Inventory Monitoring at Distribution Centers

Ramy Gendy

February 9th, 2023

Definition

Project Overview

Distribution centers often use robots to move objects as a part of their operations. Objects are carried in bins which can contain multiple objects. Since noone to check the number of objects in bin that may cause loss of some of the bin items which will result low quality of service and decrease profit. Have a way to check the count of those items and avoid losing them to ensure quality of service is good practice for any distribution center.

In this project, I built a machine learning model that can count the number of objects in each bin. A system like this can be used to track inventory and make sure that delivery consignments have the correct number of items.

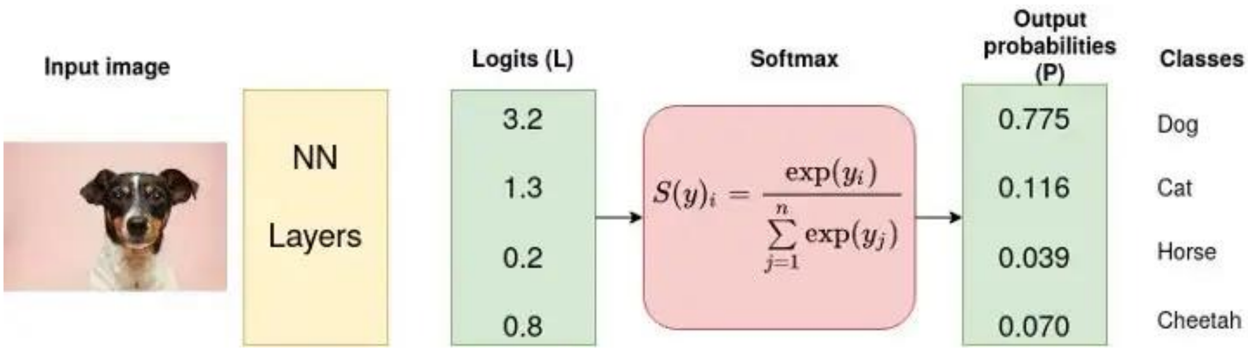
To finish this project, I used *AWS SageMaker* and its services with good machine learning engineering practices to fetch data from a database, preprocess it, then proceed to train, refine, evaluate and validate a machine learning model.

Problem Statement

Loss of objects at the distribution centers since robots doesn't notice the missing ones while moving them around as robots lack vision which states clearly our problem is related to *computer vision domain*. Where my implemented solution lies in training of machine learning model with computer vision capabilities to check bin and decide the count based on the number of items in it. For that I will need a good source of data to build the model based on it, for that the Amazon Bin Image Dataset which 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.

Metrics

After training my ML model I will use the standard *Cross Entropy Loss function* as my metric to study my training process results along with hyperparameter tuning and optimization for quality and improvements. The reason for this is that *Cross-Entropy* is to take the output probabilities (P) and measure the distance from the truth values (as shown in Figure below) since *Cross-entropy* builds upon the idea of entropy from information theory and calculates the number of bits required to represent or transmit an average event from one distribution compared to another distribution.



II. Analysis

Data Exploration

The Dataset contains over 500,000 images and metadata from bins of a pod in an operating Amazon Fulfillment Center. Images are located in the bin-images directory, and metadata for each image is located in the metadata directory. Images and their associated metadata share simple numerical unique identifiers. For example, the metadata for the image at [523.jpg](523.jpg) is found at 523.json. If you use the AWS Command Line Interface, you can list images in the bucket with the "ls" command:

aws s3 ls s3://aft-vbi-pds/bin-images/

To download data using the AWS Command Line Interface, you can use the "cp" command. For instance, the following command will copy the image named 523.jpg to your local directory:

aws s3 cp s3://aft-vbi-pds/bin-images/523.jpg 523.jpg

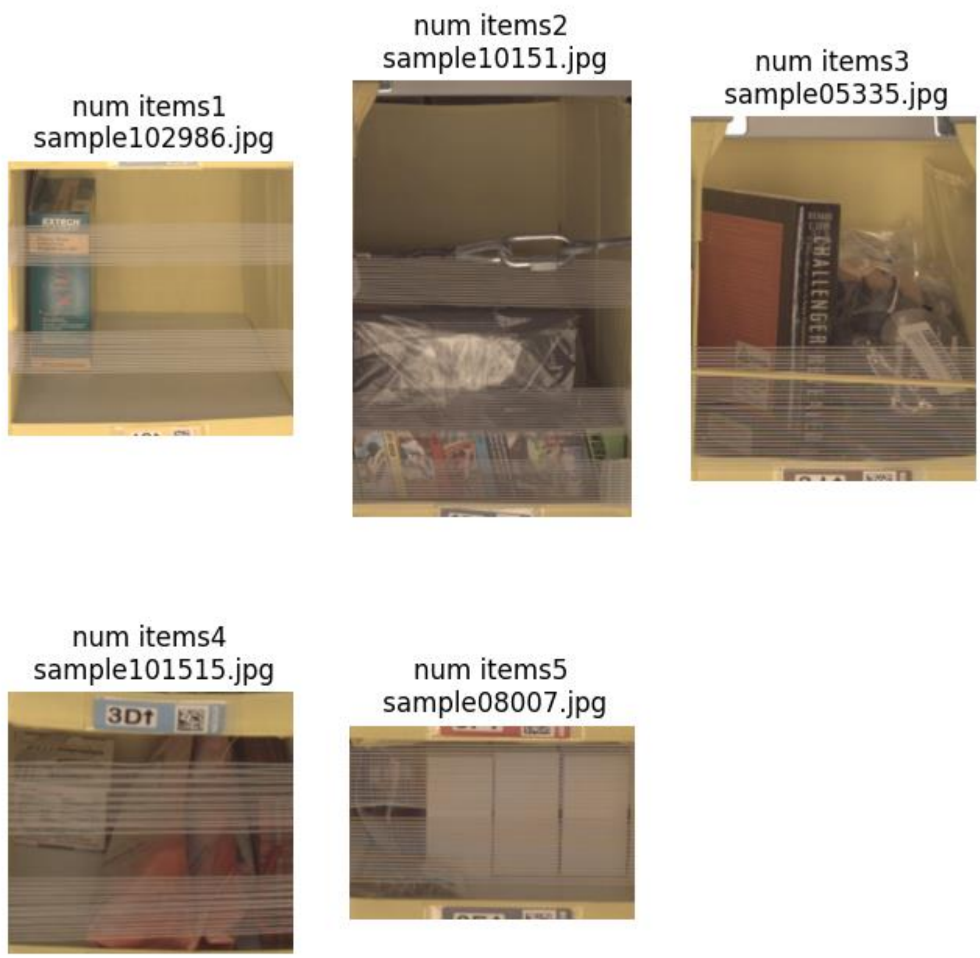
In order to limit training time and AWS costs a JSON file with list of 10441 images divided into 5 classes was used.

Each class has a number of items in range [1, 2, 3, 4, 5]:

- Class 1 with: 1228 images with 1 item in it.
- Class 2 with: 2299 images with 2 items in it.
- Class 3 with: 2666 images with 3 items in it.
- Class 4 with: 2373 images with 4 items in it.
- Class 5 with: 1875 images with 5 items in it.

