 16:51:08 (UTC-07:00)	8.0 B	Standard
train_lst/	Folder	-	-	-
validation_lst/	Folder	-	-	-

Então faço isso os imagens que preciso para treinar meu modelo

Depois venho para o SageMaker usando a base de perguntas

WSND C1 L03 A08 Demo A Sample Project With AWS DeepLens Part 1 V2

amazon sagemaker

Pressione Esc para sair do modo tela cheia.

Search results for amazon sagemaker

Services (50)

Amazon S3

Services (50)

Amazon SageMaker Build, Train, and Deploy Machine Learning Models

Amazon Redshift Fast, Simple, Cost-Effective Data Warehousing

Amazon EventBridge Serverless event-bus that connects application data from your own apps, SaaS, and AWS services

Amazon Timestream Amazon Timestream is a fast, scalable, and serverless time series database for IoT and time-series workloads

Copy S3

Amazon SageMaker Studio

Dashboard Search Images

- ▶ Ground Truth
- ▶ Notebook **Notebook**
- ▶ Processing
- ▶ Training
- ▶ Inference
- ▶ Edge Manager
- ▶ Augmented AI
- ▶ AWS Marketplace

Amazon SageMaker > Dashboard

AWS Marketplace Find, buy, and deploy ready to use model packages, algorithms, and data products in AWS Marketplace [Browse Catalog](#)

SageMaker Dashboard

[Open SageMaker Studio](#)

Overview

Prepare Build Train & Tune Deploy & Manage

Studio [Open Studio](#)

Data Wrangler	Studio Notebooks	One-click Training	One-click Deployment
Processing	Built-in and Bring-your-own Algorithms	Experiments	Multi-Model Endpoints
Feature Store	Autopilot	Automatic Model Tuning	Model Monitor
Clarify	JumpStart	Debugger	Pipelines
		Managed Spot Training	Post-processing Job

allows you to train machine learning models in the Cloud.

Ground Truth AWS Marketplace Inference SageMaker Edge Manager

Amazon SageMaker >

Amazon SageMaker Studio

Dashboard Search Images

- ▶ Ground Truth
- ▶ Notebook **Notebook instances**
- Lifecycle configurations
- Git repositories
- ▶ Processing
- ▶ Training
- ▶ Inference
- ▶ Edge Manager
- ▶ Augmented AI
- ▶ AWS Marketplace

Amazon SageMaker > Notebook instances

Notebook instances

Create notebook instance

Search notebook instances

Name	Instance	Creation time	Status	Actions
dl-challenge	ml.t2.medium	Apr 24, 2020 07:14 UTC	InService	Open Jupyter Open JupyterLab

uma instância de Jupyter Notebook → abra aqui

This notebook demonstrates how to leverage transfer learning to use your own image dataset to build and train an image classification model using MXNet and Amazon SageMaker.

We use, as an example, the creation of a trash classification model which, given some image, classifies it into one of three classes: compost, landfill, recycle. This is based on the [Show Before You Throw](#) project from an AWS DeepLens hackathon and the [Smart Recycle Arm](#) project presented at the AWS Public Sector Summit 2019.

1. [Prerequisites](#)
2. [Download Data](#)
3. [Fine-tuning the Image Classification Model](#)
4. [Start the Training](#)
5. [Test your Model](#)
6. [Deploy your Model to AWS DeepLens](#)

Prerequisites

- Amazon Sagemaker notebook should have internet access to download images needed for testing this notebook. This is turned ON by default. To explore options review this link : [Sagemaker routing options](#)
- The IAM role assigned to this notebook should have permissions to create a bucket (if it does not exist)
 - [IAM role for Amazon Sagemaker](#)
 - [S3 create bucket permissions](#)

Now you have your Jupyter Notebook up and running.

Permissions and environment variables

Here we set up the linkage and authentication to AWS services. There are 2 parts to this:

- The roles used to give learning and hosting access to your data. This will automatically be obtained from the role used to start the notebook
- The Amazon sagemaker image classification docker image which need not be changed

Permissions and environment variables

Here we set up the linkage and authentication to AWS services. There are 2 parts to this:

- The roles used to give learning and hosting access to your data. This will automatically be obtained from the role used to start the notebook
- The Amazon sagemaker image classification docker image which need not be changed

```
In [53]: import os
import urllib.request
import boto3, botocore

import sagemaker
from sagemaker import get_execution_role

import mxnet as mx
mxnet_path = mx.__file__[:mx.__file__.rfind('/')]
print(mxnet_path)

role = get_execution_role()
print(role)

sess = sagemaker.Session()

/home/ec2-user/anaconda3/envs/mxnet_p36/lib/python3.6/site-packages/mxnet
arn:aws:iam::695958022619:role/service-role/AmazonSageMaker-ExecutionRole-20200424T001490
```

In order to execute a code cell,

Amazon S3 bucket info

Amazon S3 bucket info

Enter your Amazon S3 Bucket name where your data will be stored, make sure that your SageMaker notebook has access to this S3 Bucket by granting `S3FullAccess` in the SageMaker role attached to this instance. See [here](#) for more info.

