**Machine Learning Engineer Nanodegree**

**Inventory Monitoring at Distribution Centers Capstone Project**

Fortas Chihabeddine  
April 8th, 2023

**I. Definition**

**Project Overview**

Artificial Intelligence generates value in the warehouse through various technologies, Machine learning uses algorithms to “learn from experience” and make practical decisions for the warehouse. And using computer vision, cameras placed around the warehouse enable end-to-end product tracking.

Amazon Fulfillment Centers are highly active innovation centers that enable Amazon to deliver millions of products to more than 100 countries worldwide using robotics and computer vision technologies.

Distribution centers often use robots to transport items as part of their operations. Items are carried in bins that can hold multiple items. The supply process, from procurement to delivery, is managed by artificial intelligence. Sometimes products are misplaced during processing, resulting in a mismatch between the recorded inventory and the actual items in the bins.

This mismatch can be detected by implementing machine learning technology, which will improve the effectiveness and efficiency of the distribution center. [1]

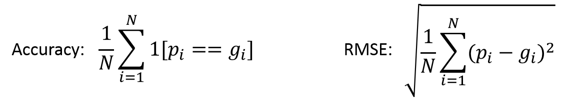
**Problem Statement**

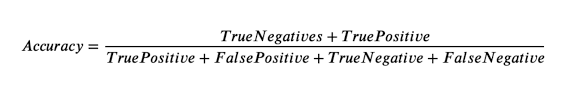
Amazon uses a random storage scheme where items are placed into accessible bins with available space, so the contents of each bin are random, rather than organized by specific product types. Thus, each bin image may show only one type of product or a diverse range of products. Occasionally, items are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

Now, the proposed solution to this project is to start by cleaning and preparing the dataset provided to us by Amazon warehouses that contain approximately 500,000 image of bins, build a model that can detect and count the number of objects in each bin. Then deploy this model as a predictor for inferencing, a system like this can be used to track inventory and make sure that delivery consignments have the correct number of items. [2]

**Metrics**

For the counting task, we will evaluate our model using the standard measures of accuracy (precision). 1 is the indicator function, and p and g are the predictions and ground truth respectively. [3]

[](https://github.com/silverbottlep/abid_challenge/blob/master/figs/eq_metrics.png)

The predictions of the model are evaluated by a standard metric called accuracy for each class, as this is a multi-class image classification problem. This metric is chosen because the classes in the dataset are somewhat unbalanced

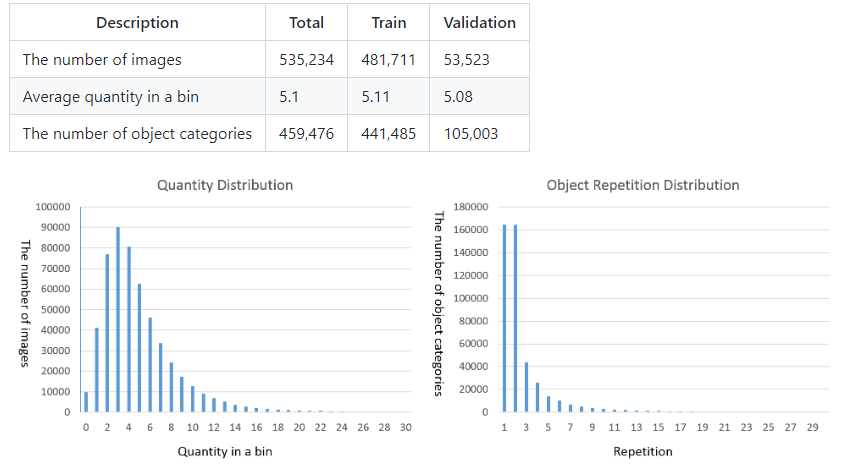
The precision is calculated for each class (number of objects in the bin). The precision indicates the number of images correctly labeled as "x" compared to the total number of images labeled as "x".