Exploratory Visualization

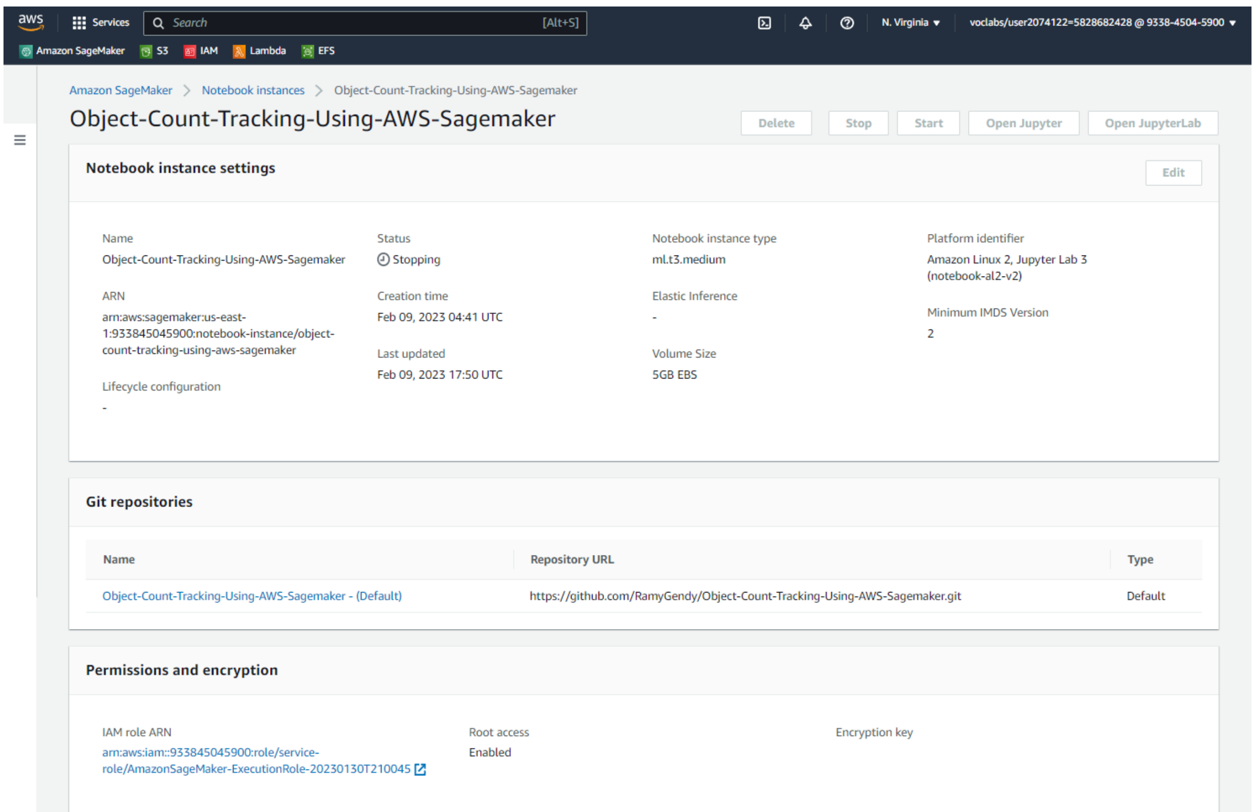
A visualization summary about the data is done by taking a random sample image from the dataset are presented in Figure below.



Algorithms and Techniques

For the algorithm used to complete this project is *ResNet50 CNN* it is the best to use pretrained image classification network, which already is able to process and classify some image data, because of similar data structures, high results and fast training time with *PyTorch* models. As for techniques was to train the model after finding the best hyperparameter values, which is done by optimizing the hyperparameters before creating a training job, all that in *AWS Sagemaker* platform.

For that we, I have created AWS Sagemaker notebook instance before starting working on the project with the following setup:



Benchmark

In the same domain, I found 2 contributions:

- Title: Amazon Bin Image Dataset Challenge:
  - URL: https://github.com/silverbottlep/abid_challenge
  - Author Name: silverbottlep
  - Author URL: https://github.com/silverbottlep

Title: Amazon Inventory Reconciliation using AI:

- URL: https://github.com/OneNow/AI-Inventory-Reconciliation
- Author Name: Pablo Rodriguez Bertorello, Sravan Sripada, Nutchapol Dendumrongsup
- Author URL: https://github.com/pablo-tech

The resulted accuracy of both contributions is 55% approximately with a RMSE of 0.94. I will be trying to challenge or provide similar accuracy.

