Project report

Project: Amazon Distribution Center
By: Miguel Mayhuire

Definition

Project Overview

This project addresses a use case in Amazon logistics which is the number of items per bin. A solution to use case can leverage problems like orders assignments (is the number of items per order, correct?) or stock planning.
The proposed solution is an endpoint deployed in SageMaker. This endpoint is going to process images of the bins and it is going to respond with the number of items per bin. To do so, a computer vision model is trained and deployed. But before this step, the data is processed, analyzed, and finally ingested to a production machine learning model.

Problem Statement

The project and business need are very clear.

- Incorrect number of items per bin means incorrect order fulfillment and therefore low customer satisfaction.
- Incorrect number of items per bin means that items stock imbalance which means logistic costs and time.
- The strategy and means to deploy a solution are doable. Using Sagemaker capacities and photos coming from the warehouse machines, it is possible to deploy an end-to-end computer vision solution based on data.

Metrics

First, the problem is clearly a supervised machine learning problem with multi-labels. the model in its training loop uses log-loss metric and accuracy for evaluation. Accuracy is the right measure because what we are looking for is high degree of correct prediction for each class.

Analysis

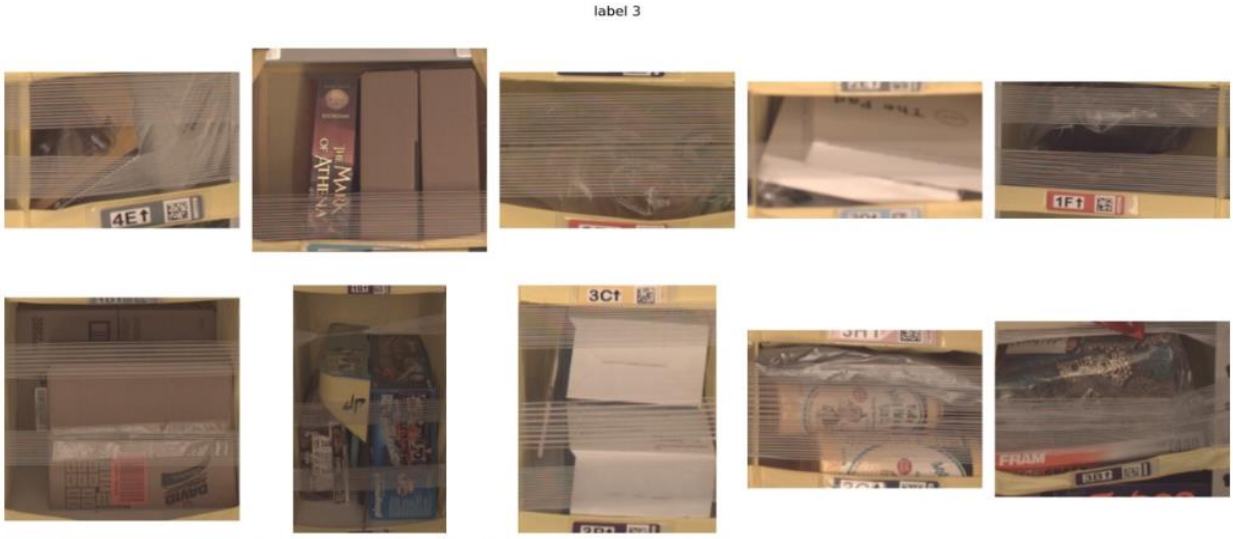
Data Exploration

The data is provided by Amazon and can be found in the following link: https://registry.opendata.aws/amazon-bin-imagery/. The data is provided in a .zip file that has to be unzip. The unzip file have 5 files with names 1,2,3,4 and 5 that correspond the labels or number of items by bin. Each file has different number of images:

- 1 item 1K images
- 2 items 1.8K images
- 3 items 2.3K images
- 4 items 2K images
- 5 items 1.8K images

Exploratory Visualization

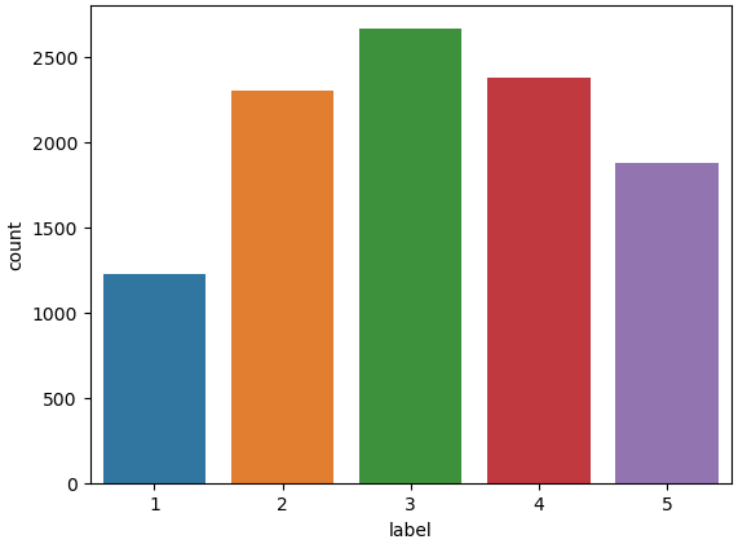
Here I share a sample of images with label 3 or 3 items in a bin:



the images are jpg and some characteristics are:

- images are somewhat dark
- some images seem to be difficult to classify (even using image inspection)
- some images seem to have incorrect labels
-

the distribution of the whole file images is:

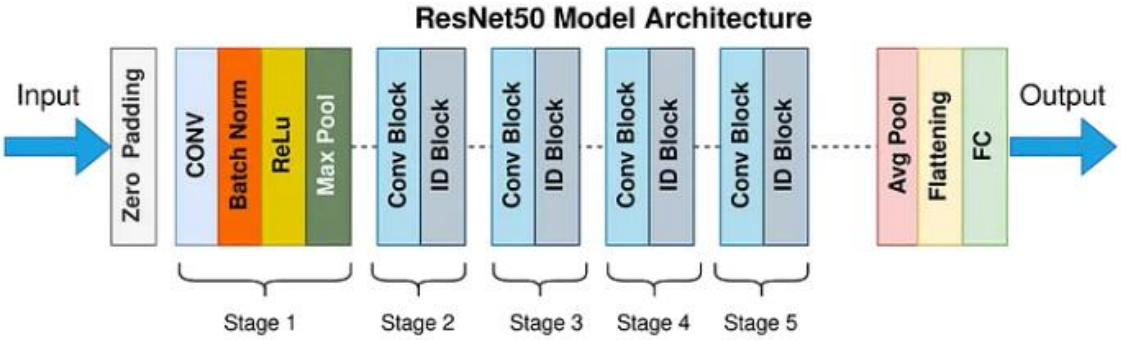


For the purpose of this use case, a stratified sample is taken 10% of validation data and 10% of testing data

Algorithms and Techniques

The choosen model for this task was ResNet50 (pretrained model) plus a custom layer for finetuning.

RestNet is a good model because it is powerful model for computer vision. In short, the architecture relies on a very long chains of convolutional cells, each cell contains convolutional layers from the previous cell that is used as a residual layer that helps to add noise to the CNN and to address overfit



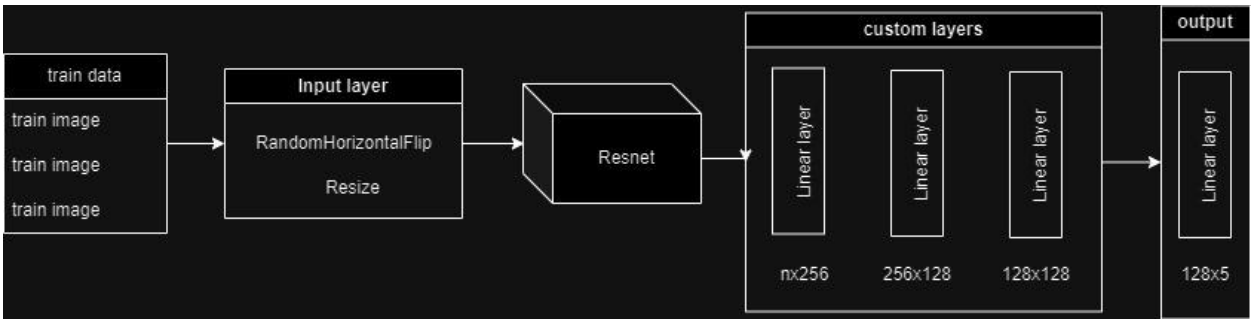
for hyperparameter tunning I used:

- larning rate tuning between 0.001 and 0.1
- batch size options: 16, 32 and 64
- epochs between 2 and 6

Benchmark

In the previous section I described the in general terms what kind of algorithm was suitable for this use case. In this section I am going to explain in depth how the base line model is going to be set.