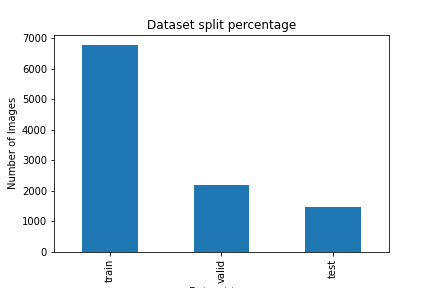
**II. Analysis**

**Data Exploration**

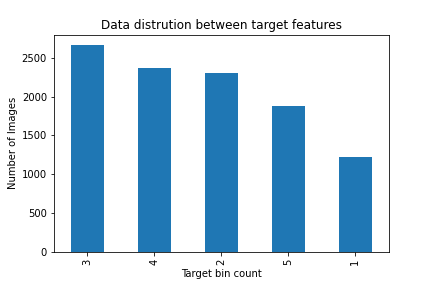
Our dataset is very large (500,000 images). In our project, we will only consider a small part of this dataset, about 10,441 images split between training, validation, and testing, in order to evaluate the performance of the model and then launch a large training job on the whole dataset. [4]

The original dataset:

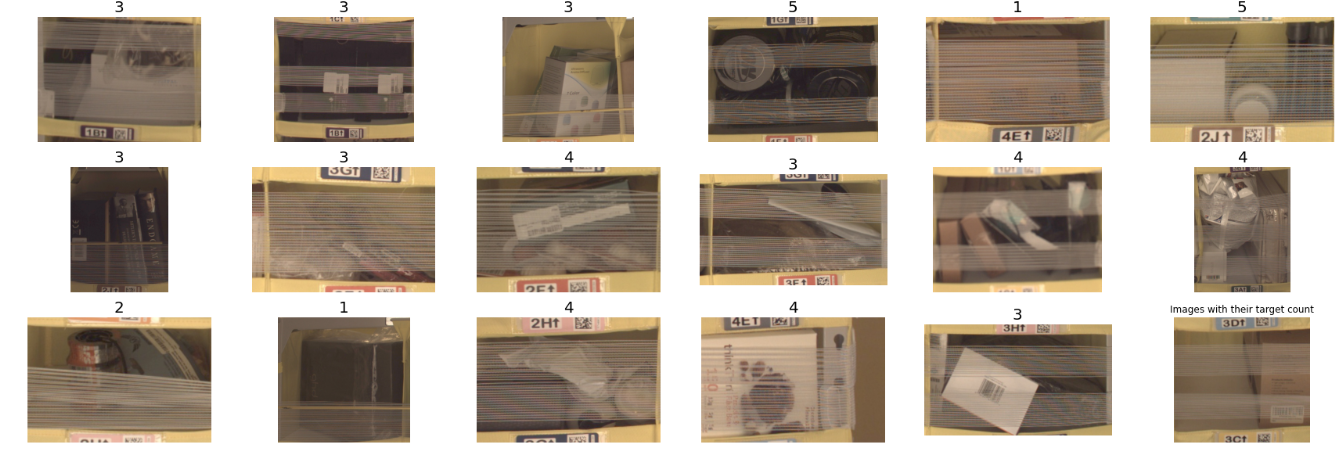


The subset dataset:

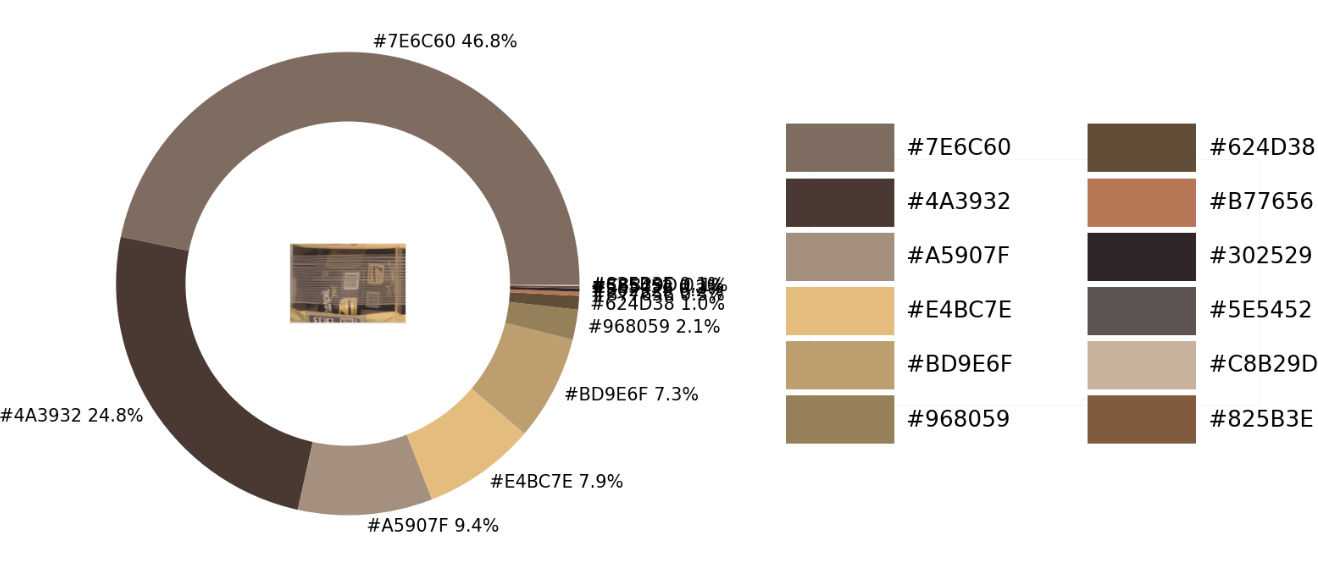
we divided our data into 66% training, 22% validation, and 12% testing.