III. Methodology

Data Preprocessing

Since I used AWS Sagemaker platform to complete this project, then my steps in data preparation and preprocessing based on my data exploration are:

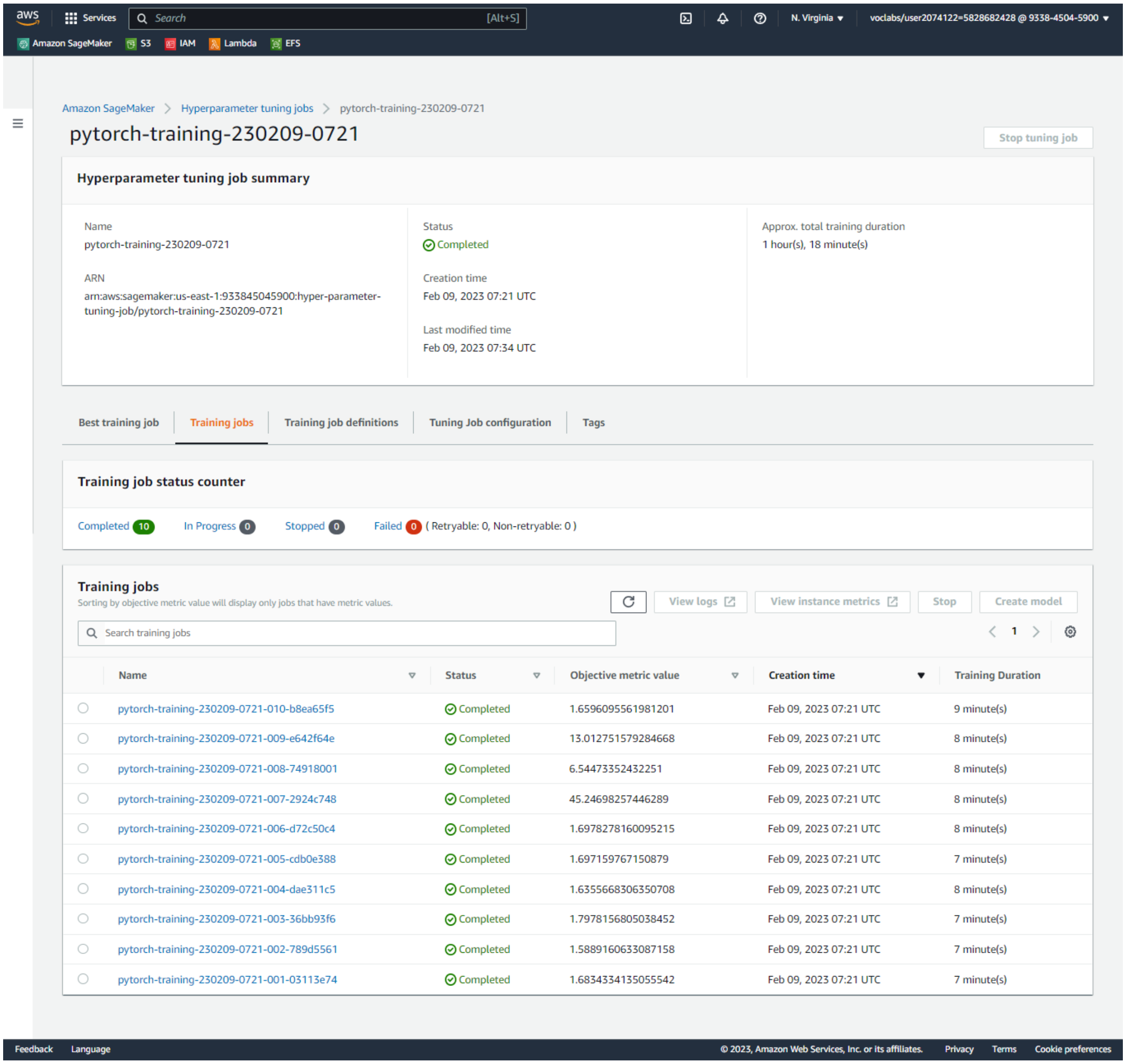
1. Fetch data from *Amazon* Database.
2. Select the data I will be working with through the help of *JSON* file.
3. Split the result into 3 subsets: Train, Test and Validation sets with ratios 60%, 20% and 20% respectively.
4. Upload the new dataset to *AWS S3 Bucket*.

