Capstone Project

Machine Learning Engineer Nanodegree

Yalu Su

Jan 28th,2023

Inventory Monitoring at Distribution Centres

1. Definition

1.1 Project Overview

Traditional supply chain models focus on storing large quantities of product across different locations, which is heavily relied on manual processes. Modern supply chain models focus on having the right quantity of product , available at right time at the right place.[1]

In distribution centres, robots are often used to move objects as part of their operations. Bins size and the number of objects in each bin differs, thus delivery of correct consignments can be challenging. To complete the task efficiently, machine leaning could be utilized to predict the number of items in each bin.

The pipeline of the project is designed as the following steps:

1. Download the data and upload to S3 bucket

2. Complete multi-instance Hyperparameter tuning

3. Model profiling and debugging

4. Model Performance

6. Endpoint deployment and Lambda querying

1.2 Problem Statement

Manual processes in traditional supply chain models may lead to low efficiency and waste of resources, such as employ time, machine cost, human error etc.

Leveraging computer vision and machine learning techniques will enable an automation of inventory monitoring systems, which is able to minimize these problems and eliminate human error. This will in turn assist in saving costs, time and increase efficiency in inventory monitoring.

1.3 Metrics

Accuracy is a standard metric for evaluating classification problems.

Accuracy is defined in Figure 1, where *i* is an indicator function, *p* is the prediction, *g* is the ground truth and *N* is the number of samples in the dataset [2].

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

Figure 1: Accuracy equation

2. Analysis

2.1 Data Exploration

The publicly available Amazon Bin Image Dataset will be used in this project. This dataset contains over 500,000 images, where each image contains one or more objects. In addition, the dataset contains JSM metadata from bins of pod in an operating Amazon Fulfilment Centre, such as number of objects, it's dimension and the type of object. An example image is shown in Figure 2. [3]

Figure 2: Example image

A subset of the data will be used in the project to control budget and save computation time. Training data was download and arranged to it 5 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 `1` has images with 1 object in them. The data subset contains 10441 bin images.

2.2 Exploratory Visualization

The distribution of the number of products per bin in the data subset is shown in Figure 3. There are 5 classes, which ranges from 1 to 5. The most and least common number of products per bin is 3 and 1.