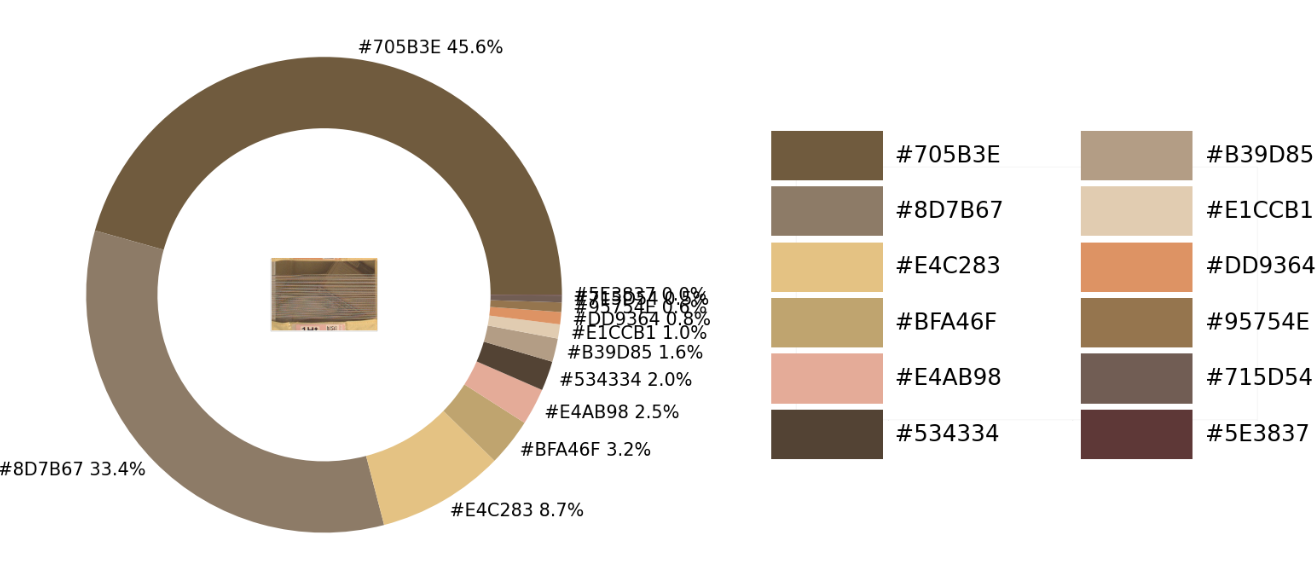


Looking at the graphs above, we see that the number of images is not uniform, which could skew our model because, during training, our model will see more images from one class than others, for example, class "3" has twice as many images as class "1".

