20/05/2022

CAPSTONE REPORT

Inventory Monitoring at Distribution Centers

AWS Machine Learning Engineer Nanodegree Program

UDACITY

Table of Contents

Introduction.

In this capstone project proposal, prior to completing the following Capstone Project, we will leverage what we have learned throughout the Nanodegree program to author a proposal for solving a problem of our choice by applying machine learning algorithms and techniques. A project proposal encompasses seven key points:

- The project's **domain background** — the field of research where the project is derived.
- A **problem statement** — a problem being investigated for which a solution will be defined.
- The **datasets and inputs** — data or inputs being used for the problem.
- A **solution statement** — the solution proposed for the problem given.
- A **benchmark model** — some simple or historical model or result to compare the defined solution to.
- A **set of evaluation metrics** — functional representations for how the solution can be measured.
- An outline of the **project design** — how the solution will be developed, and results obtained

1 Domain Background

Monitoring inventory means developing an understanding of what your best-selling products are, how often you turn over that inventory and what you need to have on hand to satisfy orders. Having the right products in stock allows you to fulfill orders in a timely manner and develop a loyal customer base.

Inventory monitoring and management is vital to a company's health because it helps make sure there is rarely too much or too little stock on hand or in a particular bin, limiting the risk of stockouts and inaccurate records.

There is an old business axiom that says, “**Nothing happens until somebody sells something**." With inventory monitoring, which could be changed to "**Nothing happens until somebody counts something**."

Inventory can be anything from boxes of ice cream cones in the storeroom at a sweet shop to a million-square-foot warehouse full of goods for a big box chain. But in either case, accurate inventory monitoring is a key to that company's success.

Simply put, inventory is the goods that a business owns that it plans to sell. If your company is an apparel retailer, products become inventory when you take possession of shirts, dresses, suits, and accessories from your suppliers. Those products leave the stock when they are sold to customers. Inventory can be stored on premises or at warehouses, distribution centers and other facilities.

2 Problem Statement

Keeping track of your Inventory is fundamental to your success in retail. At the most basic level your job is to supply the products to meet consumer demand. You cannot do that without effective inventory management.

Inventory systems suffer from several problems. Some of the more common issues include stock outs, excess inventory, misplaced inventory, and employee errors

In other hands, manual Inventory Management requires a huge workforce, and it is also error prone. So, there is a need for an effective inventory management.

The problem faced by the company is they do not have (or at least they do not have yet) any systematic system to record and keep their inventory data. It is difficult for the administrator to record the inventory data quickly and safely because they only keep it in the register and not meticulously organized. Therefore, it is difficult for the admin to estimate their profit.

With the new system developed, companies can manage their inventory data easily, quickly, and more secured. We could solve the above problem with machine automated tasks, Computer Vision Process aids us in solving the defined problem.

3 Solution Statement

Our solution to this problem is to create for example a responsive web application (also viewable on mobile) that would give real-time updates like real time update to hospital room supplies. Using video monitoring of the shelves, machine learning algorithms would recognize the types and quantities of available supplies.

For the sake of simplicity, we will (in this project) start by collecting data.

Our solution consisted of an automated object detection machine learning algorithm recognizing supplies and their quantities plus a responsive front-end. The user (for example) or someone else can click on an image to view the live photo feed of the supplies, as well as a table of its supplies and quantities.

One of the areas where computer vision has made huge progress is image classification and object detection. A neural network trained on enough labeled data will be able to detect and highlight a wide range of objects with impressive accuracy.

To do this, thousands of images of objects needed to be generated in some sort of container.

4 Datasets and Inputs

In order to train a computer vision system for inventory monitoring, a method for creating a dataset is required. Those method are sometimes quite enough difficult to obtain.

Hopefully, Amazon provides us the **Amazon Bin Image Dataset** that we can use to train, test and validate our model.

