AWS Machine Learning Engineer Nanodegree Program

Inventory Monitoring at Distribution Centers (Project Report)

Name: Luiz Otávio Matias da Luz

1 - Definition

1.1 Project Overview

Inventory monitoring is crucial and important in distribution centers. The way you manage and monitor inventory in a distribution center influences how much you spend on logistics costs and how efficient you can get. [1] This process makes sure that each bin carries the right number of objects.

Tracking things manually was the norm before the industrial age. The earliest form of inventory management dates back over 50,000 years in which people used “tally sticks” to count. [2]

It is now possible to improve customer satisfaction, prevent dead stock, monitor and optimize stock levels in inventory management by implementing machine learning.

This project aims to implement a machine learning solution to monitoring inventory at distribution centers.

1.2 Problem Statement

There are a lot of labor requirements associated with manual inventory management, as well as the possibility of errors. So, there is a need for effective inventory management.

Proper warehouse inventory management can save you money, including labor, storage, and fulfillment costs. This can be done by optimizing storage, investing in warehouse automation, organizing inventory to speed up the picking and packing process, and implementing technology to automate tasks that reduce human error. [3]

Machine learning is one technology that can automate those processes. Computer vision can be used to automate inventory monitoring using convolution neural networks to count the number of objects in each bin and ensure that delivery consignments contain the correct number of items.

1.3 Metrics

Since this is an image classification task in machine learning, accuracy is used to evaluate the model performance.

2 - Analysis

2.1 Data Exploration

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. [4]

Since this is a large dataset, a sample of the dataset is used. The dataset is available on Amazon S3.

There will be subfolders for the data subset created. The number of objects in each of these subfolders equals the name of the folder. In folder 1, for example, there is exactly one object in every image. Afterwards, training, testing, and validation datasets will be developed.

Example of image:

Example metadata (json):

```
{
 "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
 }
 },
 "ZZXI0WUSIB": {
 "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
}

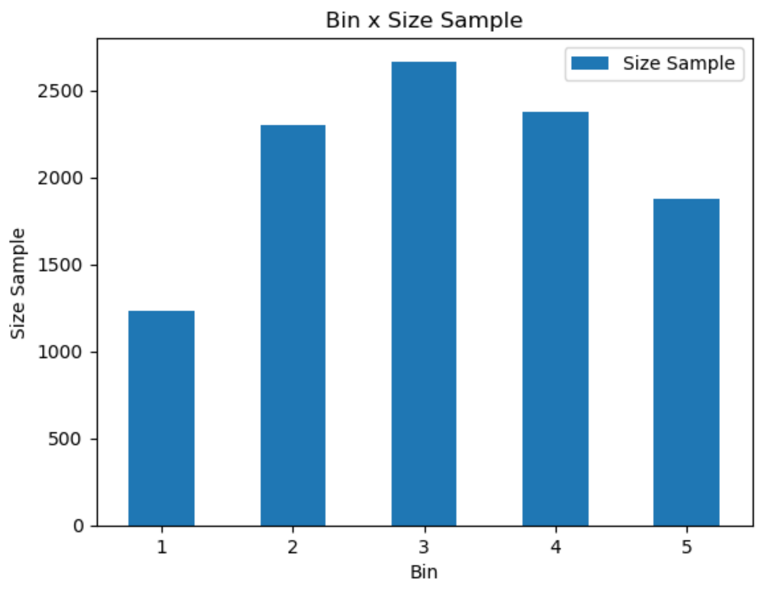
2.1 Exploratory Visualization

Since the dataset is large, a sample of the dataset was used.

The table below shows the bins distribution for the sample taken. So, it can be verified that the data was a little unbalanced since bin 1 had 1229 samples, while bin 3 had 2666.

Bin	Size Sample
1	1229
2	2299
3	2666
4	2373
5	1875

The figure below shows the distribution in a graphic way, so it is possible to see the differences more clearly. It can be seen that bin 1 was significantly smaller than the other bins.



2.2 Algorithms and Techniques

To solve this image classification problem, computer vision techniques will be used, in this case a pre-trained (Resnet 50) deep learning model. As input, we provide an image of a bin containing products and as output, we give predicted scores for each category type, for instance, number of objects. All data will be stored at S3.

The algorithm will be built using Convolution Neural Network (CNN) and the framework Pytorch. To improve the performance of the model, SageMaker's Hyperparameter Tuning will be used. After that, SageMaker endpoint will be created to deploy the model.

The SageMaker Studio will be used as our environment. To train the model, the instance ml.g4dn.xlarge was used. The endpoint used the ml.m5.large instance.

The Hyperparameter tuning was used in the batch size and the learning rate. As this is a multi-class classification problem, the cross-entropy loss function was used and the Adam for optimizer.

2.3 Benchmark

The dataset has two benchmarks that can be found at the Registry of Open Data on AWS. The publication Amazon Bin Image Dataset Challenge by silverbottlep achieved an accuracy of 55.67%. Unfortunately, we were unable to reach the benchmark. We could have tried another neural network architecture and tuned other hyperparameters.

