

AWS Machine Learning Engineer Nanodegree

Capstone Project

Inventory Monitoring at Distribution Centers

Naresh K.
23 January 2022

I. Definition

Project Overview

In the recent times, Artificial Intelligence and Machine Learning technologies have been adopted by different verticals. One of the reason that they have become popular now a days is that we now have increased computing power to train large machine learning models and more data to train our models to get an increase in accuracy.

Modern supply chain management can greatly benefit by adopting these technologies. Machine learning in supply chain management can especially impact inventory levels, quality, supply and demand, production planning and transport management. This is important for supply chain management moving forward, particularly when applied to warehouse management.

A smart warehouse combines various interconnected technologies to form an ecosystem whereby an entire business operation, from supply to delivery, is governed by Artificial Intelligence. Goods are received at the warehouse, identified and sorted, processed, packaged, and pulled for shipment, all automatically and with minimal margin for error.

Incase of amazon, the distribution centers like Amazon Fulfillment Centers play an important role in delivering millions of products all over the world. These centers often use robots to move objects as a part of their operations. These products are randomly placed in bins, which are carried by robots. Occasionally, items are misplaced while being handled, resulting in a mismatch between the recorded bin inventory and its actual content.

This problem of mismatched inventory bins can be detected by adopting the Machine learning techniques, which will definitely help further in improving the productivity and efficiency of the distribution centre.

Problem Statement

Amazon uses a random storage scheme where items are placed into accessible bins with available space, so the contents of each bin are random, rather than organized by specific product types. Thus, each bin image may show only one type of product or a diverse range of products. Occasionally, items are misplaced while being handled, so the contents of some bin images may not match the recorded inventory of that bin.

The proposed solution is to train and build a Deep Learning model using a pretrained model that can classify the image based on the number of objects in the bin. The input to the model is an image of a bin with the products in it and the output is list of predicted scores for each category type in this case the number of objects in the bin. Individual instances need to be counted separately, which means if there are two same objects in the bin, count them as two.

Metrics

For this image classification based on the object count, the model predictions can be evaluated by a standrad metrics, accuracy(precision). 1 is indicator function, and p and g is prediction and ground truth respectively.

$$\text{Accuracy: } \frac{1}{N} \sum_{i=1}^N 1[p_i == g_i]$$

II. Analysis

Data Exploration

For this project a subset of Amazon Bin Image Dataset^[1] will be used.

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.

The contents of each bin are random, rather than organized by specific product types. Thus, each bin image may show only one type of product or a diverse range of products.

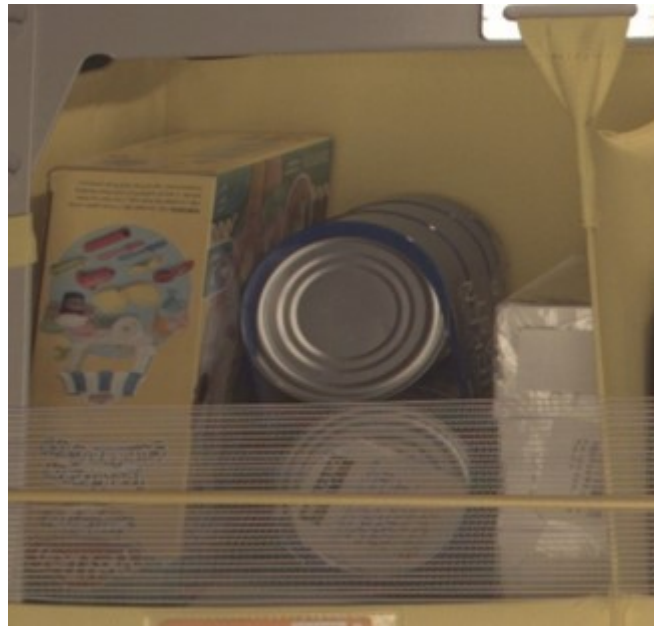
Sample images from the dataset;



A bin contains multiple object categories and various number of instances. The size of bins are various depending on the size of objects in it. The tapes in front of the bins are for preventing the items from falling out of the bins and sometimes it might make the objects unclear. Objects are sometimes heavily occluded by other objects or limited viewpoint of the images.