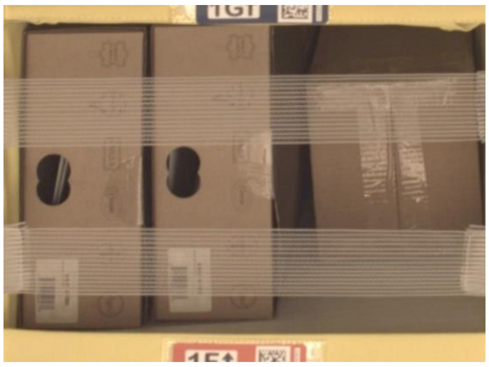
4.1 Description of the Image Dataset

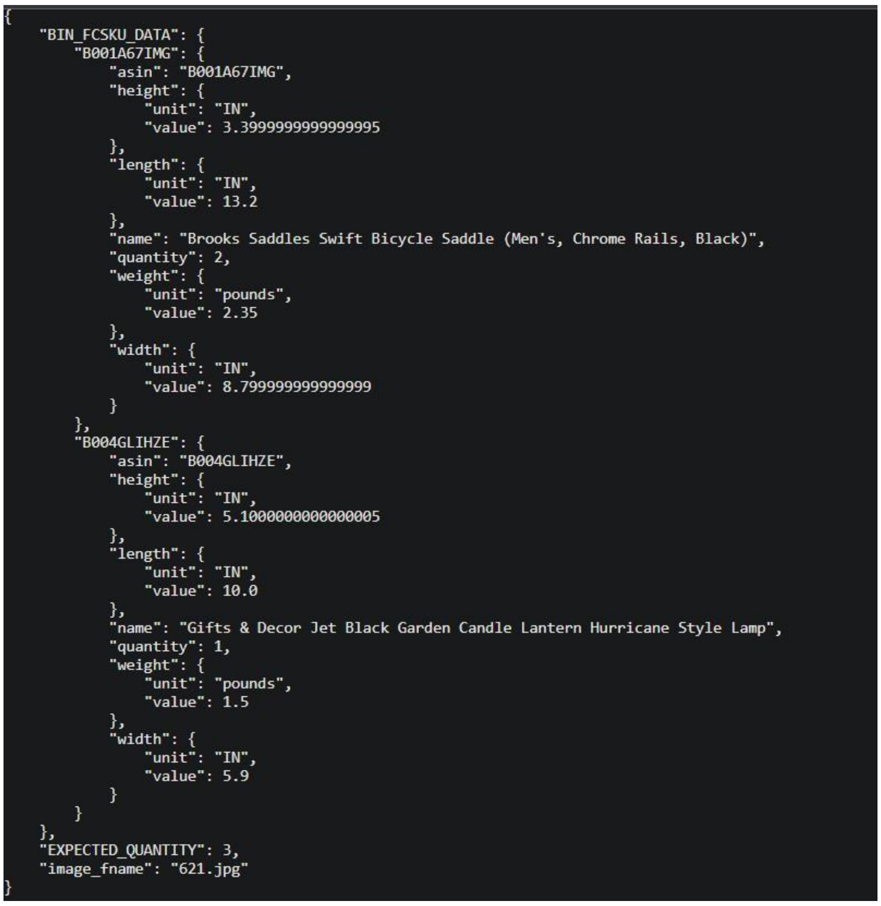
The **Amazon Bin Image Dataset** contains over 500,000 images and metadata from bins of a pod in an operating Amazon Fulfillment Center. The bin images in this dataset are captured as robot units carry pods as part of normal Amazon Fulfillment Center operations. Bin images dataset provides the metadata for each image from where number of items can be derived.

These are some typical images in the dataset. A bin contains multiple object categories and various number of instances. The corresponding metadata exist for each bin image, and it includes the object category identification (Amazon Standard Identification Number, ASIN), quantity, size of objects, weights.

Images are extracted from the source (provides by Amazon) which are available in JPEG format and the target or label data is extracted from the corresponding JSON file.

- Each image contains a different number of objects categories, this number is also called **EXPECTED_QUANTITY**.
- For each object categories, denote by a unique identifier category(**“asin”**), field **quantity** is set to 1.



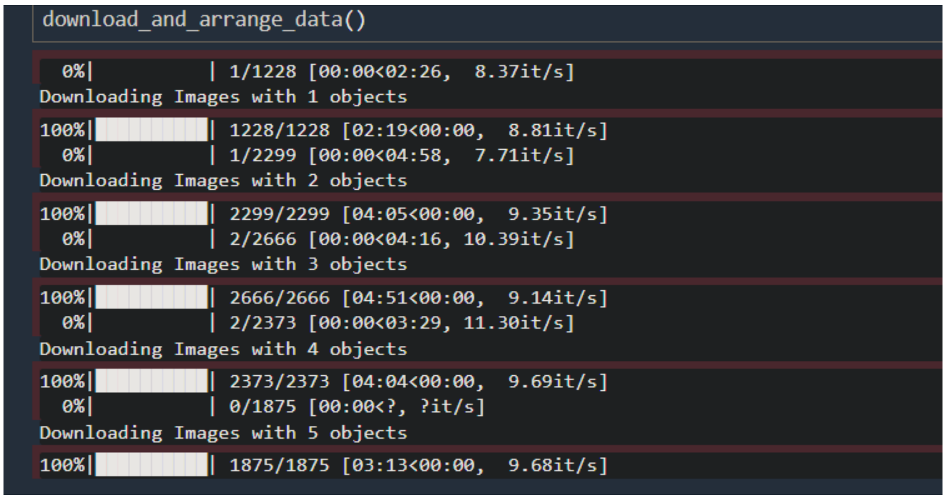


```
{
    "BIN_FCSKU_DATA": {
        "B001A67IMG": {
            "asin": "B001A67IMG",
            "height": {
                "unit": "IN",
                "value": 3.3999999999999995
            },
            "length": {
                "unit": "IN",
                "value": 13.2
            },
            "name": "Brooks Saddles Swift Bicycle Saddle (Men's, Chrome Rails, Black)",
            "quantity": 2,
            "weight": {
                "unit": "pounds",
                "value": 2.35
            },
            "width": {
                "unit": "IN",
                "value": 8.799999999999999
            }
        },
        "B004GLIHZE": {
            "asin": "B004GLIHZE",
            "height": {
                "unit": "IN",
                "value": 5.1000000000000005
            },
            "length": {
                "unit": "IN",
                "value": 10.0
            },
            "name": "Gifts & Decor Jet Black Garden Candle Lantern Hurricane Style Lamp",
            "quantity": 1,
            "weight": {
                "unit": "pounds",
                "value": 1.5
            },
            "width": {
                "unit": "IN",
                "value": 5.9
            }
        }
    },
    "EXPECTED_QUANTITY": 3,
    "image_fname": "621.jpg"
}
```