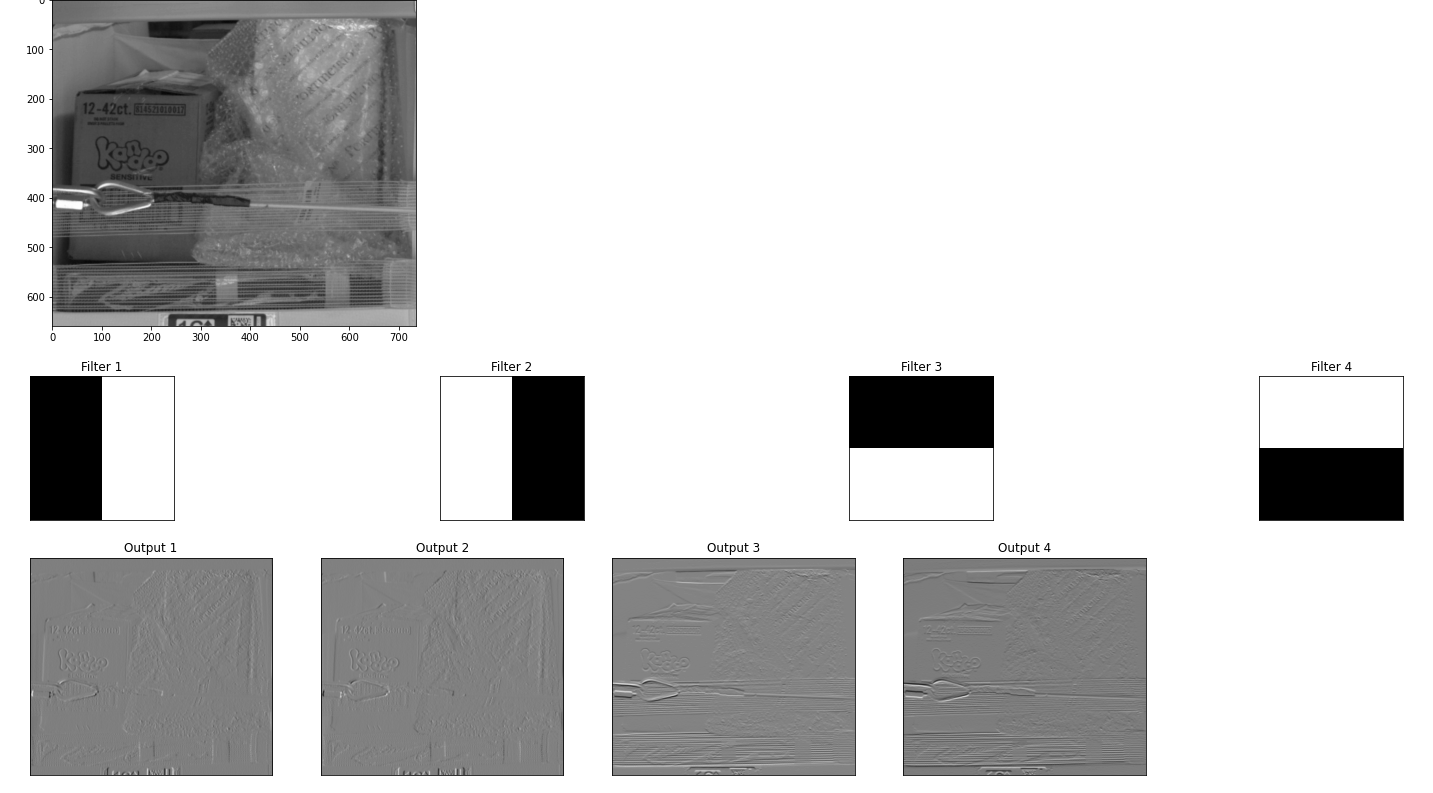
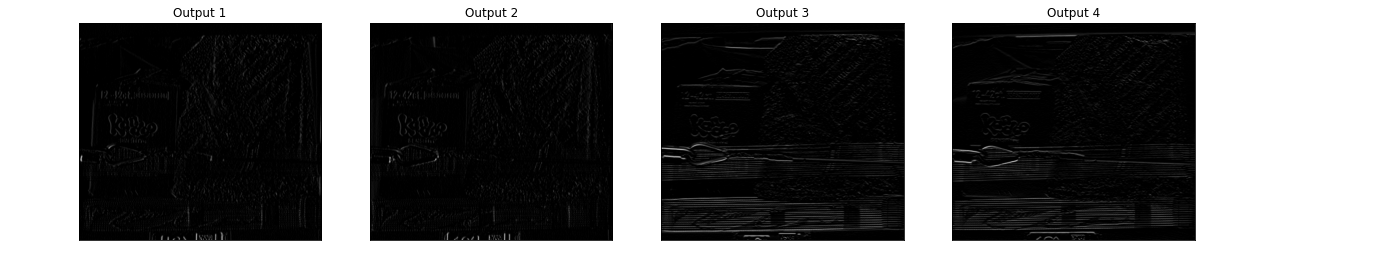
**Exploratory Visualization**

While looking at random images from our dataset, I noticed that the images are a bit difficult to distinguish from the count items because:

* the image colors and saturation
* the presence of packing tape that covers some of the details
* the items are wrapped in the same wrapping paper
* the position and angle of the items (too close together, they can be seen as one item)

We used a method of extracting the dominant color of the image, and we noticed that the dominant color is the color of the wrapper, which will not help our model to train better, because we only need useful information to pass to our model (this method was taken from a this [URL publication](https://towardsdatascience.com/image-color-extraction-with-python-in-4-steps-8d9370d9216e)). 





