Inventory Monitoring at Distribution Centers

AWS Machine Learning Capstone Report

By

Urmil Tamboli

Problem Scoping

Domain Background

AI and Machine learning has become an integrated part of our society and has been used in things never been imagined before with the advances in technology. The scope is AI and ML is vast and it has been a catalyst in solving many problems and savings billions for the industries. In the past decade, there has been tremendous increase in online shopping, and it has impacted many industries like supply chain logistics, warehouse etc. For all the industries to work in harmony and have a smooth operation, it is important to adopt new technologies to keep with the growing demand along with maintaining the efficiency and throughput. There have been many advances in the warehouses due to advances in robotics (Automated Guided/Retrieval System). Modern supply chain is slowing implementing AI&ML in their day to day to improve their operation and reduce errors. For example, Amazon solely handles the supply chain of its products, from warehouse to delivery which makes them a perfect example on why it is important for them to use AI and solve problem and redundancies throughout the process[1,2].

Problem Statement

Amazon currently have a manual process of loading items into the bins depending on the order in the warehouse. It has deployed robots within few warehouses which remove the question of human error but still most of the warehouses are operated by humans. Investing in robots is a huge front investment and it might have a few years to be implemented throughout all the warehouses. When the work is done by human, there comes multiple challenges like placing the right item, right color, right quantity etc. Also, humans get fatigue after long working hours and tends to make more mistakes. We will discuss a few ways to solve this problem – implementing automated warehouse system with robots which could solve it but as discussed above but it might take a long time to be implemented. Second solution would be to find a way to implement AI/ML in such a manner that the human employee is able to track different metrics on a screen which can help in reducing errors.

Solution Statement

In order to solve the above problem, we will go with the second solution. We will be using CNN model to process the Amazon Bin Image Set to classify them. PyTorch Deep Learning framework will be used and different services of AWS like Sagemaker, S3, Endpoints etc. will be used to help us achieve the required solution. This solution will help classify the data. We will be using ResNet pre-trained model and tune hyperparameter as per our requirement to achieve the best possible model to give us the count of items in the bin.

Project Setup and Installation

AWS Sagemaker, S3, Jupyter notebook, Python PyTorch and other libraries were used for this project. Detailed description can be found within the Jupyter notebook. Within SageMaker Studio, we used ml.t3medium instance as it low cost and the training is done on more powerful instance like ml.g4dn.xlarge. Jupyter notebook was used for below cases:

- Download data and upload it to S3 bucket
- Data exploration on the data
- Defining hyperparameters, rules, training containers, debugger, and profiler
- Training the job and hyperparameter tuning and running on multiple instances
- Display profiler report and debugger output on best hyperparameter
- Deploy the endpoints and make predictions.

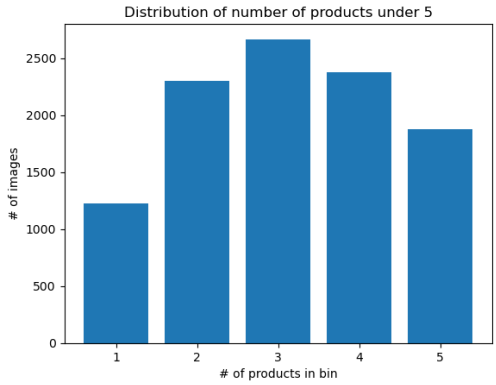
Different scripts were also used along with Jupyter Notebook to complete the above process.

- hpo.py was used to train the model with default parameters and again training it on hyperparameters.
- train.py was used to train the model on the best hyperparameters and then used to obtain the profiler report and debugging output and used to run multi-instance training.
- inference.py was used to create the endpoint to get our predictions.

Data Acquisition and Exploration

Datasets and Inputs

For this project, we will be using the Amazon Bin Image Dataset[3]. This dataset includes over 500,000 bin JPEG images and corresponding JSON metadata files describing items in bins in Amazon Fulfillment Centers. The dataset is available on Amazon S3 and can be imported from there. In this project we will focusing on bin sizes which can contain up to 5 pieces as including all the data would consume heavy resources and as per the histogram most of the images have pieces less than 10 so 5 would be a ideal choice for this exercise. It contains ~10,000 images which would not consume a lot of resources. Below is the histogram and we can see up to 5 pieces, 3 pieces bins have the highest quantity. The data was randomly split into training 60%, validation 20% and testing 20% in the training script.