Here is an example of image and its corresponding metadata.

- Image contains 3 different objects categories denote by "**EXPECTED_QUANTITY": 3**
- Those categories are:
    - **Asin=B001A67IMG with quantity=2**
    - **Asin=B004GLIHZE with quantity=1**

Each categories have an **asin, height, length, name, quantity, weight, and width**. Before digging deeper into our analysis, we first download and plot dataset distribution according to label.



```
download_and_arrange_data()

  0%|          | 1/1228 [00:00<02:26,  8.37it/s]
Downloading Images with 1 objects
100%|██████████| 1228/1228 [02:19<00:00,  8.81it/s]
  0%|          | 1/2299 [00:00<04:58,  7.71it/s]
Downloading Images with 2 objects
100%|██████████| 2299/2299 [04:05<00:00,  9.35it/s]
  0%|          | 2/2666 [00:00<04:16, 10.39it/s]
Downloading Images with 3 objects
100%|██████████| 2666/2666 [04:51<00:00,  9.14it/s]
  0%|          | 2/2373 [00:00<03:29, 11.30it/s]
Downloading Images with 4 objects
100%|██████████| 2373/2373 [04:04<00:00,  9.69it/s]
  0%|          | 0/1875 [00:00<?, ?it/s]
Downloading Images with 5 objects
100%|██████████| 1875/1875 [03:13<00:00,  9.68it/s]
```

Figure 1 - Download dataset

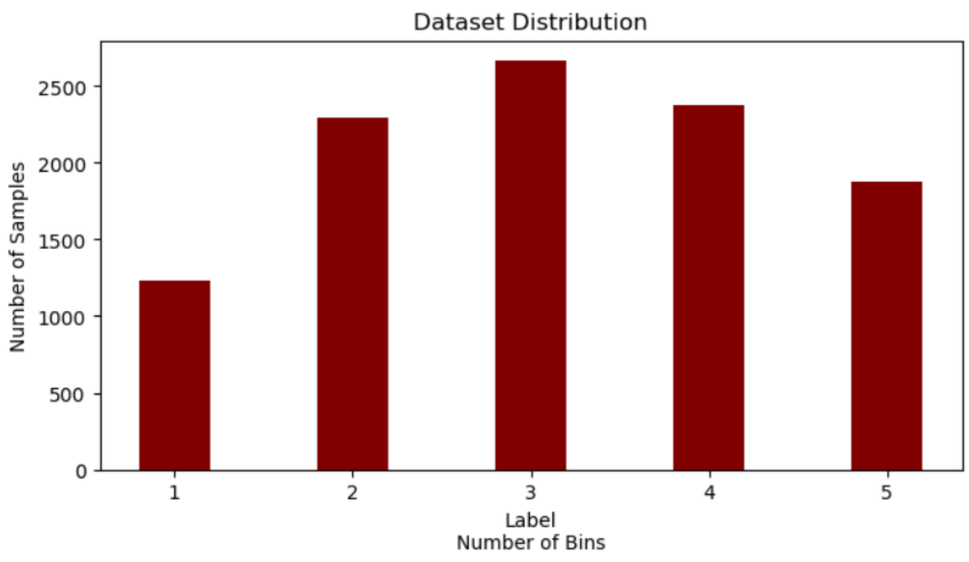


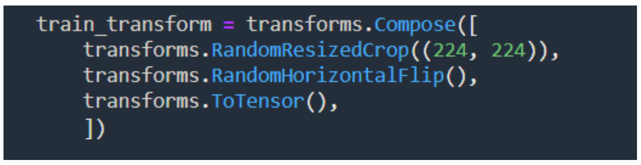
Figure 2 - Number of samples vs. Label

We could find a slight imbalance in the no of samples used for label – 1. As for the futuristic step the samples for label – 1 could be up samples.