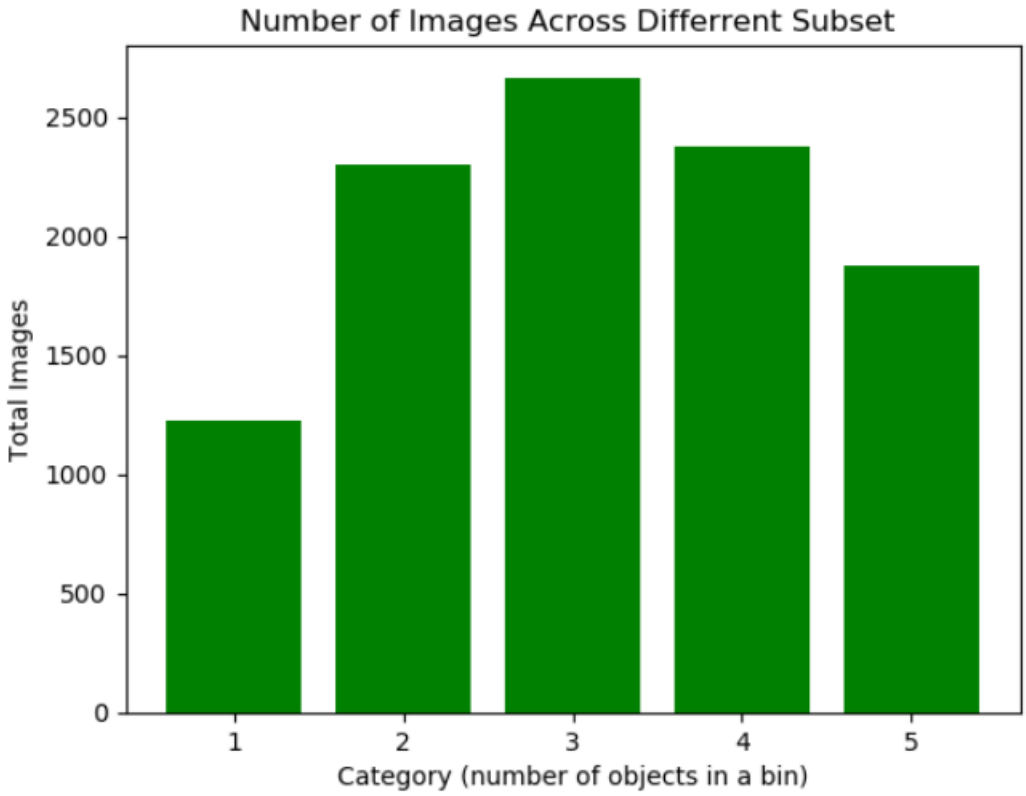

Figure 3: Distribution of Subsets

The exact number of images in each subset is shown in Table 1.

Table 1: Number of Images in Each Subset

Subset	**Total Images**
1	1228
2	2299
3	2666
4	2373
5	1875

2.3 Algorithms and Techniques

Restnet50 has been used to fine tune the amazon image bin dataset. It was selected as it gives top accuracy with less computing time and data size comparing with other pre-trained models, as shown in Figure 4.

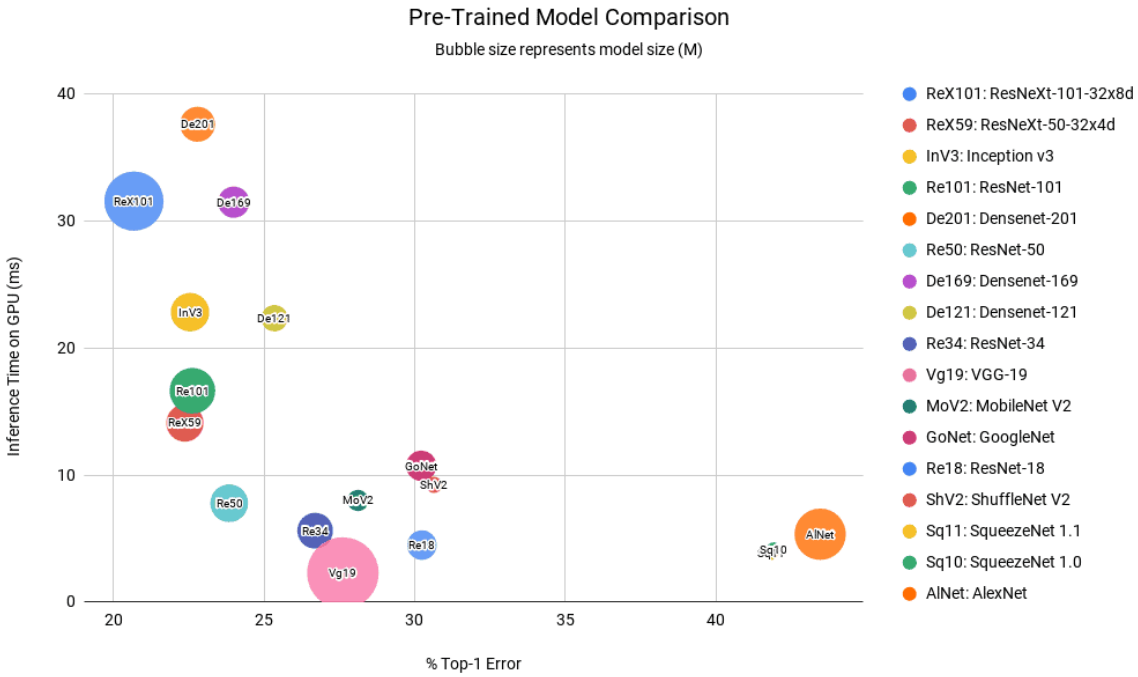

Figure 4: Pre-trained Model Comparison[4]

Cross Entropy Loss is widely used as cost function for classification problems. Image predictions and true labels are the input.[5]

Adaptive Moment Estimation (Adam) is a widely used optimization algorithm, which can update network weights iterative based in training data.[6]

Transfer Learning fine tunes a pre-trained model on new dataset. The following process was performed:

1. Load pre-trained model
2. Frozen convolutional layers
3. Add fully connected layers
4. Trained model

Hyperparameter Tunning was applied to find best hyperparameters in each given range. Hyperparameters selected are:

1. Learning rate (speed of learning)
2. Batch size (number of images to look at during a single training step)

Multi-instance Training was performed to train model in at least 2 compute instances at the same time, for better device utilization and quicker training.