Implementation

Once our data is uploaded, I started with hpo.py which executes just a single epoch on a part of training data before the real training job starts to find the best hyperparameters for the training job with the following ranges:

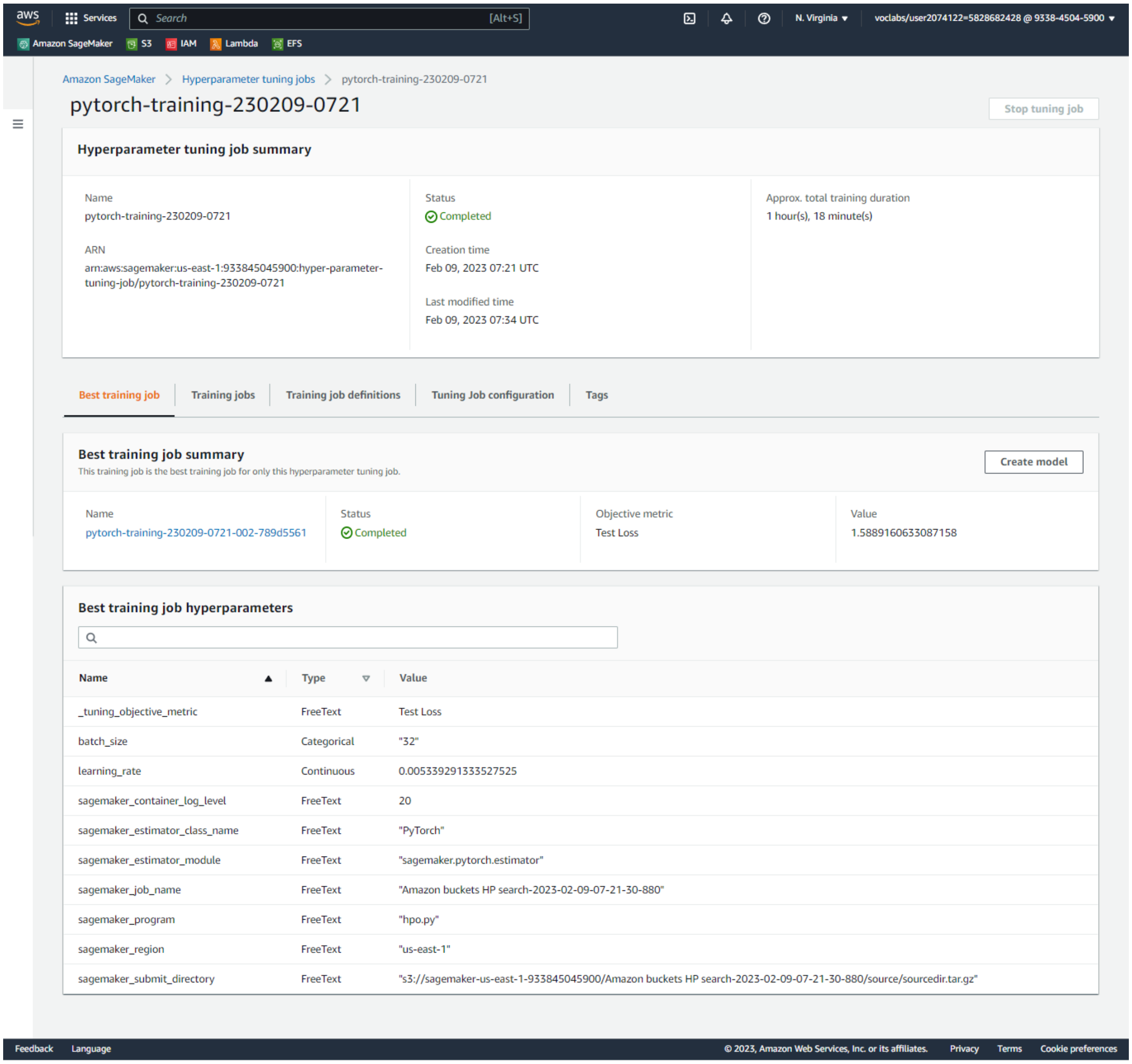
- ✓ Learning rate range: 0.001, 0.1
- ✓ Batch size values: 32, 64, 128, 256, 512