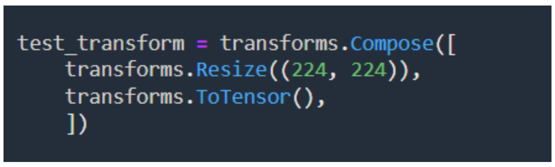
4.2 Data processing

All images are download from Github and then resized to fit the input shape of our resnet50 model.

Images are first cropped, flipped, and then converted to Tensor. All these steps were done using PyTorch transformers.

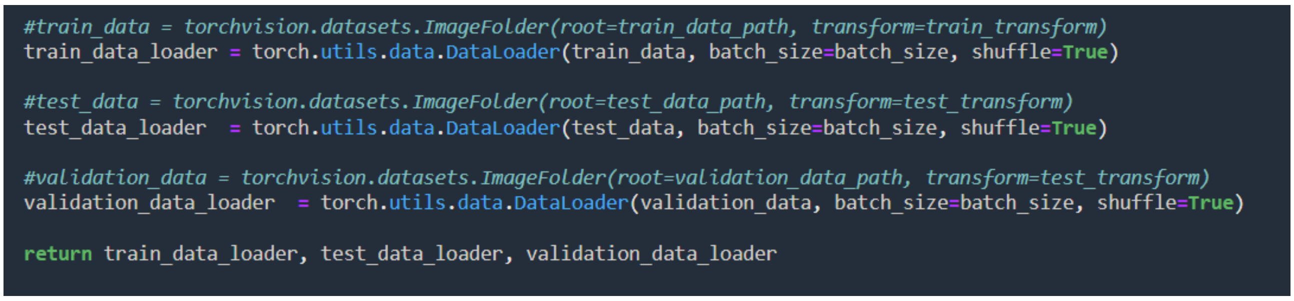


```
train_transform = transforms.Compose([
    transforms.RandomResizedCrop((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    ])
```



```
test_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    ])
```

Now we need to use an **ImageFolder** Class, who’s a data loader class in **torchvision** that helps us load our own image dataset.



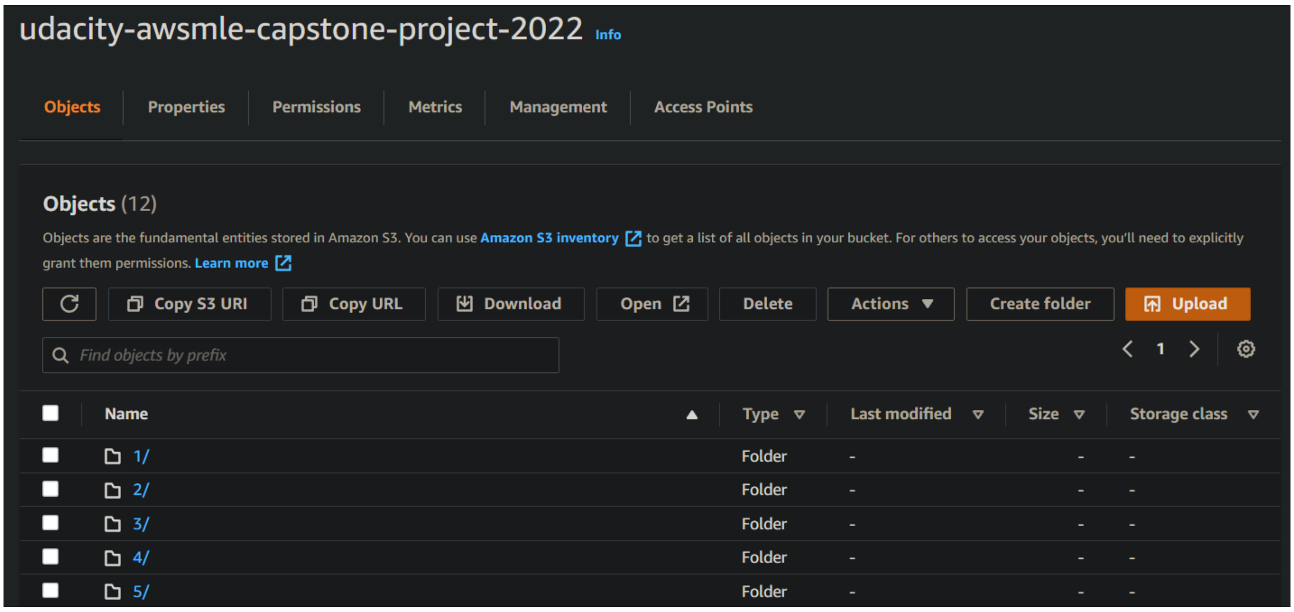
```
#train_data = torchvision.datasets.ImageFolder(root=train_data_path, transform=train_transform)
train_data_loader = torch.utils.data.DataLoader(train_data, batch_size=batch_size, shuffle=True)

#test_data = torchvision.datasets.ImageFolder(root=test_data_path, transform=test_transform)
test_data_loader  = torch.utils.data.DataLoader(test_data, batch_size=batch_size, shuffle=True)

#validation_data = torchvision.datasets.ImageFolder(root=validation_data_path, transform=test_transform)
validation_data_loader  = torch.utils.data.DataLoader(validation_data, batch_size=batch_size, shuffle=True)

return train_data_loader, test_data_loader, validation_data_loader
```

4.3 Algorithms

Many convolutions neural network algorithms can be used to solve this kind of problem, but for this project we focus on Resnet50.

ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from our dataset. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224.

As we will use Amazon Sagemaker, all images were downloaded on Amazon S3.


udacity-awsmle-capstone-project-2022 Info

Objects (12)

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

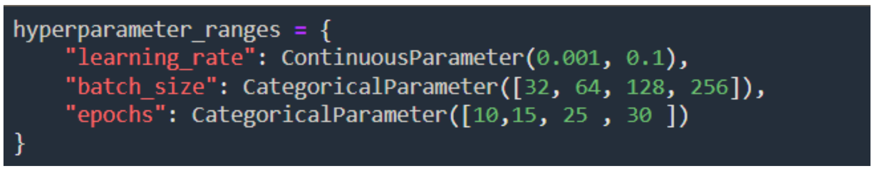
Name	Type	Last modified	Size	Storage class
1/	Folder	-	-	-
2/	Folder	-	-	-
3/	Folder	-	-	-
4/	Folder	-	-	-
5/	Folder	-	-	-

Resnet50 model have a bunch of parameters, so we have to do an Hyperparameter tunning to choose the best one.

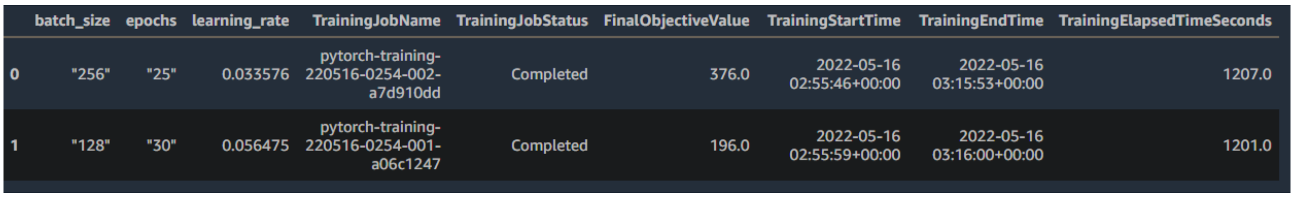
Hyper parameter tuning (optimization) is an essential aspect of machine learning process. A good choice of hyperparameters can really make a model succeed in meeting desired metric value or on the contrary it can lead to an unending cycle of continuous training and optimization.

4.4 Hyperparameters

For our Resnet50 model, we focused on this set of hy^perparameters;

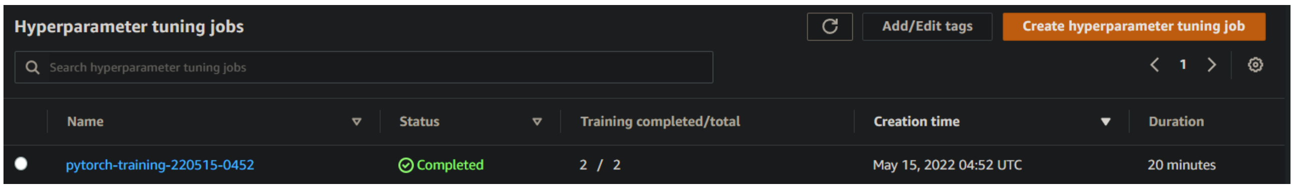

```
hyperparameter_ranges = {
    "learning_rate": ContinuousParameter(0.001, 0.1),
    "batch_size": CategoricalParameter([32, 64, 128, 256]),
    "epochs": CategoricalParameter([10,15, 25 , 30 ])
}
```

Our best model is now:



	batch_size	epochs	learning_rate	TrainingJobName	TrainingJobStatus	FinalObjectiveValue	TrainingStartTime	TrainingEndTime	TrainingElapsedTimeSeconds
0	"256"	"25"	0.033576	pytorch-training-220516-0254-002-a7d910dd	Completed	376.0	2022-05-16 02:55:46+00:00	2022-05-16 03:15:53+00:00	1207.0
1	"128"	"30"	0.056475	pytorch-training-220516-0254-001-a06c1247	Completed	196.0	2022-05-16 02:55:59+00:00	2022-05-16 03:16:00+00:00	1201.0

Figure 3 - Hyperparameter jiob


Hyperparameter tuning jobs
Add/Edit tags
Create hyperparameter tuning job
Search hyperparameter tuning jobs
1