Viewing our image after it has gone through some CNN filters, we see that some filters make the wrapper ribbon dominant over the other elements, which will make the learning worse, while other filters remove the wrapper ribbon but make the image contain information to be passed to our model. (This method is taken from an Udacity course at this [link](https://github.com/udacity/machine-learning/blob/master/projects/practice_projects/cnn/conv-visualization/conv_visualization.ipynb)).

Finally, since this data set can be passed to our model, I decided to leave it raw and upload it directly to AWS S3 Bucket.

**Algorithms and Techniques**

Image processing is a difficult problem., due to the high diversity and complexity of visual data. This has led to many research activities that lead to CNN, Convolutional Neural Networks are types of neural networks specialized for data processing, especially in the image data space. They give excellent results in computer vision tasks such as image classification, object detection, and image recognition. The pre-trained models can be used and refined for other tasks.

We will use pre-trained CNN models and transfer learning to learn the weights of the last layer of the network. You start with a trained neural network used for image recognition and then tweak it a bit here and there to form a model suitable for our particular use case.

We choose the ResNet model, which is short for Residual Network, is one of the best-performing models for computer vision tasks. To refine our model, we will start with a few more steps:

- Create and load the pre-trained model from the framework database

- Add fully connected layers and train it.

We will start by finding the best hyperparameters by running a hyperparameter tuning task with our pre-trained model. The choice of hyperparameter ranges is arbitrary and the two most important parameters are the learning rate and the batch size.

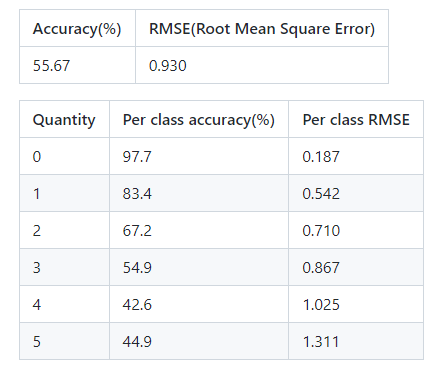
- The learning rate: is very important to speed up the learning process, as a wrong or too low learning rate can lead to overfitting, but a too high rate can also produce non-optimal results.

- Batch size: is also very important as it controls the accuracy of the error gradient estimation when training neural networks.

Since this is a multi-class classification problem, the cross-entropy loss function is used as the cost function. The cross-entropy loss function is used as the cost function and the adaptive moment estimation (Adam) is used as the optimizer.

**Benchmark**

The objective of this project is to obtain the accuracies by class also we should be able to get 55.67% accuracy. These values were derived from the Amazon Bin Image Dataset Challenge. [3]



**III. Methodology**

**Data Preprocessing**

Preprocessing is performed in the part of the training where the data loaders prepare the data for the training, i.e., the following steps:

1. The list of images is shuffled
2. The images are divided into a training set and a validation set.
3. The images are converted to grayscale
4. Images are randomly resized and cropped with horizontal inversion and color variation.
5. The average pixel value is subtracted and the pixel values are divided by the standard deviation of the pixel values (normalization process). note that the average pixel value and the standard deviation are constants.