Hyperparameter tuning jobs:

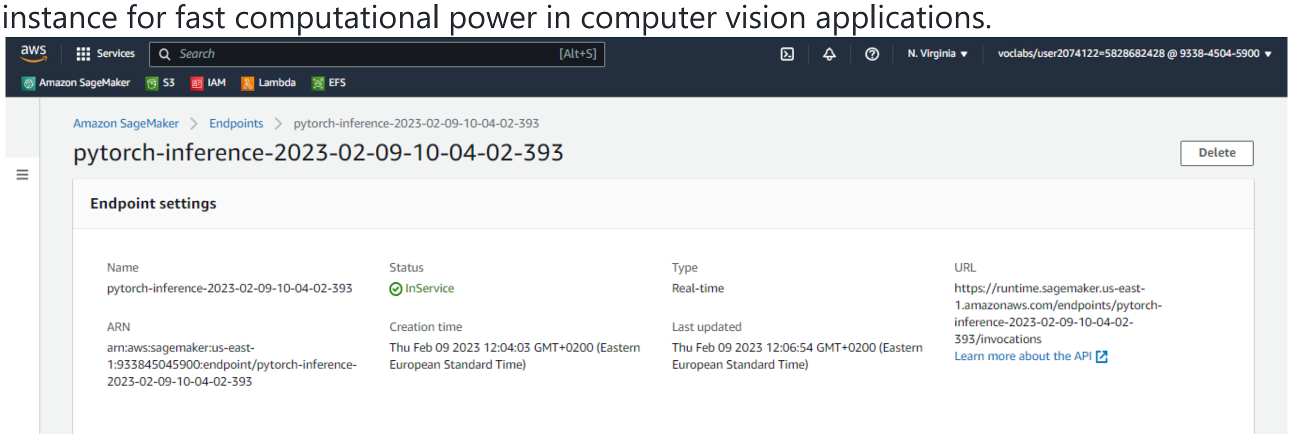


After execution, found the best hyperparameters are:

- ✓ Learning rate: 0.005339291333527525
- ✓ Batch size: 32

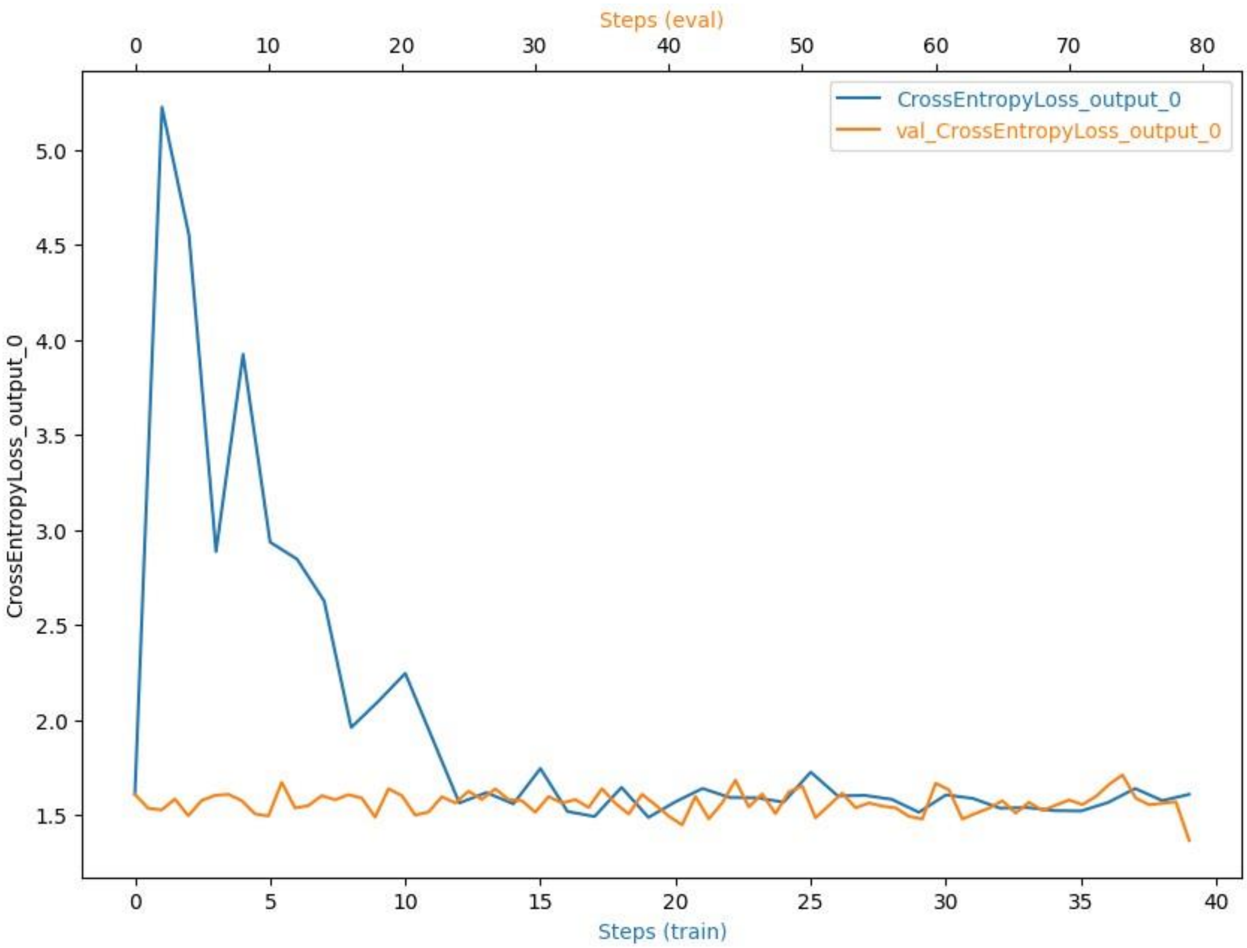


Note: hyperparameter tuning was performed to search the 2D parameter space. After that I started training my model with the best hyperparameters and deploying it to an endpoint on *AWS Sagemaker Endpoints*. Where I used "ml.m5.xlarge" as training instance for fast computational power in computer vision applications.



IV. Results

Model Evaluation and Validation



For the model inference deployment, I have created" inference.py" script, which can be used for creating service functions like *AWS Lambda or Sagemaker*. After deploying the model to the endpoint, I started evaluation it with my metrics (Cross Entropy Loss Function) as in the above figure.

Then tested the model operation on the test image below

Where it is clear that the bin has 3 objects.

Once successful pass of the test image to the predictor, I received the below response:

```
labeled_predictions = dict(zip(object_count_range, response[-1]))
print("Labeled predictions: ", labeled_predictions)

Labeled predictions:  {1: 0.019855573773384094, 2: 0.34309279918670654, 3: 0.7109078168869019, 4: 0.6135687828063965, 5: -0.08671106398105621}

print("Most likely answer: {} with probability {}".format(max(labeled_predictions, key=labeled_pred

Most likely answer: 3 with probability 0.7109078168869019

print("Predictor Answer:", max(labeled_predictions, key=labeled_predictions.get))

Predictor Answer: 3
```

Where the model has successfully predicted the right count.

V. Conclusion

In conclusion, implementing the hyperparameter tuning step was important and performed properly where the selected hyperparameter values are acceptable for further development even though the pre-trained ResNet50 model suits my data perfectly but the final accuracy of the trained model is about 0.25, which is considered relatively low however the fact that only a small quantity of the Amazon Dataset with the limited the training process finished in about 20 mins with achieved accuracy is acceptable.

Even though the model is not ready for real time application but it can act as a verification tool for other complex networks.

Improvement

With the current aspect of implementation and designed, I would improve:

- ✓ the training dataset on a bigger part of the dataset should be performed.
- ✓ Comparison with the additional pretrained models and compare performance.
- ✓ As an example, consider ways your implementation can be made more general, and what would need to be modified.
- ✓ The use of multi-instance training would help in decreasing the training time or the capacity of handling the training data size.