Data Preprocessing

Following preprocessing was on the data for our requirement:

- Load the image from the input data
- Resize the image to a fixed size
- Convert the image to a PyTorch tensor
- Normalize the pixel values of the image
- Pass the normalized image through the machine learning model for prediction

```
test_transform = transforms.Compose([
transforms.Resize((224, 224)),
transforms.ToTensor(),
transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

logger.info("Transforming input")
normalized=test_transform(input_object)
batchified = normalized.unsqueeze(0)
input1 = batchified.to(device)
```

Modelling

Pipeline

For this project, we have used Resnet50 pretrained model as it has pretty good accuracy and speed compared to other models and is smaller in size. The following steps were done during the modelling:

- Data was first trained on default hyperparameters, and its performance was measured.
- Hyperparameter tuning was done to the above model, and it was trained again with the best hyperparameters.
- Endpoints were created and prediction were made by querying the images.
- Performance was evaluated compared to benchmark model.
- Multi-instance training was performed.

Evaluation Metrics

In this project, we will be using accuracy and RMSE to evaluate the model and compare it with the benchmark model

```
accuracy = correct predictions / total predictions

RMSE = sqrt(sum((predicted_count - true_count)^2) / n)

      n = Total number of observations
```

Cross-Entropy Loss

$$H(p,q) = -\sum p(x) * log(q(x))$$

Where p(x) is the true probability distribution and q(x) is the predicted probability distribution

Loss

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

where n is the number of samples in the test dataset, yi is the true value of the ith sample, and yi^ is the predicted value of the ith sample.

Model Training Metrics

For the model trained with default hyperparameters of learning rate 0.01 and Batch size 64 we get the below results:

```
Testing Loss: 96.47792582078414
Testing Accuracy: 18
RMSE: 1.3228756555322954
```

Training the model with different ranges of hyperparameters as below:

```
"lr": ContinuousParameter(0.001, 0.1),
"batch_size": CategoricalParameter([32, 64, 128, 256, 512]),
"epochs": IntegerParameter(10,40)
```

The best hyperparameter selected were as below:

```
{'batch_size': '32', 'lr': '0.025074731220441404', 'epochs': '30'}
```

We get the below results:

```
Testing Loss: 50.02496290206909
Testing Accuracy: 8.07575798034668
RMSE: 1.3228756555322954
```

Model Benchmarking

For this project we have used made by author can be found here[4]. The performance of the benchmark is as below:

```
Testing Accuracy: 55.67
RMSE: 0.866
```

We can see our model doesn't perform better than the benchmark model and further improvements are needed.

Model Deployment and Predictions

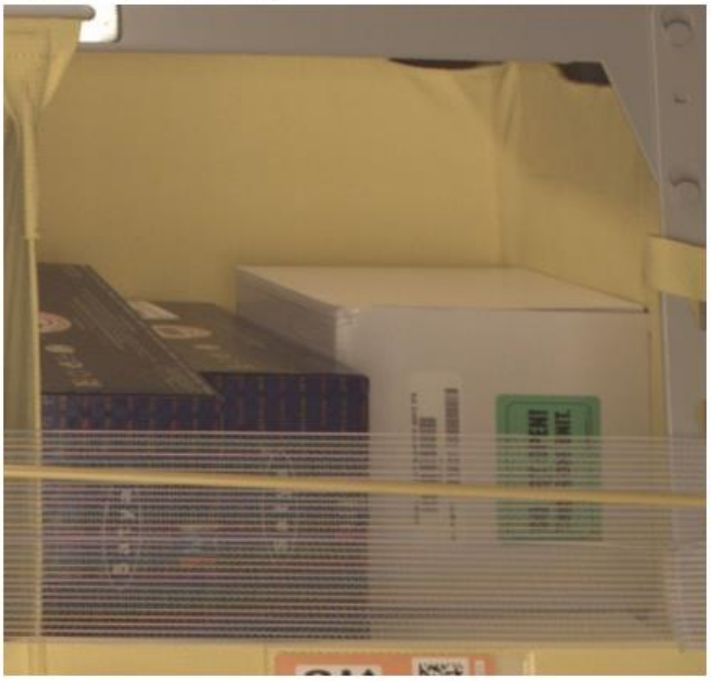
Once we deploy the model, we test to predict it using one of the images from the set named 767. We can see it predicts to have 3 pieces in the bin. It is indexed which is the reason it is showing array([2]) which falls on the third as marked below.

```
In [131]:
import requests

request_dict={ "url": "https://aft-vbi-pds.s3.amazonaws.com/bin-images/767.jpg" }

img_bytes = requests.get(request_dict['url']).content
type(img_bytes)
```

```
Out[131]: bytes
```

```
In [132]:
from PIL import Image
import io
Image.open(io.BytesIO(img_bytes))
```

Out[132]:



```
In [133]:
response=predictor.predict(img_bytes, initial_args={"ContentType": "image/jpeg"})
```

```
In [134]:
response
```