DeepLens-compatible buckets must start with `deeplens`

```
In [54]: BUCKET = 'deeplens-phun-demo'
PREFIX = 'deeplens-trash'
```

```
In [55]: from sagemaker.amazon.amazon_estimator import get_image_uri
training_image = get_image_uri(sess.boto_region_name, 'image-classification', repo_version="latest")
print (training_image)
```

The method `get_image_uri` has been renamed in `sagemaker>=2`.
See: <https://sagemaker.readthedocs.io/en/stable/v2.html> for details.

Defaulting to the only supported framework/algorithm version: 1. Ignoring framework/algorithm version: latest.

```
811284229777.dkr.ecr.us-east-1.amazonaws.com/image-classification:1
```

Prepare data

It is assumed that your custom dataset's images are present in an S3 bucket and that different classes are separated by named folders, as shown in the following directory structure:

```
|-deeplens-bucket  
  |-deeplens-trash  
    |-images  
      |-Compost  
      |-Landfill  
      |-Recycle
```

Since we are providing the data for you in this example, first we'll download the example data, unzip it and upload it to your bucket.

```
In [*]: !wget https://deeplens-public.s3.amazonaws.com/samples/deeplens-trash/trash-images.zip  
In [57]: !rm -rf data/ && mkdir -p data  
!mkdir -p data/images  
!unzip -qq trash-images.zip -d data/images  
!rm trash-images.zip
```

Fine-tuning the Image Classification Model

Now that we are done with all the setup that is needed, we are ready to train our trash detector. To begin, let us create a `sageMaker.estimator.Estimator` object. This estimator will launch the training job.

Training parameters

There are two kinds of parameters that need to be set for training. The first one are the parameters for the training job. These include:

- **Training instance count:** This is the number of instances on which to run the training. When the number of instances is greater than one, then the image classification algorithm will run in distributed settings.
- **Training instance type:** This indicates the type of machine on which to run the training. Typically, we use GPU instances for these training
- **Output path:** This is the s3 folder in which the training output is stored

```
In [15]: s3_output_location = 's3://{}{}/output'.format(BUCKET, PREFIX)  
ic = sagemaker.estimator.Estimator(training_image,  
                                     role,  
                                     train_instance_count=1,  
                                     train_instance_type='ml.p2.xlarge',  
                                     train_volume_size = 50,  
                                     train_max_run = 360000,  
                                     input_mode='File',  
                                     output_path=s3_output_location,  
                                     sagemaker_session=sess,  
                                     base_job_name='ic-trash')
```

and how many computers do we need to train the model?

See: <https://sagemaker.readthedocs.io/en/stable/v2.html> for details.
train_instance_count has been renamed in sagemaker>2.
See: <https://sagemaker.readthedocs.io/en/stable/v2.html> for details.
train_volume_size has been renamed in sagemaker>2.
See: <https://sagemaker.readthedocs.io/en/stable/v2.html> for details.

Deploy to a Sagemaker endpoint

After training your model is complete, you can test your model by asking it to predict the class of a sample trash image that the model has not seen before. This step is called inference.