**Implementation**

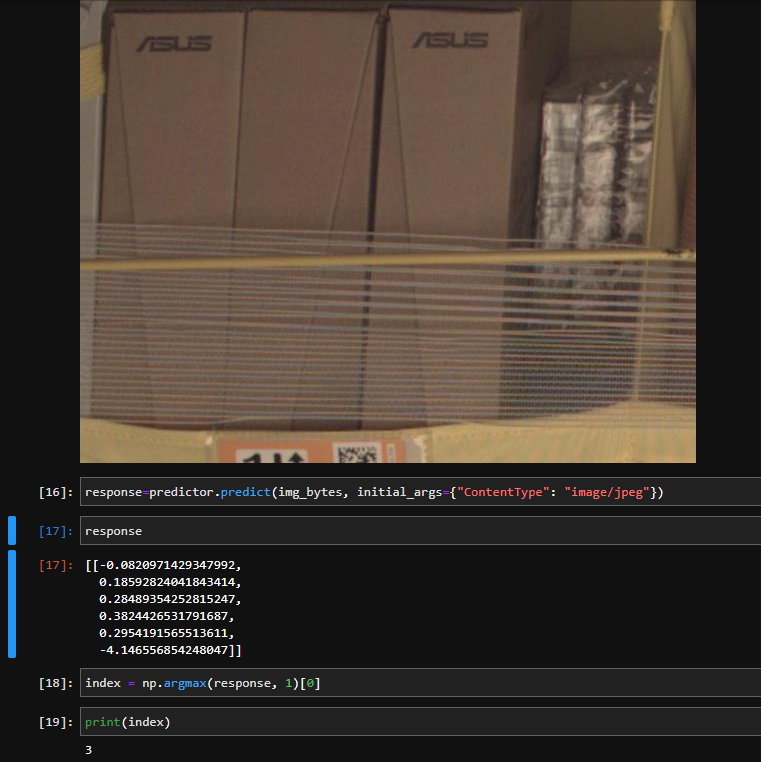
in the first part, we configured some services provided by AWS such as SageMaker, S3 and Elastic Compute 2. These are the services we will use throughout our project. All the essential code for training and inferring our model is implemented using python and PyTorch frameworks. where our pre-trained ResNet model is acquired from the torchvision library. here are the next steps:

1. Setup SageMaker Studio and clone the starter code from the GitHub repo.
2. Next, we create project files such as the training and inference scripts.
3. After uploading the data for analysis and processing, we upload it to S3.
4. To begin training, we first run a hyperparameter fitting task to find the best parameters to train our model with.
5. We then take those parameters and pass them to a training task that also takes debugging and profiling parameters to record specific metrics during training.

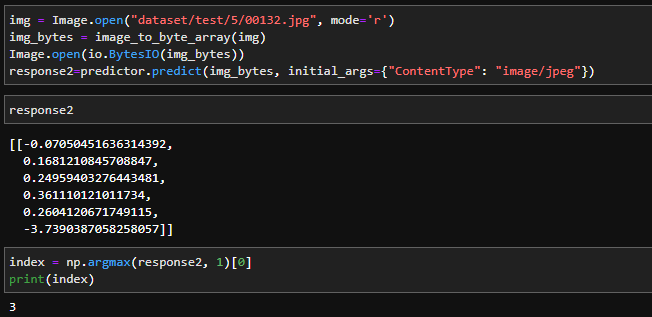
**Refinement**

These are the best hyperparameters returned from the tuning job : ***{'batch\_size': 128, 'learning\_rate': '0.06246976097402943', 'epochs': '11'}***

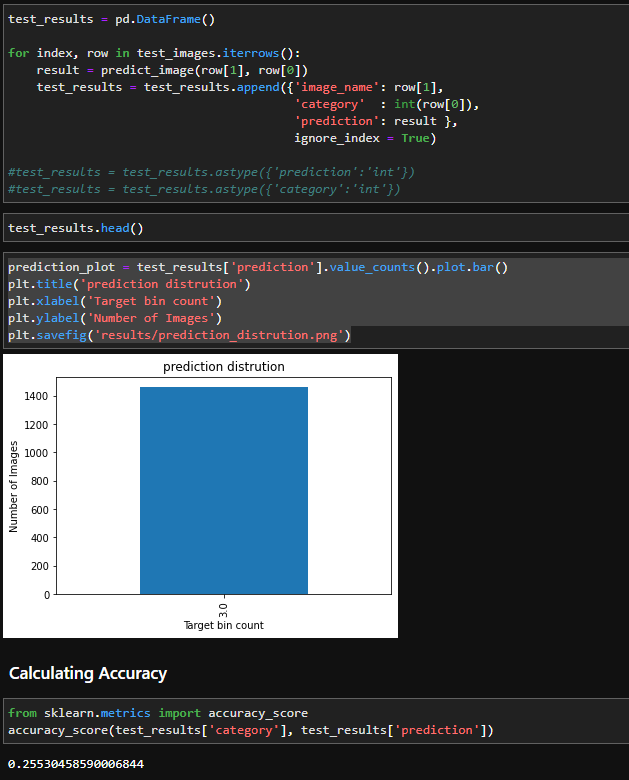
After training and deploying the model for inference, we test the accuracy and performance based on the results of the test dataset. After that, with some code modifications, the training can be done iteratively with different hyperparameters and larger datasets. In this way, we will be able to obtain a more accurate model.

And this is the result of inferencing one test image, and we noticed that the same prediction result was obtained for all other images, regardless of their number of boxes, so our model seems not to have learned to count yet.