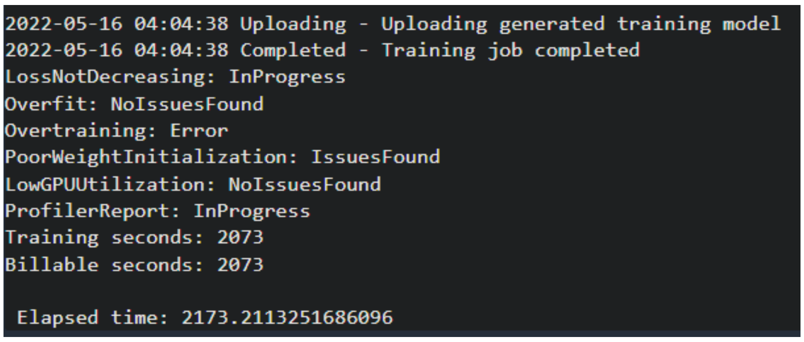
Name	Status	Training completed/total	Creation time	Duration
pytorch-training-220515-0452	Completed	2 / 2	May 15, 2022 04:52 UTC	20 minutes

Figure 4- Hyperparameter job in Sagemaker

4.5 Model Profiling and Debugging

We then proceed to model to model debugging and profiling to better monitor and debug your model training job.

SageMaker Debugger auto generated report. You can generate such reports on all supported training jobs. The report provides summary of training job, system resource usage statistics, framework metrics, rules summary, and detailed analysis from each rule.



```
2022-05-16 04:04:38 Uploading - Uploading generated training model
2022-05-16 04:04:38 Completed - Training job completed
LossNotDecreasing: InProgress
Overfit: NoIssuesFound
Overtraining: Error
PoorWeightInitialization: IssuesFound
LowGPUUtilization: NoIssuesFound
ProfilerReport: InProgress
Training seconds: 2073
Billable seconds: 2073

 Elapsed time: 2173.2113251686096
```

Figure 5- Model profiling

4.6 Model Evaluation

Hyperparameter tunning was performed to find out the best hyperparameters to train the model.

The validation loss of the training process remains almost constant and there are frequent highs and lows, when compared with the training loss, it seems to fit along with it as shown below.

The accuracy of the benchmark model chosen is 56 % (Approx), the experiment model didn't achieve the results of the Benchmark. So as next steps, must work more on the data and its transformation. Maybe doing some data augmentation or transfer learning.