The corresponding metadata exists for each bin image and it includes the object category identification (Amazon Standard Identification Number, ASIN), quantity, size of objects, weights, and so on.

An example of image(jpg) and metadata(json) pair.



```
{
  "BIN_FCSKU_DATA": {
    "B00CFQWRPS": {
      "asin": "B00CFQWRPS",
      "height": {
        "unit": "IN",
        "value": 2.399999997552
      },
      "length": {
        "unit": "IN",
        "value": 8.199999991636
      },
      "name": "Fleet Saline Enema, 7.8 Ounce (Pack of 3)",
      "normalizedName": "(Pack of 3) Fleet Saline Enema, 7.8 Ounce",
      "quantity": 1,
      "weight": {
        "unit": "pounds",
        "value": 1.8999999999999997
      },
      "width": {
        "unit": "IN",
        "value": 7.199999992656
      }
    },
    "ZZXIOWUSIB": {
      "asin": "B00T0BUKW8",
      "height": {
        "unit": "IN",
        "value": 3.99999999592
      },
      "length": {
        "unit": "IN",
        "value": 7.899999991942001
      },
      "name": "Kirkland Signature Premium Chunk Chicken Breast Packed in Water, 12.5 Ounce, 6 Count",
      "normalizedName": "Kirkland Signature Premium Chunk Chicken Breast Packed in Water, 12.5 Ounce, 6 Count",
      "quantity": 1,
      "weight": {

```

```

        "unit": "pounds",
        "value": 5.7
    },
    "width": {
        "unit": "IN",
        "value": 6.49999999337
    }
},
"ZZXVVS669V": {
    "asin": "B00C3WXJHY",
    "height": {
        "unit": "IN",
        "value": 4.330708657
    },
    "length": {
        "unit": "IN",
        "value": 11.1417322721
    },
    "name": "Play-Doh Sweet Shoppe Ice Cream Sundae Cart Playset",
    "normalizedName": "Play-Doh Sweet Shoppe Ice Cream Sundae Cart
Playset",
    "quantity": 1,
    "weight": {
        "unit": "pounds",
        "value": 1.4109440759087915
    },
    "width": {
        "unit": "IN",
        "value": 9.448818888
    }
},
    "EXPECTED_QUANTITY": 3
}

```

This image contains 3 different object categories. For each category, there is one instance. So, "EXPECTED_QUANTITY" is 3, and for each object category "quantity" field was 1. Unique identifier("asin") is assigned to each object category, here "B00CFQWRPS", "B00T0BUKW8", and "B00C3WXJHY".

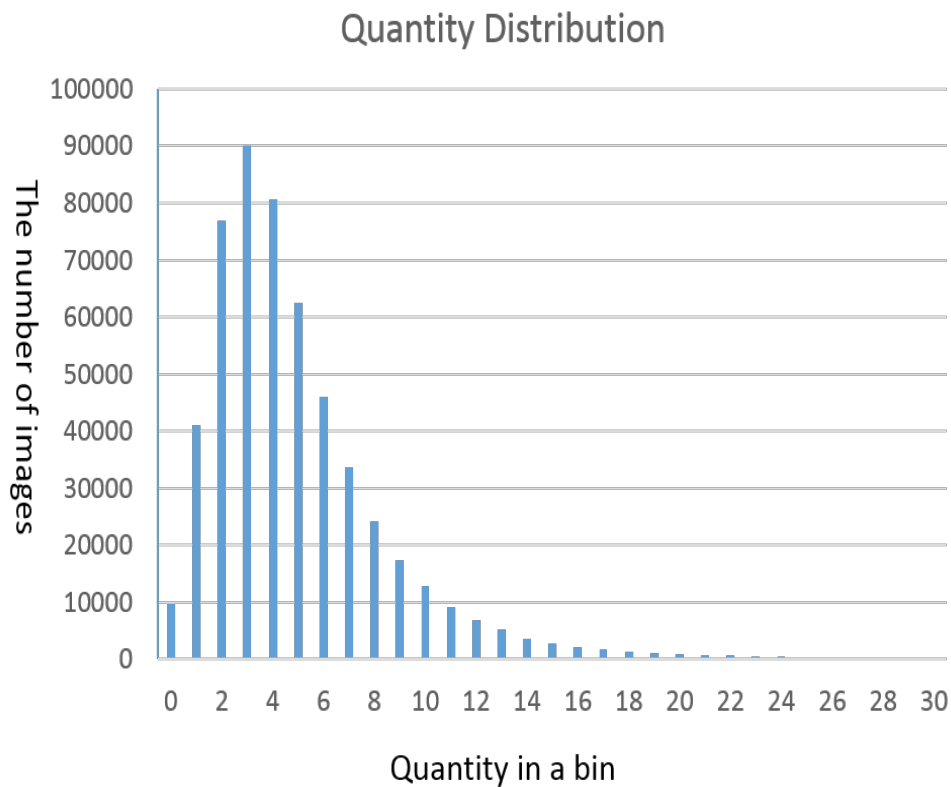
The dataset is available on Amazon S3^[1]. Images are located in the bin-images directory, and metadata for each image is located in the metadata directory.