2.4 Benchmark

The benchmark is referred to the Abid Challenge, which the author obtained 55.67% accuracy.[2]

3. Methodology

3.1 Data Pre-processing

The dataset is split to training, testing and validation sets to [5221, 2611, 2609], which is approximately ratio of 2:1:1.

Images were resized to 224×224 before being passed the model.

3.2 Implementation

3.2.1 Download the data and upload to S3 bucket

The training data was uploaded to S3 so SageMaker can use them for training, as shown in Figure 5.

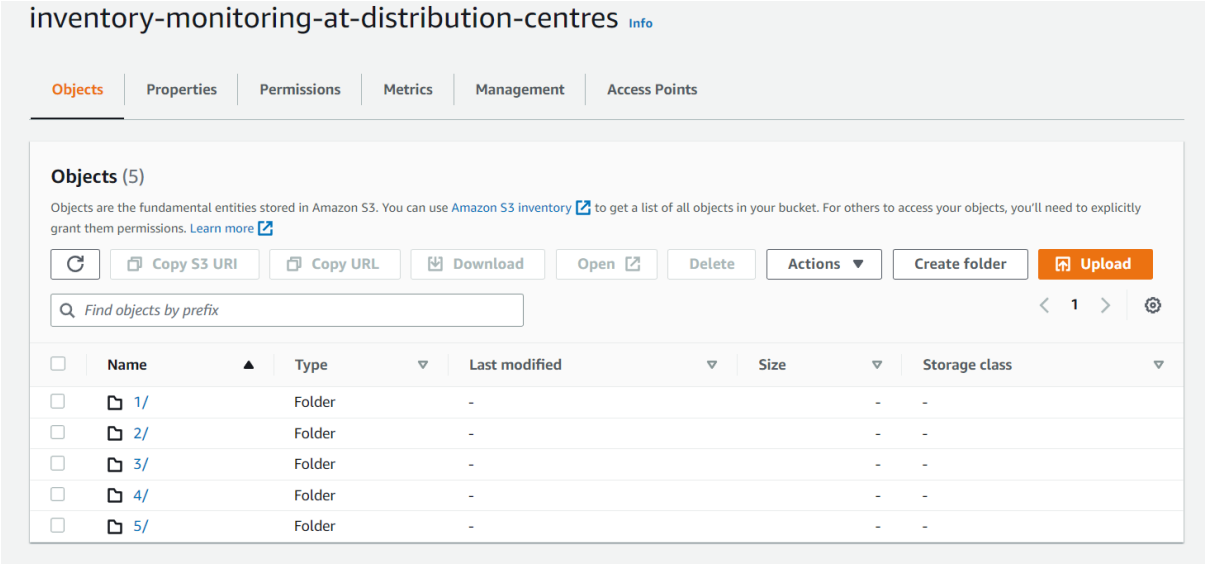

Figure 5: Data uploaded to S3

3.2.2. Complete multi-instance hyperparameter tuning

Hyperparameters and range selected:

1. learning_rate: (0.001, 0.1)
2. batch_size: [32, 64, 128, 256, 512]

Best hyperparameters found to be:

1. learning_rate: 0.0027
2. batch_size: 128

Completed training job and best hyperparameters are shown in Figure 6.

Hyperparameters

Key	Value
_tuning_objective_metric	Test Loss
batch_size	"128"
learning_rate	0.0027047446326561825
sagemaker_container_log_level	20
sagemaker_estimator_class_name	"PyTorch"
sagemaker_estimator_module	"sagemaker.pytorch.estimator"
sagemaker_job_name	"pytorch_hpo-2023-01-25-18-16-21-219"
sagemaker_program	"train.py"
sagemaker_region	"eu-west-2"
sagemaker_submit_directory	"s3://sagemaker-eu-west-2-237497095970/pytorch_hpo-2023-01-25-18-16-21-219/source/sourcedir.tar.gz"

Figure 6: Best hyperparameters

3.2.3 Model profiling and debugging

Another model was trained with best hyperparameters found previously, with the following rules:

1. Vanishing gradient
2. Overfit
3. Overtraining
4. Profile Report

3.2.4 Model performance

The plot of cross entropy loss captured against the number of epochs during model training is shown in Figure 7.