We’ve tried another test image and the result is the same bin count “3”



Now when we inference the whole test data and we see the results, now we are sure that our model didn’t learn how to count yet, I think that because the unbalanced dataset where category “3” is dominant above all other categories.



**IV. Results**

**Model Evaluation and Validation**

The final model was deployed with the best hyperparameters. I used the PyTorchModel class to deploy the model. After successfully deploying the model to the endpoint, we started testing it with the test dataset using the PyTorchModel.predict() method

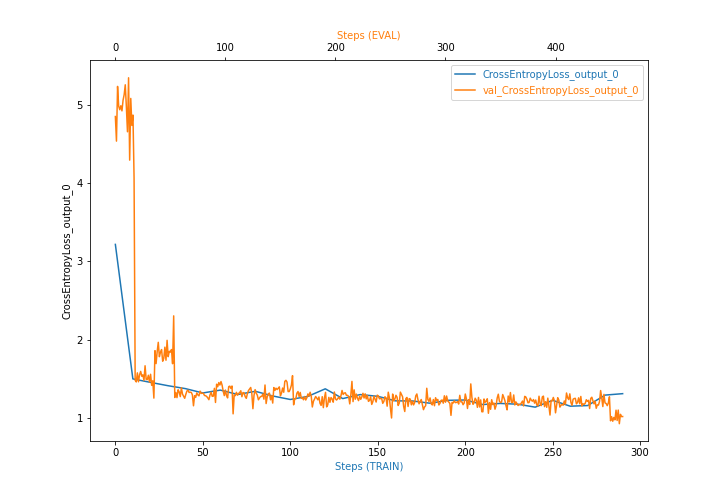
The prediction results are far from the ground truth and the accuracy is too low. Thus, the model has not yet learned to count, needs further training with larger datasets, and also needs to iteratively explore the best performing CNN models and hyperparameters.

**Justification**

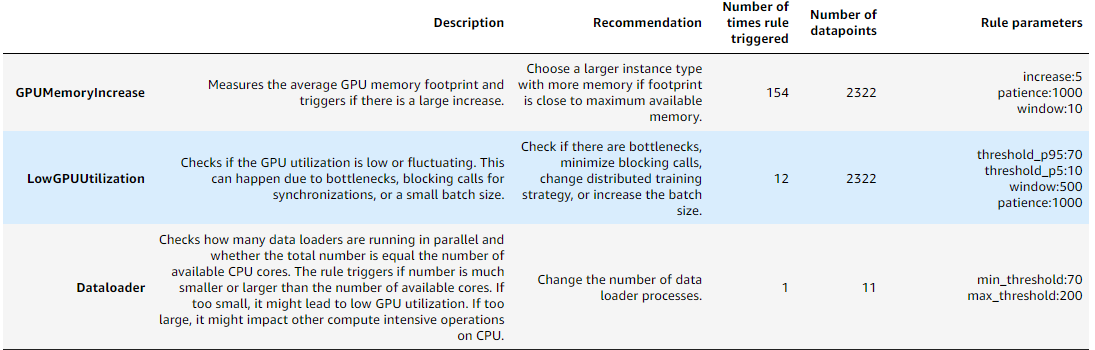
With the small subset of image data, the accuracy of the model reached the saturation point and reached the test accuracy of 25%, which is much lower than the baseline result mentioned earlier. Therefore, the model needs more complex training to distinguish the class and features in the bins.

**V. Conclusion**

**Free-Form Visualization**

classes.

We plotted the cross-entropy loss captured at each epoch during training. and here is the result: the output of our model is fixed between two classes, which is bad because we can see that our model is overfitted and did not learn about the different