Exploratory Visualization

The plot below shows the distribution of number of products in a bin. Here we can see 90% of bin images contains less than 10 object instances in a bin and the average quantity in a bin is 5.1^[3].

Data Statistics:

Description	Total
The number of images	535,234
Average quantity in a bin	5.1



Algorithms and Techniques

A Convolutional Neural Network is trained for this image classification task using transfer learning. Here a pretrained model called '**resnet50**' has been used to fine tune on the amazon bin image dataset.

To finetune the resnet50 CNN, following steps are performed:

- create or load the pre-trained model
- freeze all the convolutional layers
- add a fully connected layer and train it.

In this training paradigm, the forward pass will run for the whole network and since the convolutional layers are frozen, the backward pass (and update weights) will run for only the fully connected layers.

The following hyperparameters are used during tuning and training;

- training length (number of epochs)
- batch size (number of images to look at once during a single training step)
- learning rate (how fast to learn)

As this is a multi class classification problem **Cross entropy loss** function is used as the cost function and **Adaptive Moment Estimation (Adam)** method is used as an optimizer.

Benchmark

Since this is a classification problem the model output can be evaluated by accuracy.

The benchmark for this project is to be able to get the model accuracy of **55.67%** or more. This value has been taken from the Amazon Bin Image Dataset Challenge^[3].

III. Methodology

Data Preprocessing

Since this is a large dataset, only a subset of the data has been used for this project. The subset of bin images dataset were chosen so that the number of objects in a bin range from '1' to '5' only.

The subset of data created are arranged in subfolders. Each of these subfolders contain images where the number of objects is equal to the name of the folder. For instance, all images in folder 3 has images with 3 objects in them. The number of objects will serve as the categories for classification of bin images.

The list of file names to be downloaded is collected in file_list.json. The metadata(json) filenames are arranged in json object using number of images in bin as the 'key'.

Eg:

```
{
  "1": ["data/metadata/100313.json",
        "data/metadata/09915.json",.....],
  "2": ["data/metadata/102489.json",
        "data/metadata/104982.json",.....],
  .....
  "5": ["data/metadata/03858.json",
        "data/metadata/101598.json",.....]
}
```

The number of images in each class are;

Category (number of objects in a bin)	Total Images
1	1228
2	2299
3	2666
4	2373
5	1875

This dataset is split into training, testing and validation sets in ratio of 8:1:1.

I observed, some of the images are of very poor quality and the objects are not at all visible to human eye. So, some images may need to be deleted to improve the training performance.

Implementation

The AWS SageMaker, S3, Cloudwatch etc. have been used to carry out the implementation of the project. The code is implemented using python and its libraries like pytorch, torchvision etc.

Setup AWS SageMaker Notebook instance

Create and open a Sagemaker instance, I have chosen 'ml.t2.medium' instance type. This instance comes with 2vCPUs and 4GiB memory. This configuration is good enough for running the project jupyter notebook and the price is also one of the least.

Then create or load the project files required to your workspace.