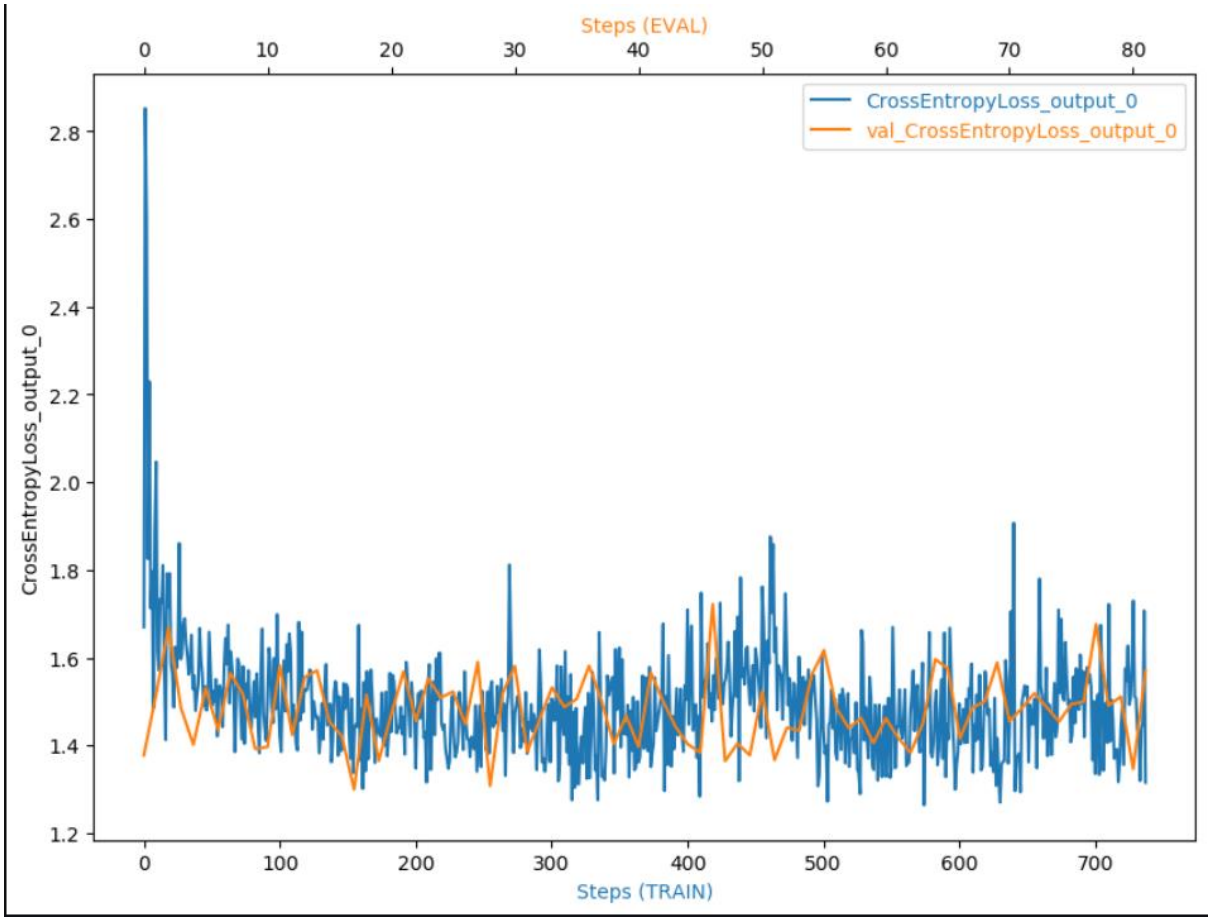

Figure 7: Performance Output

Both lines are stabilizing, which indicates the model is learning through the steps.

3.2.5 Endpoint deployment and Lambda querying

Final model was deployed to endpoint as shown in Figure 8.