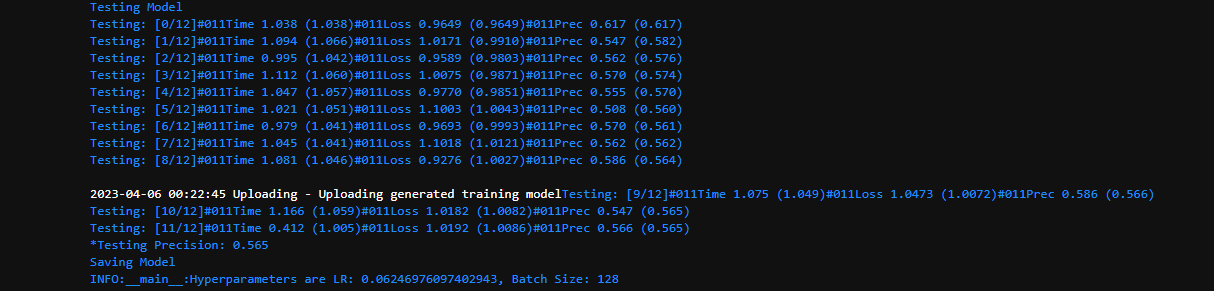


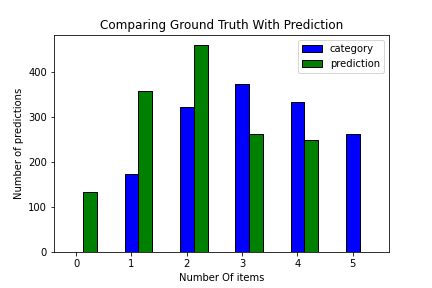
Using debugging and profiling libraries to analyze the training process, looking at the result here, we see that: Maximum and minimum GPU usage is triggered, which means a problem in the data loaders. So we need to modify the data loaders to use the GPU memory consistently.

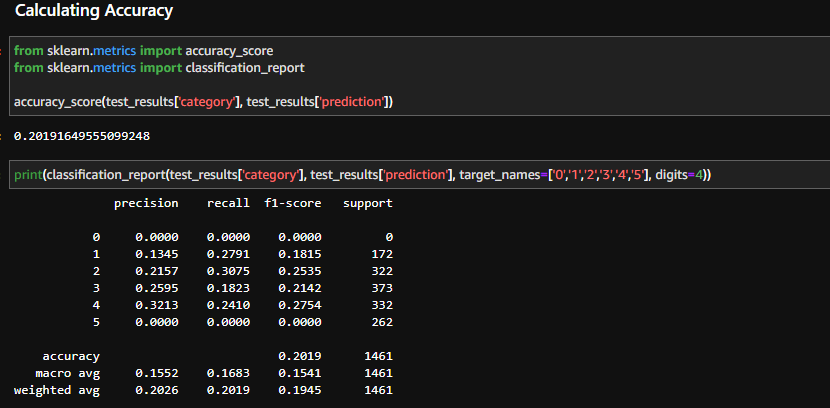
**Improvement**

To achieve the benchmark results required, we going to use a pre-trained model from Amazon Bin Image Dataset (ABID) Challenge, the model is a resnet34 but it was trained in a GPU environment so we need to tweak our code a little bit.

here the model output is 6 classes since a 0 class is added to the dataset, after training we see that the testing accuracy is 56% which is much better and near the benchmark numbers.







We find that the accuracy of our model dropped even when predicting more classes, which means that the model learns diversity but not accuracy, but I think that with a larger data set, different model architectures, different parameters, loss function and optimizer, the model will perform better.

**Reflection**

The process used for this project can be summarized by the following steps:

1. An initial problem and relevant public data sets were found.
2. The data was downloaded and pre-processed (segmented).
3. The data was pre-processed and cleaned, then uploaded to the S3 bin.
4. The model was fitted using the data (using hyperparameter fitting, until a good set of parameters was found).
5. The model was trained with debugging and profiling libraries to extract useful information.
6. Next, the Sagemaker endpoint was set to run inference on the model.

I found steps 4 and 5 the most difficult, as I had to become familiar with the startup code files.

As for the most interesting aspects of the project, I'm very happy to have found the abid\_challenge repo, as I'm sure it will be very useful for future projects/experiments.

# Bibliography

|  |  |
| --- | --- |
| [1] | O. Kholodenko, "MACHINE LEARNING IN SUPPLY CHAIN," 20 December 2022. [Online]. Available: https://codeit.us/blog/machine-learning-in-supply-chain#:~:text=Machine%20learning%20in%20the%20supply,in%20your%20store%20on%20time.. |
| [2] | Amazon, "https://registry.opendata.aws/amazon-bin-imagery/," 2023. [Online]. Available: https://registry.opendata.aws/amazon-bin-imagery/. |
| [3] | silverbottlep, "Amazon Bin Image Dataset(ABID) Challenge," 20 Jully 2017. [Online]. Available: https://github.com/silverbottlep/abid\_challenge. |
| [4] | D. DAVE, "Amazon Bin Image Dataset," 2023. [Online]. Available: https://www.kaggle.com/datasets/dhruvildave/amazon-bin-image-dataset. |