Training Script (*train.py* / *hpo.py* / *inference.py*)

The training script takes the hyperparameters as arguments and reads, loads and preprocesses the train, test and validation data. Then after creating and loading the pretrained 'resnet50' model it trains the model using the given parameters.

hpo.py: used for hyperparameter tuning job.

train.py: used for fine tuning the model with best hyperparameters.

inference.py: used for deploying the trained model.

Submission Script (*sagemaker.ipynb*)

This jupyter notebook has the code cells to interface with the sagemaker and submit all the jobs to it. The following jobs are done using this script;

- Data download and preparation: Download the subset of the Amazon bin image dataset^[1]. The *file_list.json* has the list of files to download. The dataset is split into train, test and validation directories in the ratio of 8:1:1.
- Data Upload to S3: Upload the downloaded data to the required S3 bucket.
- Setup the Hyperparameter search space
- Setup the model debugging and profiling rules
- Create the model estimator
- Create the Hyperparameter tuning job
- Submit the tuning job
- Fetch the best training job and the hyperparameters
- Finetune the model with the best hyperparameters
- Deploy the model
- Query the model with a bin image and get prediction

Refinement

In this project, I created a deep learning work flow using AWS SageMaker, which we can use to train the image classification model on Amazon bin image dataset. With minimal changes in the scripts the training process can be performed iteratively with various parameters and larger datasets. So that, we can get the better performing model.

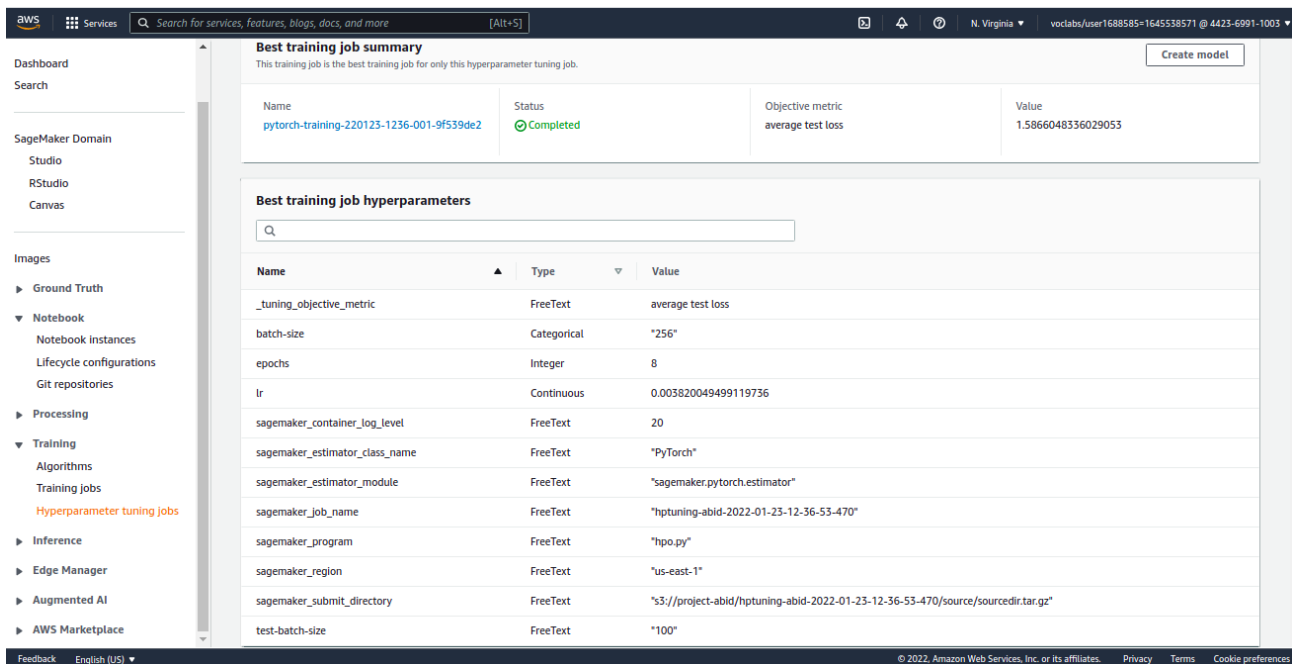
Initially, I used the pretrained model '**resnet50**' to fine tune the model on the data subset using fixed hyperparameters;