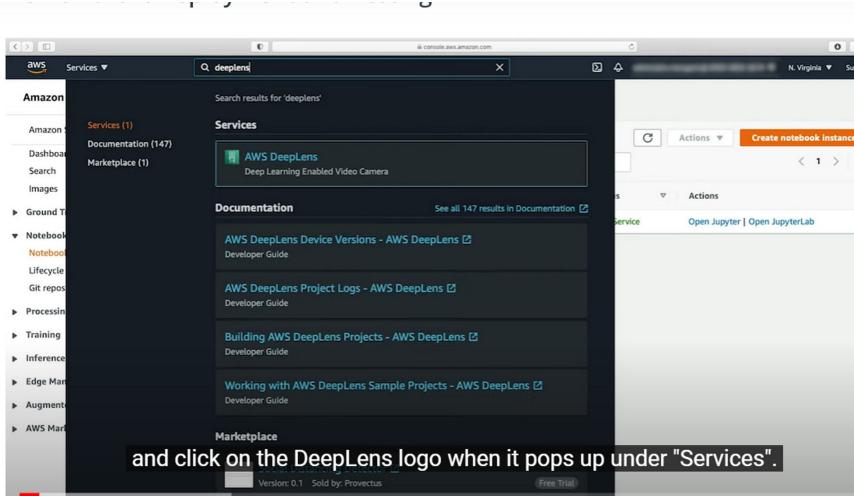
Amazon SageMaker provides an HTTPS endpoint where your machine learning model is available to provide inferences. For more information see the [Amazon SageMaker documentation](#).

```
In [23]: ic_infer = ic.deploy(initial_instance_count=1, instance_type='ml.t2.medium')  
-----!
```



Demo Part 2: Deployment and Testing

<https://youtu.be/sy33KRapsWE>

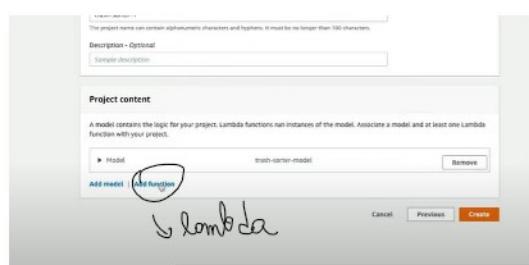
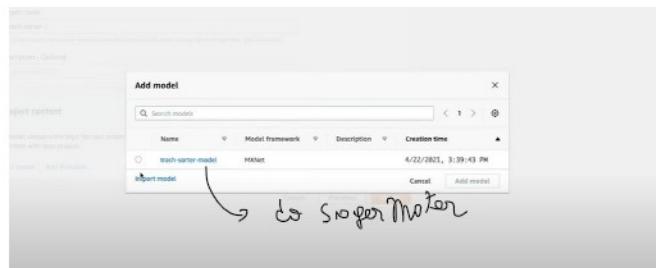
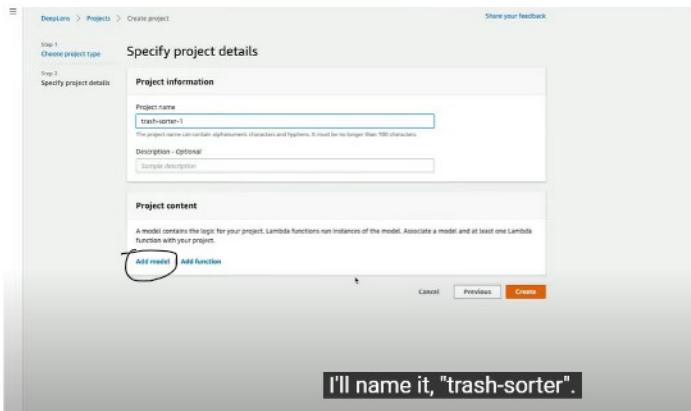


Demo Part 2: Deployment and Testing

As you can see, I'm going to

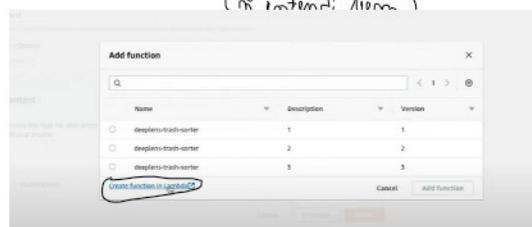
As you can see, I'm going to

As you can see, I'm going to



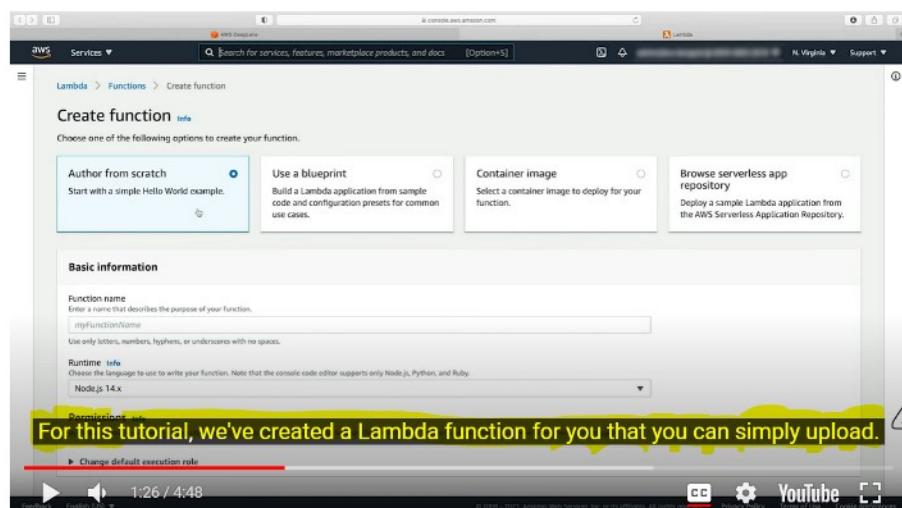
Lambda function hosts the code that will be running inference on AWS DeepLens.