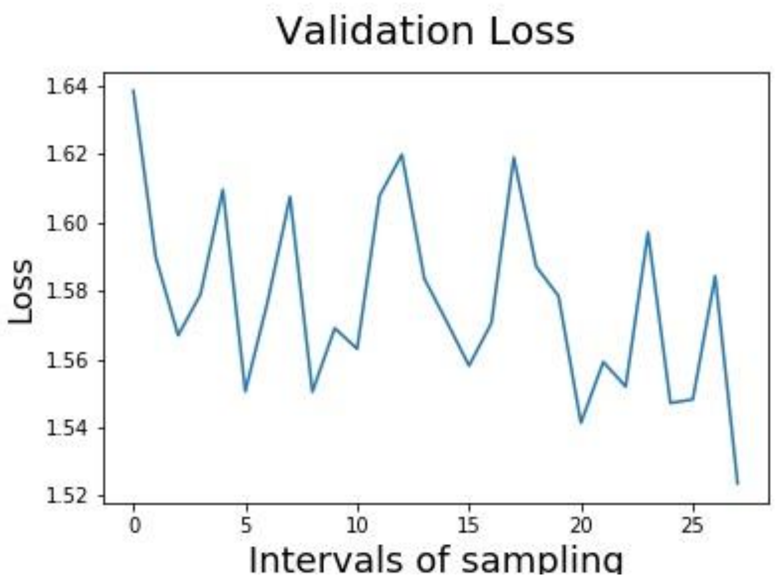


Figure 6- Validation Loss

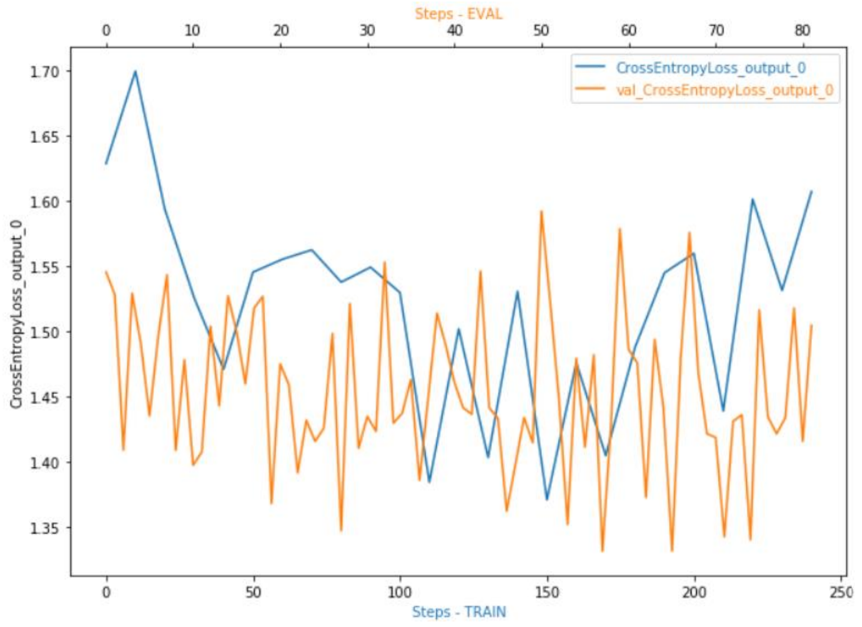
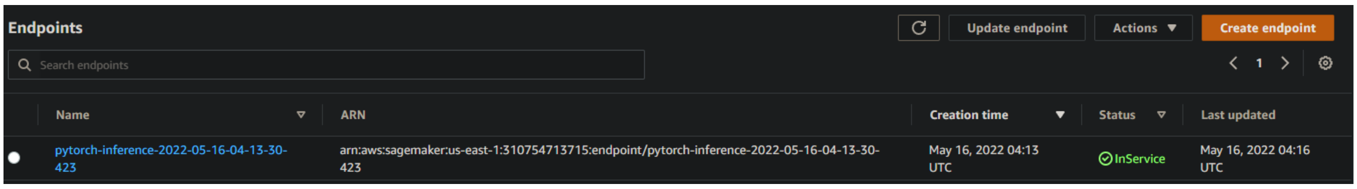


Figure 7-CrossEntropy Loss for Train And Eval

4.7 Model Deploying

Model was deployed for inference.



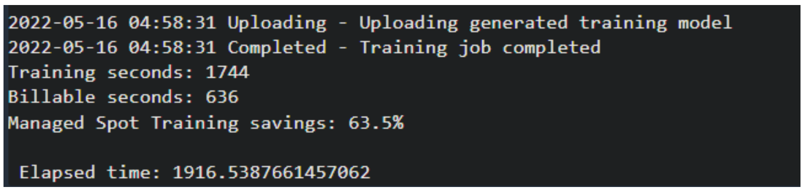
	Name	ARN	Creation time	Status	Last updated
○	pytorch-inference-2022-05-16-04-13-30-423	arn:aws:sagemaker:us-east-1:310754713715:endpoint/pytorch-inference-2022-05-16-04-13-30-423	May 16, 2022 04:13 UTC	✓ InService	May 16, 2022 04:16 UTC

Figure 8- Model Endpoint

4.8 Cost Analysis

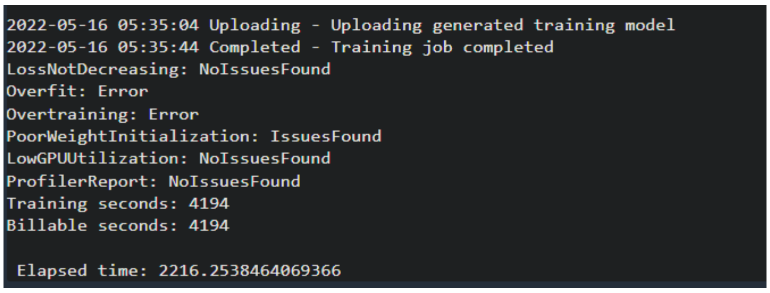
We have performed a cost analysis of our system and then use spot instances to lessen our model training cost. To use spot instance, we have enabled the **use_spot_instances** var in our script. We have seen a notable difference between X (the actual compute-time your training job spent) and Y (the time you will be billed for after Spot discounting is applied.) signifying the cost savings you will get for having chosen Managed Spot Training. This should be reflected in an additional line:

- Managed Spot Training savings: $(1 - \frac{Y}{X})$*100 %



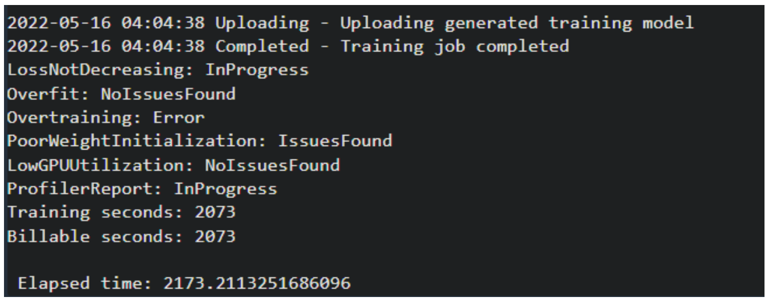
```
2022-05-16 04:58:31 Uploading - Uploading generated training model
2022-05-16 04:58:31 Completed - Training job completed
Training seconds: 1744
Billable seconds: 636
Managed Spot Training savings: 63.5%

 Elapsed time: 1916.5387661457062
```

- We have also train our model on multiple instances, to measure the costs



```
2022-05-16 05:35:04 Uploading - Uploading generated training model
2022-05-16 05:35:44 Completed - Training job completed
LossNotDecreasing: NoIssuesFound
Overfit: Error
Overtraining: Error
PoorWeightInitialization: IssuesFound
LowGPUUtilization: NoIssuesFound
ProfilerReport: NoIssuesFound
Training seconds: 4194
Billable seconds: 4194

 Elapsed time: 2216.2538464069366
```

- Compare to training our model in one instance.