The model test set loss and accuracy are;

Test set: Average loss: 1.5935008831864572

Testing Accuracy: 0.22701148688793182

I performed hyperparameter tuning with the subset of bin images dataset. Below is the screen shot of hyperparameter tuning job with the best training job hyperparameters;



The screenshot displays the AWS SageMaker console interface. The top navigation bar includes the AWS logo, a search bar, and user information. The left sidebar shows the SageMaker Domain menu with options like Studio, RStudio, and Canvas. The main content area is titled 'Best training job summary' and includes a 'Create model' button. Below this, a table lists the job details: Name (pytorch-training-220123-1236-001-9f539de2), Status (Completed), Objective metric (average test loss), and Value (1.5866048336029053). The 'Best training job hyperparameters' section features a search bar and a table with columns for Name, Type, and Value. The hyperparameters listed are: _tuning_objective_metric (average test loss), batch-size (256), epochs (8), lr (0.003820049499119736), sagemaker_container_log_level (20), sagemaker_estimator_class_name (PyTorch), sagemaker_estimator_module (sagemaker.pytorch.estimator), sagemaker_job_name (hptuning-abid-2022-01-23-12-36-53-470), sagemaker_program (hpo.py), sagemaker_region (us-east-1), sagemaker_submit_directory (%3:/project-abid/hptuning-abid-2022-01-23-12-36-53-470/source/sourcedir.tar.gz), and test-batch-size (100).

Name	Type	Value
_tuning_objective_metric	FreeText	average test loss
batch-size	Categorical	"256"
epochs	Integer	8
lr	Continuous	0.003820049499119736
sagemaker_container_log_level	FreeText	20
sagemaker_estimator_class_name	FreeText	"PyTorch"
sagemaker_estimator_module	FreeText	"sagemaker.pytorch.estimator"
sagemaker_job_name	FreeText	"hptuning-abid-2022-01-23-12-36-53-470"
sagemaker_program	FreeText	"hpo.py"
sagemaker_region	FreeText	"us-east-1"
sagemaker_submit_directory	FreeText	"%3:/project-abid/hptuning-abid-2022-01-23-12-36-53-470/source/sourcedir.tar.gz"
test-batch-size	FreeText	"100"

The best performing hyperparameters are;

```
{  'batch-size': "256",
  'epochs': '8',
  'lr': '0.003820049499119736',
  'test-batch-size': "100"
}
```

IV. Results

Model Evaluation and Validation

The final model has been deployed with the best performing hyperparameters. I used PyTorchModel for deploying the model.

Once the model is successfully deployed on Endpoint, queried the endpoint using the predict() method.

Screenshot of the endpoint results;

```
In [60]: from PIL import Image
import io
Image.open(io.BytesIO(img_bytes))
```

Out[60]:



```
In [61]: response = predictor.predict(img_bytes, initial_args={"ContentType": "image/jpeg"})
```

```
In [62]: type(response)
```

Out[62]: list

```
In [63]: response
```

```
Out[63]: [[-0.1935795694589615,
            0.01356017217048062,
            0.010463185608386993,
            0.06341414898633957,
            -0.06604141741991043]]
```

```
In [64]: import numpy as np
np.argmax(response, 1)
```

Out[64]: array([3])

The prediction results for this image is correct, but the accuracy percentage is low.

When I queried the endpoint with different images it returned wrong results.

The overall workflow of the model training looks robust but the model needs more training with larger datasets and also need to explore iteratively for better performing CNN models and hyperparameters.

The one important thing to consider is using the larger dataset and cleanup of images with the wrong label and unclear bin images.