3 - Methodology

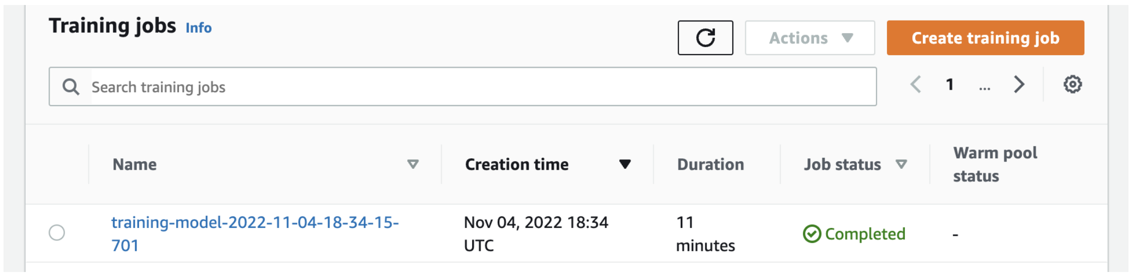
3.1 Data Preprocessing

The dataset was downloaded from a URL. Every image in the sample dataset was converted into image bytes and stored in S3. All images were resized using PyTorch transformers to 224x224 pixels. (transforms.Resize((224, 224))). The training dataset contains 60% of the data, the validation and test dataset contains 20%. The data loaders are created at a batch size of 64.

3.2 Implementation

All implementation was done in SageMaker. First the data was downloaded and stored on the S3. An instance of SageMaker (ml.t3.medium) was used to develop the algorithm.

A pre-trained model (ResNet50) with the PyTorch framework was used in the training and a notebook instance with GPU (ml.g4dn.xlarge) was used in the training job.



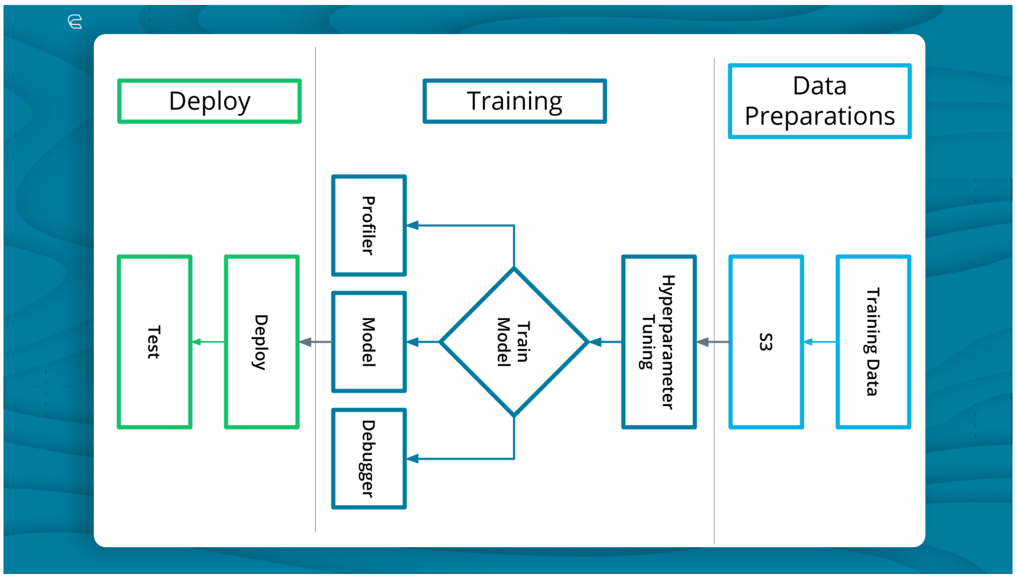
	Name	Creation time	Duration	Job status	Warm pool status
○	training-model-2022-11-04-18-34-15-701	Nov 04, 2022 18:34 UTC	11 minutes	Completed	-

During training, we used a batch size of 64 and a learning rate of 0.00075, which were found after tuning the model.

Model debugging and profiling was also used to better monitor and debug the model training job. The model was deployed to an endpoint (ml.m5.large instance) and tested using an image.

As this is a multi-class classification problem, the cross-entropy loss function was used and the Adam for optimizer.

The project pipeline can be seen in the image below:



3.3 Refinement

Hyperparameters tuning was performed with SageMaker to find the best hyperparameters and refine the model.

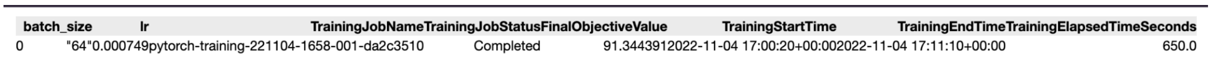
Parameters range used for the hyperparameters search:

- learning rate - 0.0001 to 0.1
- batch size - 64, 128 and 256

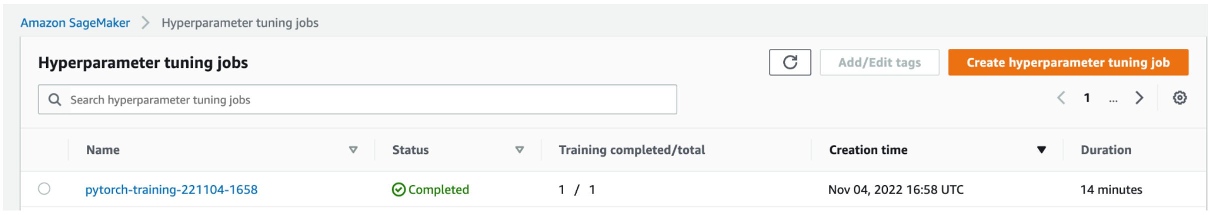
We can set this value behind the algorithm below:

```
hyperparameter_ranges = {
  "lr": ContinuousParameter(0.0001, 0.1),
  "batch_size": CategoricalParameter([64, 128, 256]),
}
```

After running the hyperparameters job, best hyperparameters found can be seen below:



	batch_size	lr	TrainingJobName	TrainingJobStatus	FinalObjectiveValue	TrainingStartTime	TrainingEndTime	TrainingElapsedTimeSeconds
0	"64"	0.000749	pytorch-training-221104-1658-001-da2c3510	Completed	91.344391	2022-11-04 17:00:20+00:00	2022-11-04 17:11:10+00:00	650.0



Amazon SageMaker > Hyperparameter tuning jobs

Hyperparameter tuning jobs

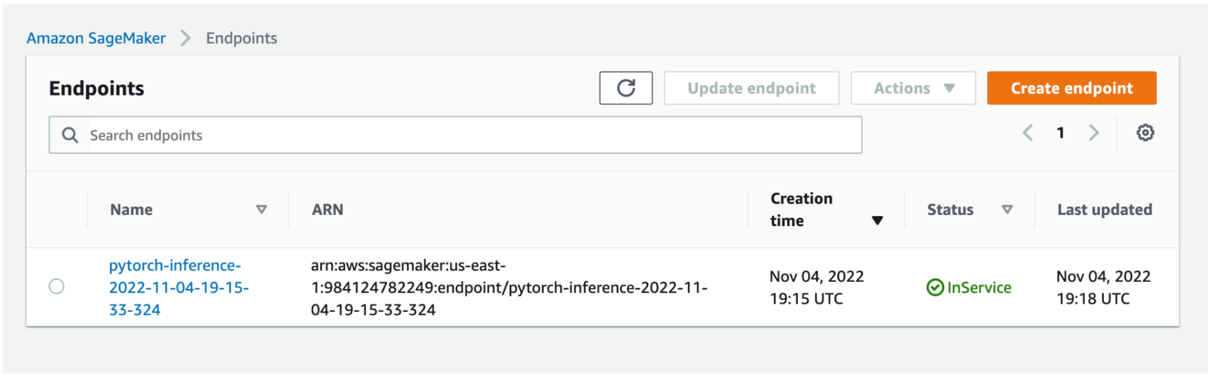
	Name	Status	Training completed/total	Creation time	Duration
○	pytorch-training-221104-1658	Completed	1 / 1	Nov 04, 2022 16:58 UTC	14 minutes

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

4 - Results

4.1 Model Evaluation and Validation

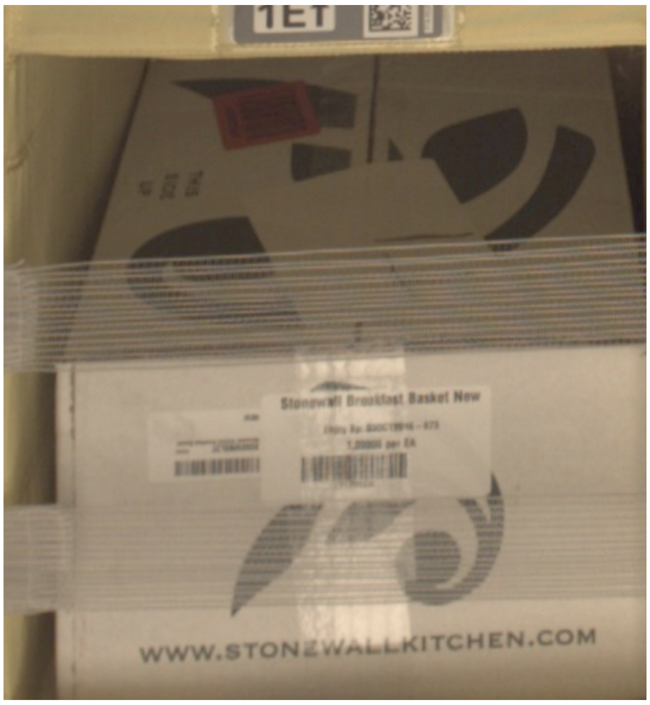
A SageMaker endpoint was deployed using a ml.m5.large instance. Below is the endpoint in sagemaker and the prediction in a test image.


Amazon SageMaker > Endpoints

Endpoints

Name	ARN	Creation time	Status	Last updated
pytorch-inference-2022-11-04-19-15-33-324	arn:aws:sagemaker:us-east-1:984124782249:endpoint/pytorch-inference-2022-11-04-19-15-33-324	Nov 04, 2022 19:15 UTC	InService	Nov 04, 2022 19:18 UTC

Image example:

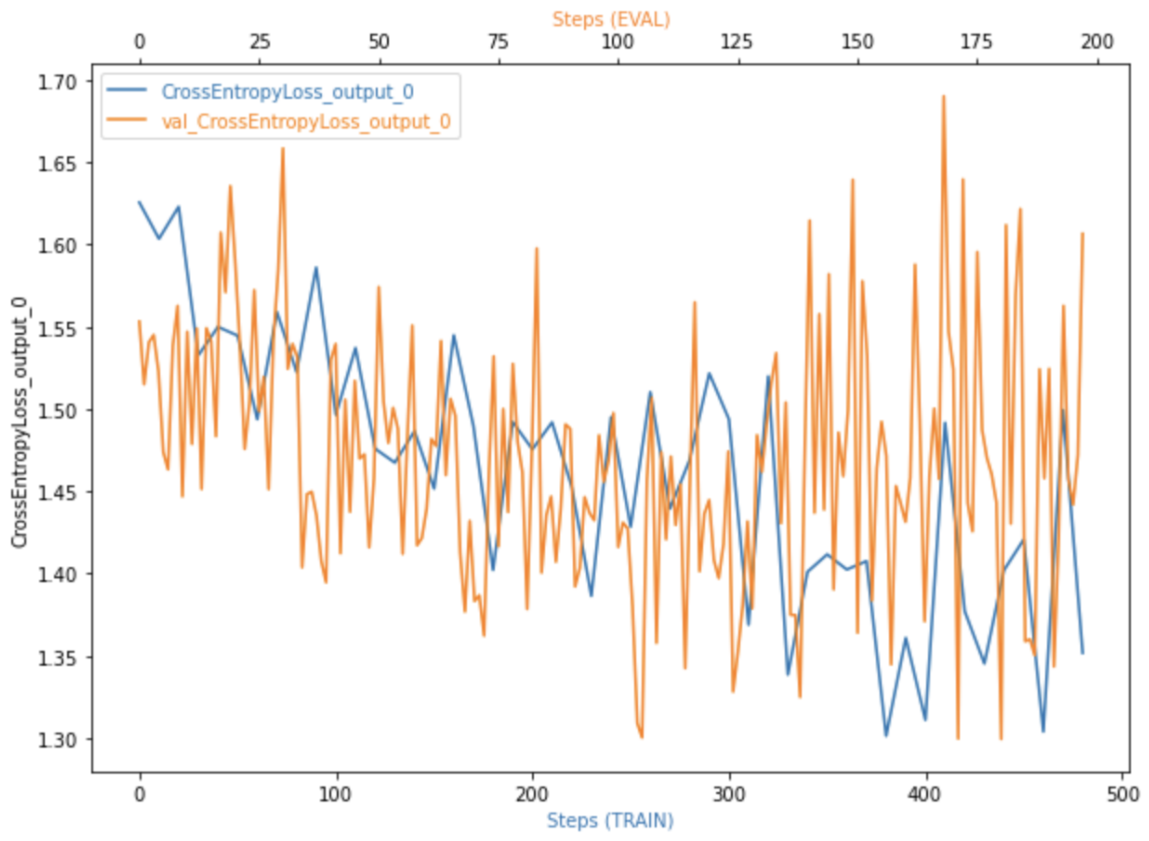


response:

```
[[-3.1894218921661377,
  0.16693083941936493,
  0.7788431644439697,
  0.41088956594467163,
  -0.43211162090301514]]
```

In this example, the model did not predict the right label.

The Debugging hook track the Cross Entropy Loss from training and validation phases as show below:



Seems to have a little overfitting. We could try different neural network architecture or use some regularization like dropout.

4.2 Justification

Because our model used only part of the database, we were unable to beat the benchmark accuracy of 55.67%. However, the result is still considered acceptable. I had to be careful when making the hyperparameters of the model, such as using an instance that has a GPU for a short time (because I developed the work on a personal account). If we had trained the model on the complete database, and had more budget, we would probably get an even better result. But the result achieved is already of great value.

5 - References

[1] Lopienski Kristina, 2021. Warehouse Inventory Management Processes You Need to Know in 2022
[2] Writing Intern, 2018. HISTORY OF INVENTORY MANAGEMENT TECHNOLOGY
[3] Kristina Lopienski, 2021 Warehouse Inventory Management Processes You Need to Know in 2022
[4] amazon, Amazon Bin Image Dataset https://registry.opendata.aws/amazon-bin- imagery/