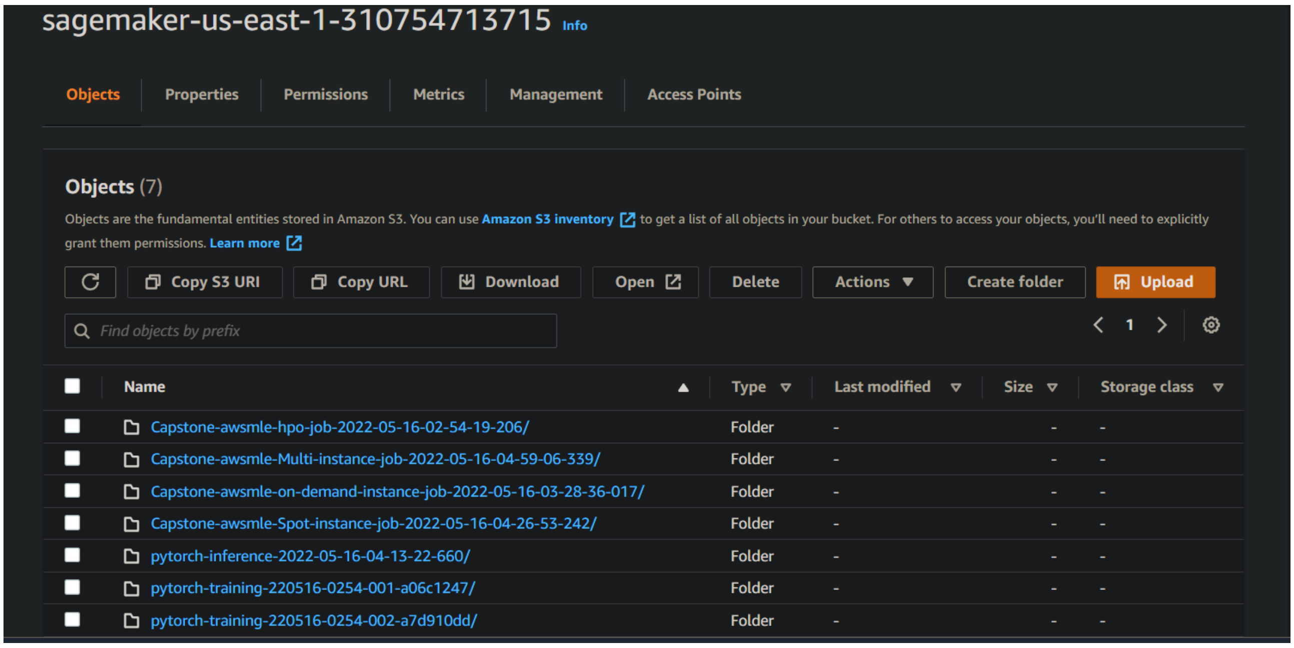
```
2022-05-16 04:04:38 Uploading - Uploading generated training model
2022-05-16 04:04:38 Completed - Training job completed
LossNotDecreasing: InProgress
Overfit: NoIssuesFound
Overtraining: Error
PoorWeightInitialization: IssuesFound
LowGPUUtilization: NoIssuesFound
ProfilerReport: InProgress
Training seconds: 2073
Billable seconds: 2073

 Elapsed time: 2173.2113251686096
```

4.9 Future steps

There are still much more to do:

- We've noticed that data are unbalanced, maybe we can introduce more data with label 1 to balance our dataset or doing some down sampling of other label.
- Increase our hyperparameter tunning by including another parameter.
- Implement Data Augmentation and transfer learning techniques.

- Simulating Spot interruption after a number of epochs.
- Continue training after Spot capacity is resumed



Objects (7)

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

Name	Type	Last modified	Size	Storage class
Capstone-awsmle-hpo-job-2022-05-16-02-54-19-206/	Folder	-	-	-
Capstone-awsmle-Multi-instance-job-2022-05-16-04-59-06-339/	Folder	-	-	-
Capstone-awsmle-on-demand-instance-job-2022-05-16-03-28-36-017/	Folder	-	-	-
Capstone-awsmle-Spot-instance-job-2022-05-16-04-26-53-242/	Folder	-	-	-
pytorch-inference-2022-05-16-04-13-22-660/	Folder	-	-	-
pytorch-training-220516-0254-001-a06c1247/	Folder	-	-	-
pytorch-training-220516-0254-002-a7d910dd/	Folder	-	-	-