Justification

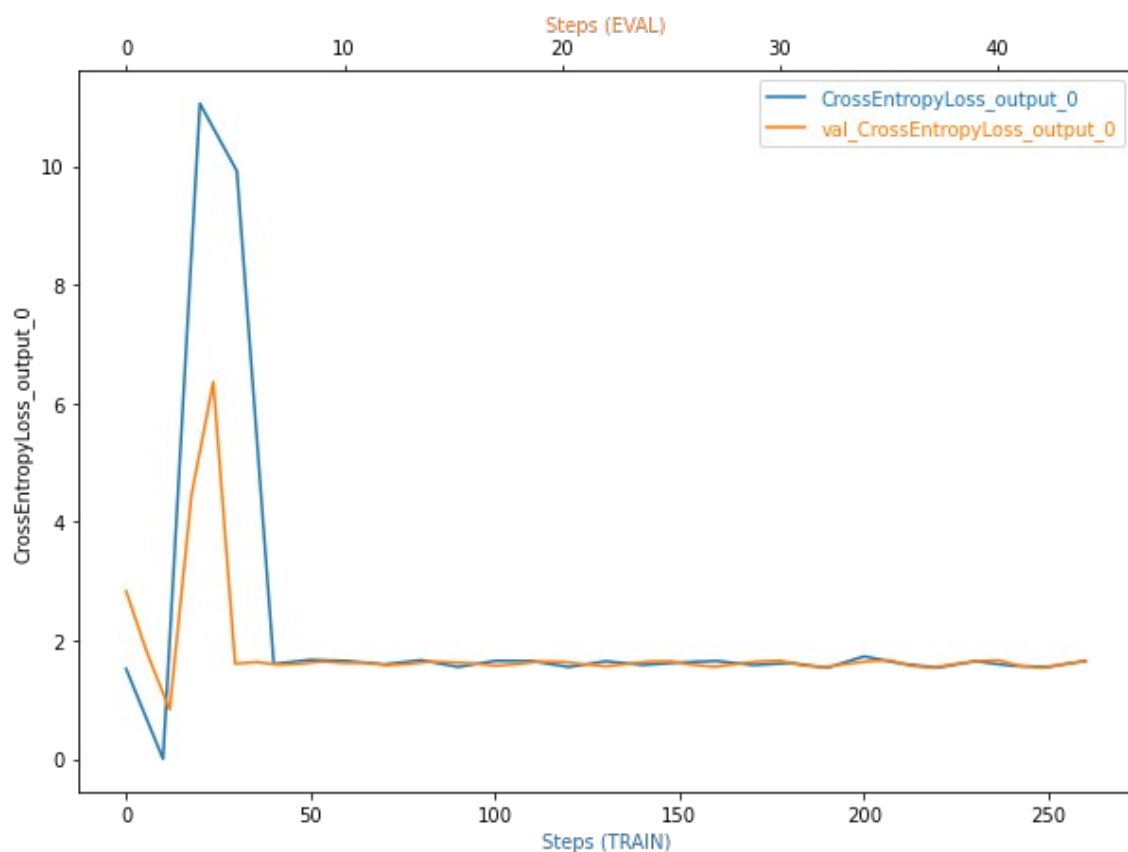
With the small subset of images dataset the model's accuracy reached saturation point and reached the testing accuracy of 0.22701148688793182 and is below the bench mark result mentioned earlier.

Using the AWS ML work flow that's been built more training can be performed to achieve better accuracy.

V. Conclusion

Free-Form Visualization

Below is the plot of CrossEntropyLoss captured against number of epochs during the model training;



The AWS SageMaker provides robust platform for developing machine learning models. There are debugger and profiling libraries or services which can be used in debugging and analysing the model training process.

To build an ML model some of the important things to take care include but not limited to are accuracy and quantity of the dataset, neural network model architecture, different parameters, the loss function, optimizer functions and cost of the resources.

Also, I think things like increasing the training time, using larger datasets and model architectures with more number of layers blindly may not work, one has to find the optimum values or resources when building an ML model.

Resources and references:

1. Amazon Bin Image Dataset : <https://registry.opendata.aws/amazon-bin-imagery/>
2. Documentaions : <https://github.com/awslabs/open-data-docs/tree/main/docs/aft-vbi-pds>
3. Amazon Bin Image Dataset(ABID) Challenge : https://github.com/silverbottlep/abid_challenge
4. Amazon Inventory Reconciliation using AI : <https://github.com/pablo-tech/Image-Inventory-Reconciliation-with-SVM-and-CNN>
5. Project Starter files : <https://github.com/udacity/nd009t-capstone-starter>
6. <https://towardsdatascience.com/>
7. <https://supplychaingamechanger.com/>

Thank You