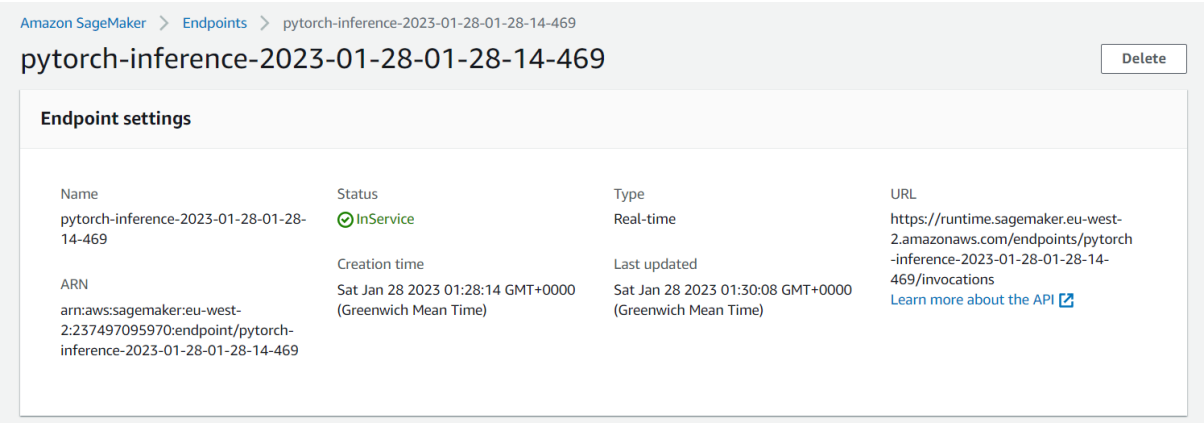

Figure 8: Endpoint Deployment

Lambda function was also created, deployed, and tested for sample images. An example of successfully tested lambda function is shown in Figure 9.

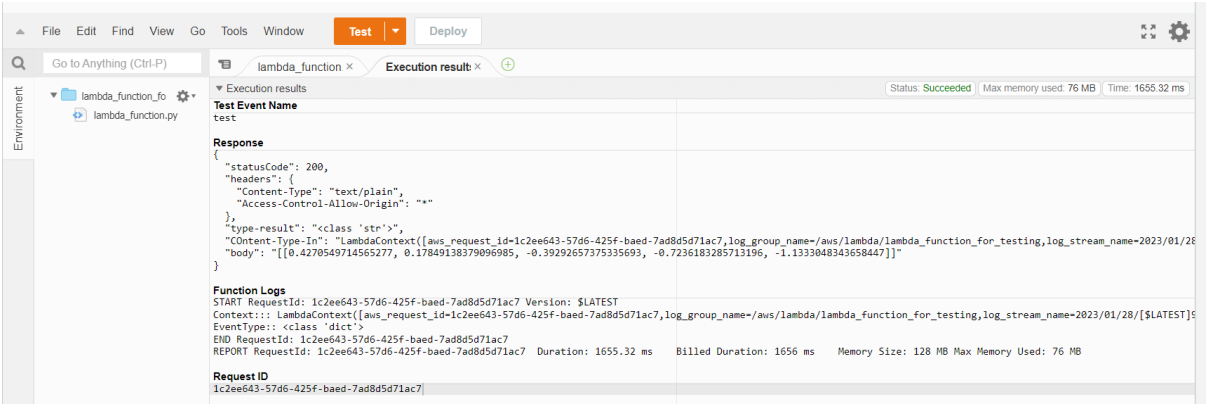

Figure 9: Lambda function tested

3.3 Refinement

3.3.1 Hyperparameter tuning (initial model)

Table 2: Hyperparameter spaces

Learning rate	(0.001, 0.1)
Batch size	[32, 64, 128, 256, 512]

Table 3: Best hyperparameters

Learning rate	0.0027047446326561825
Batch size	128

3.3.2 SageMaker debugger and profiling (final model)

Table 4: Final model performance

Loss	47
Accuracy	9

4. Results

4.1 Model Evaluation and Validation

The final model was deployed to an endpoint and a lambda function was created to test samples. There are 3 images tested in total, with 3 separate responses:

1. The first image tested is shown in Figure 10

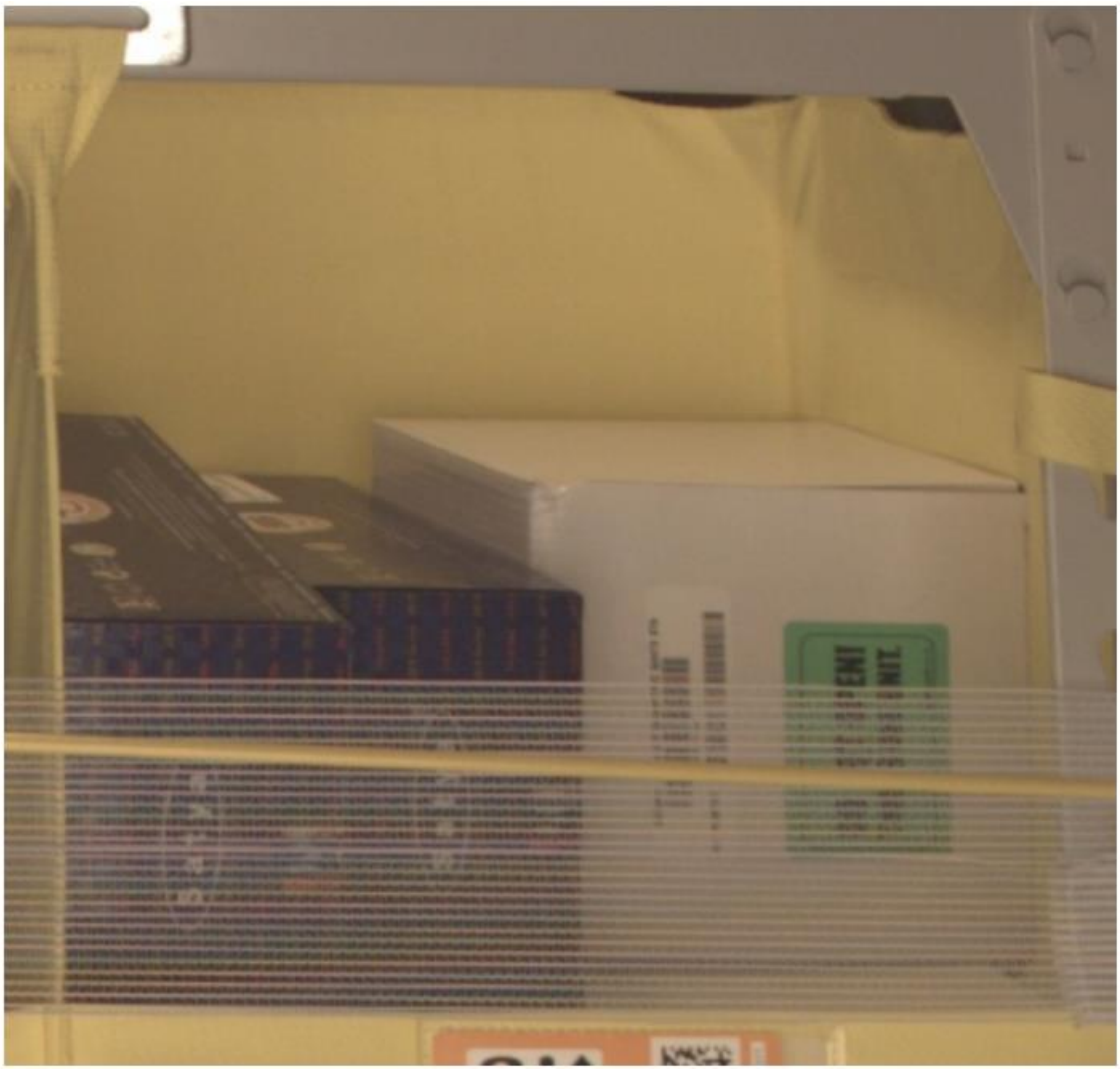

Figure 10: Example image 1

The result for the first image is shown below:

```
[-2.0685417652130127,
 -0.05794139206409454,
 0.17876869440078735,
```

```
 0.25809329748153687,
 -0.023833414539694786]
```