(⚠️ intended! Alarm)



Next, you used AWS Lambda to deploy your model onto an AWS DeepLens device. Finally, once your model has been deployed to your device, you can use AWS IoT Greengrass to view the inference output from your model actively running on your AWS DeepLens device.

For more information, see <https://classroom.udacity.com/nanodevices/n005j/part/sa8c41f8cc480d-9f00-582735a7053/mod/ex/8830116b-2ca3-453e-8d11-ea41b436b5da/lessons/8079bd0c-6a77-40c-8f96-b669c36a6103/concepts/50185815-e148-d2e0-a709-bff908026ed2#>.



Next, choose the project and click on Deploy to device to deploy the project to the device.

Projects (2)

Name	Description	Version	Creation time	Last updated
trash-sorter-1		0	4/29/2021, 3:42:08 PM	4/29/2021, 3:42:08 PM
trash-sorter		2	4/22/2021, 3:48:03 PM	4/22/2021, 5:01:23 PM

après de importer, nota

Deploy to device

Devices (1)

Name	Project	Registration status	Device status	Creation time
phud11	trash-sorter	Registered	10.0.0.200	4/19/2021, 4:48:07 PM

↳ selectionne le device et Review

Review and deploy

Deployment check

AWS DeepLens will deploy the project below to your device. Choose Deploy to continue.

New project: trash-sorter-1

Type	Name
Function	deepsensor-trash-sorter/versions/0
Model	trash-sorter-model

Current project: trash-sorter

Type	Name
Function	deepsensor-trash-sorter/versions/2
Model	trash-sorter-model

⚠ There is an existing project on this device. Do you want to replace it?

If you Deploy, AWS DeepLens will remove the current project before deploying the new project.

Deployment will incur costs

AWS DeepLens uses various services and resources to run your projects. Deployment costs are based on usage and may vary.

hit "Review", and finally click "Deploy"

2:15 / 4:48

Deploy

Deployment of project trash-sorter-1, version 0 succeeded.

4/29/2021, 3:44:43 PM Click on "View project stream" for instructions on how to view the filtered or transformed AWS DeepLens output.

phud11

Device status

Registration status	Device status	Software Version
Registered	Online	1.4.9

Current project

Name	Description	Version
trash-sorter-1	-	-
▶ Project content		

Project output

You can use AWS IoT console to view a JSON-formatted output of the project deployed to your AWS DeepLens device. [info](#)

To view MQTT messages:

1. Copy the topic that's unique to your registered device: [aws/things/DeepLens_14A1W0W1n7qjebz8Sp5aA/info](#) [Copy](#)
2. Go to [AWS IoT Core](#) in the AWS CloudWatch Metrics console and subscribe to the topic.

Now that your project has deployed to DeepLens,

USB connection troubleshooting

Device details presented here help you with the following:

- Verify whether your AWS DeepLens device is successfully registered and connected to the AWS cloud.
- Inspect the current software version.
- Check whether an update is available and choose to install it.
- Get your device's local IP address using the AWS IoT Device Defender tool. This is useful to log in to the device using the ssh -i cert.pem -R 2200 command in an SSH session.
- Get the MQTT topic ID, which you can use to publish messages to the topic in order to receive messages that are published by the inferences Lambda function for the project that is deployed to the device.

In addition, in **Device settings** you can access the AWS Greengrass system logs or the AWS Lambda Lambda function log with a single click.

Video streaming

1. Open your **Device settings** and follow the instructions to install the streaming certificate.

2. Then, click on **View video stream**.

View video stream *Your AWS DeepLens needs to be online to view the stream*

You can also view your device's video on [aws/things/DeepLens_14A1W0W1n7qjebz8Sp5aA/video](#)

you can click "View video stream"