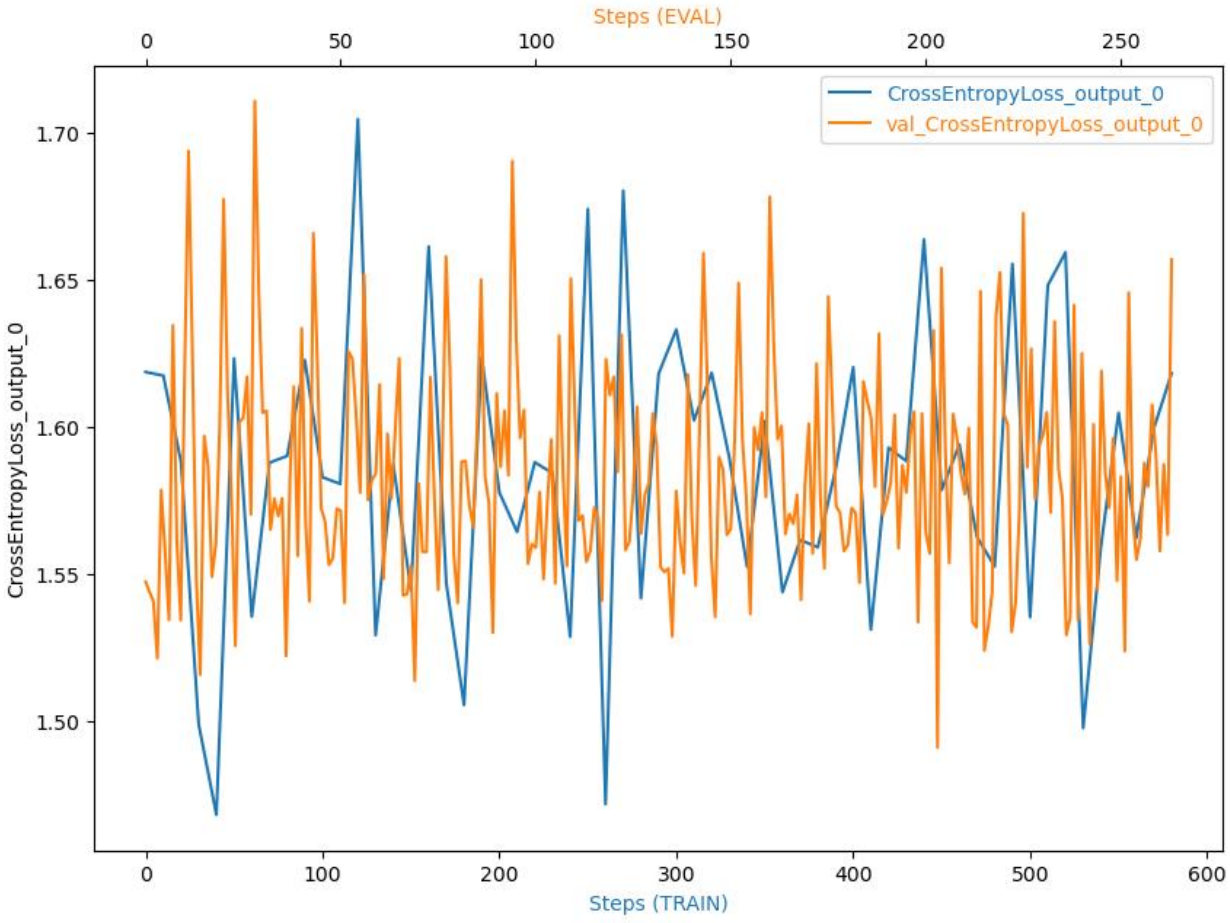
```
Out[134]: [[-0.3417279124259949,
  0.22165267169475555,
  0.32626551389694214,
  0.14200721681118011,
  -0.04346131160855293]]
```

```
In [135]:
import numpy as np
np.argmax(response, 1)
```

```
Out[135]: array([2])
```

Conclusion

The cross-entropy loss graph can be seen below. It is challenging to understand the model learning because of various spikes at many intervals caused by small dataset and outliers. This can be improved by increasing the dataset size and cleaning the dataset to remove the outlier to help us achieve a better result.



References

1. Howe, P. (2023, October 27). *Wakefern achieves inventory management with computer vision and ai*. RFID JOURNAL. https://www.rfidjournal.com/wakefern-achieves-inventory-management-with-computer-vision-and-ai
2. *How to use Computer Vision for inventory monitoring in Supply Chain*. EPAM. (2023, February 28). https://www.epam.com/insights/blogs/how-to-use-computer-vision-for-inventory-monitoring-in-supply-chain
3. *Amazon bin image dataset*. Amazon Bin Image Dataset - Registry of Open Data on AWS. (n.d.). https://registry.opendata.aws/amazon-bin-imagery/
4. Silverbottlep. (n.d.). *Silverbottlep/abid_challenge: Amazon bin image dataset challenge*. GitHub. https://github.com/silverbottlep/abid_challenge