2. The first image tested is shown in Figure 12

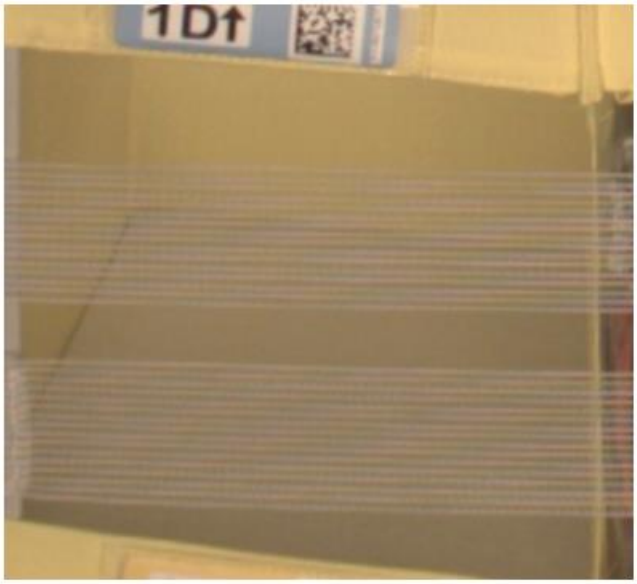

Figure 12: Example image 2

The result for the second image is shown below:

```
[0.4270549714565277,
 0.17849138379096985,
 -0.39292657375335693,
 -0.7236183285713196,
 -1.1333048343658447]
```

3. The third image tested is shown in Figure 13

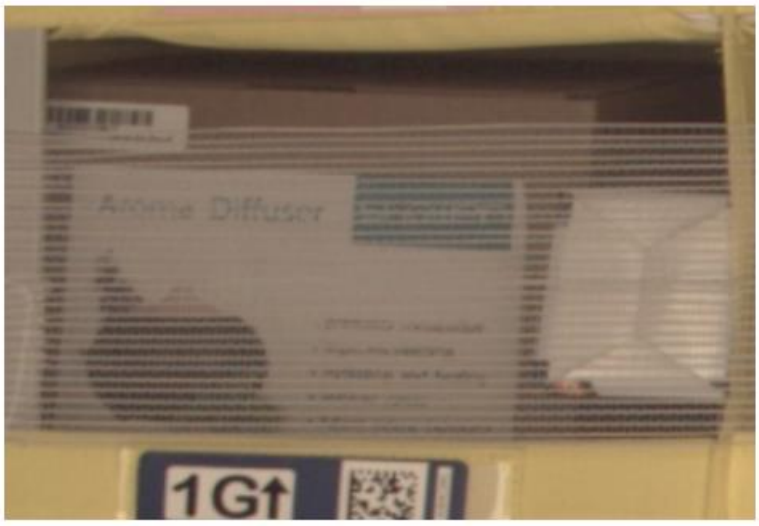

Figure 13: Example image 3

The result for the third image is shown below:

```
[-1.894127607345581,
 -0.12563461065292358,
 0.22617845237255096,
 0.3510580062866211,
 0.28727301955223083]
```

The model was not able to correctly predict these sample images. An interesting patten found here is that the prediction is always 1 less than true labels.

4.2 Justification

The performance for the benchmark and final model is compared in Table 5.

Table 5: Performance comparison

	Benchmark	Final model
Accuracy	55.67	9

Accuracy for final model has not reached the benchmark. Future improvement would be required to improve accuracy to generalize the solution and solve the problem adequately.

5. Further Improvement

1. There are images with low quality, such as blurry images and the label does not match the actual number of objects in a bin, such as Figure 12, where the bin is empty, but the true label is 1. Labelling issues could be solved through AWS SageMaker Ground Truth with high cost.
2. Increasing the training dataset might improve model performance
3. Include more hyperparameters in tuning process, such as number of epochs.
4. Improve model architectures with more layers and try different cost functions.

References

[1] Supply Chain 101: Warehouse vs. Distribution Center - Smart Warehousing

[2] silverbottlep/abid_challenge: Amazon Bin Image Dataset Challenge (github.com)

[3] https://registry.opendata.aws/amazon-bin-imagery

[4] Image Classification using Pre-trained Models in PyTorch | LearnOpenCV

[5] A Gentle Introduction to Cross-Entropy for Machine Learning - MachineLearningMastery.com

[6] Gentle Introduction to the Adam Optimization Algorithm for Deep Learning - MachineLearningMastery.com