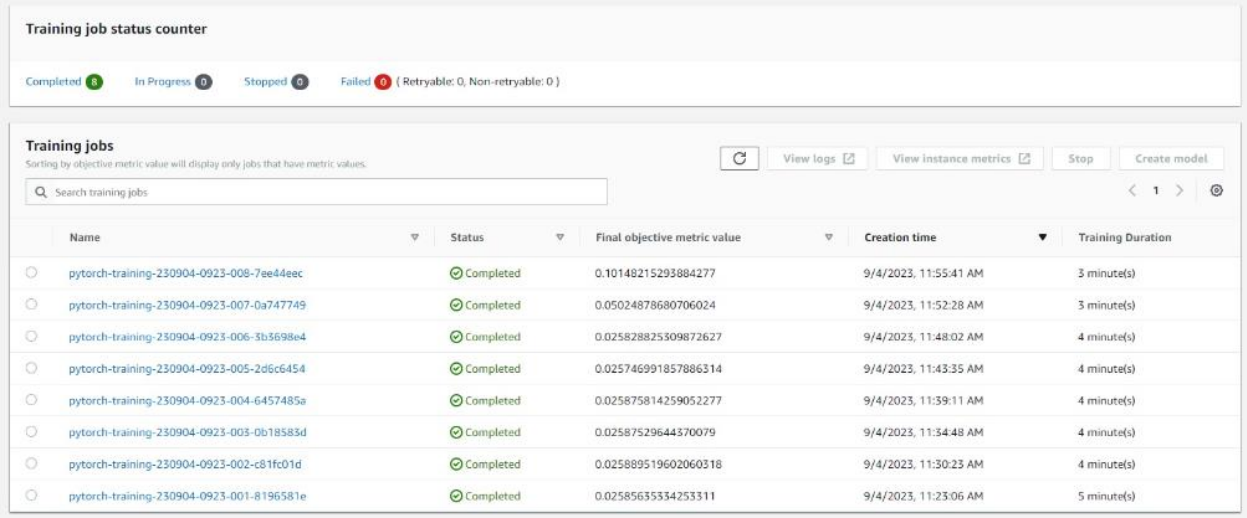
The base architecture of the base line is:



Description:

- first an input layer where there is a simple data augmentation operation called random horizontal flip. After the data augmentation there is a resizing layer that helps the input fit in the next step
- a pretrained Resnet model with fixed parameters or weights
- then the custom layers are composed of 3 linear layers wit activation function Relu. The initial layer is nx256 where n is the number of end notes from the Resnet. The next layers are 256x128 and 128x128
- finally and end layer that takes 128 input form the previous layer and out put 5 because of the number of labels

now that the base architecture is set, best parameters can be found and thanks to hyper parameter tunning job it is possible to train many models and to evaluate then completely:
For this use case I set 8 jobs or models



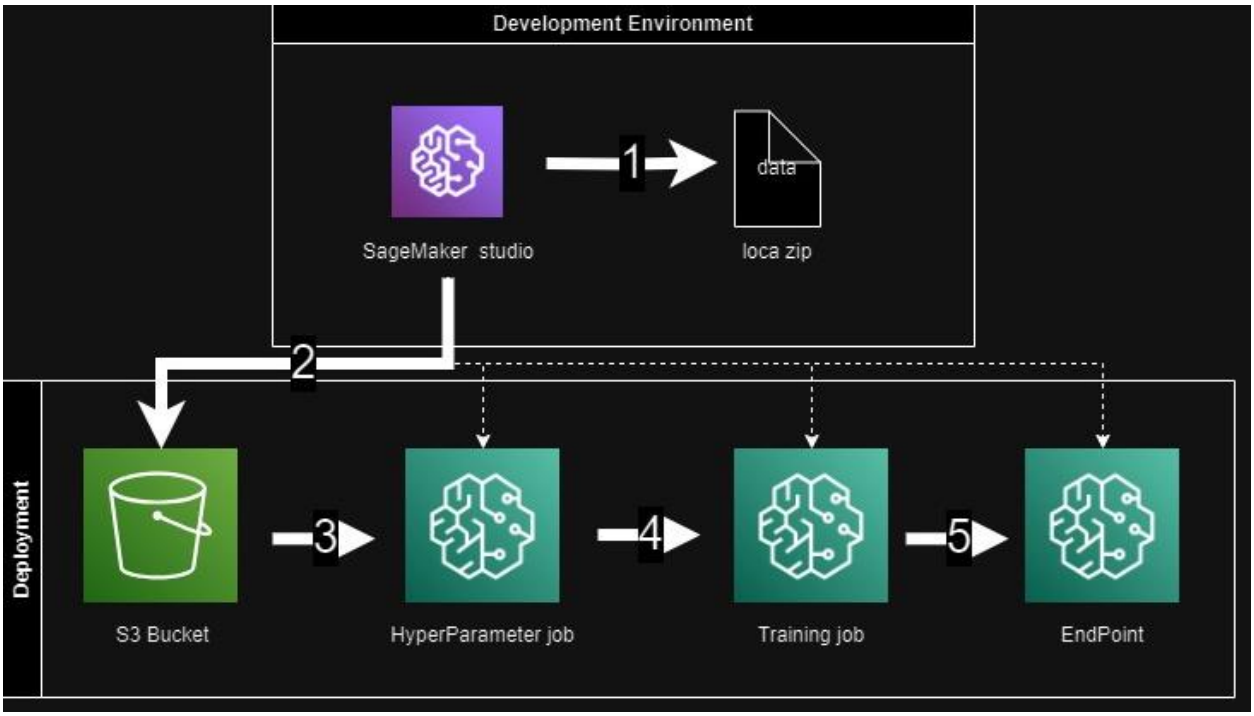
The best hyper parameter for the baseline model is:

'batch_size': 64,
'epochs':2,
'lr': 0.06524108473213124

The criteria to choose the best model was the one with the least log loss error and which log loss error was lower than 0.003 using the evaluation data (part of the training phase). Using 500 random image sample from the test set the accuracy is 0.297

Methodology

Here I share a general view of the full pipeline for this project



Data Preprocessing

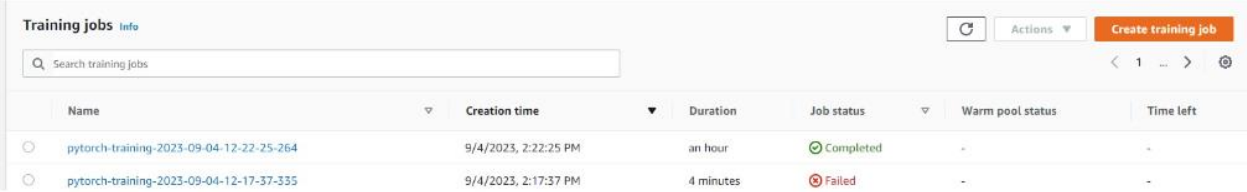
the images are jpg and some characteristics are:
images are somewhat dark
some images seems to be difficult to classify (even using image inspection)
some images seems to have incorrect labels
export the data to s3 such that execute training jobs – In order to run training jobs, data must be in an S3 bucket. The bucket structure has train, validation and test files

Implementation

Search for best hyper parameters – This stage ingests the data from s3 and trains different models using different parameter settings. Note that to improve execution time, multi-instance was set and it helped to improve execution time

Best parameters:
'batch_size': 64,
'epochs':2,
'learning rate': 0.06524108473213124

Model training – using the best parameter training jobs where launched:



The first training job failed because there were some problems with code for validation. Once the problem was resolved, a next training job was successfully executed. Here I share some results:

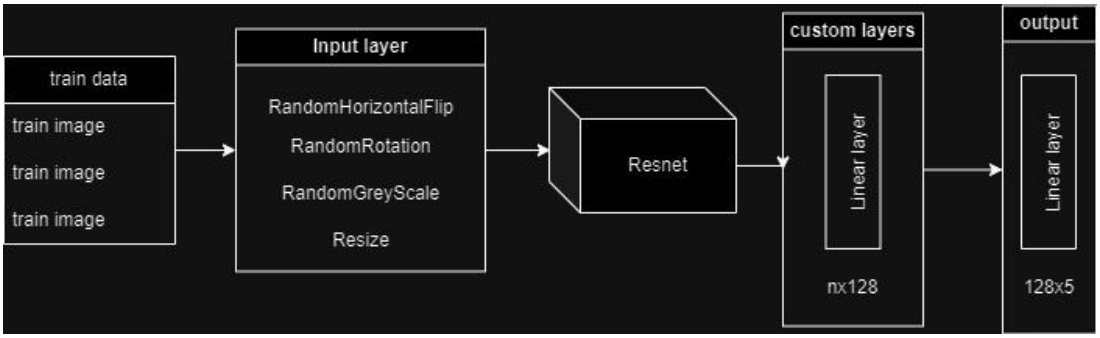
```
PoorWeightInitialization: IssuesFound
Validation set: Average accuracy: 33.42911877394636%
Validation set: Average accuracy: 34.2911877394636%
Validation set: Average accuracy: 30.076628352490 42%
Validation set: Average accuracy: 32.662835249042146%
Validation set: Average accuracy: 29.214559386973182%
Validation set: Average accuracy: 32.56704980842912%
Test set: Average loss: 0.02433362703003952, Accuracy: 29.856459330143544%
Saving the model
```

- profiler results
    - 1 training hour
    - just 30% of the execution time is spent in the validation and training then the rest of the time is spent in other tasks
    - of the training time, the model is spending mainly in executing the convolutional layers.
- Debugger:
    - Poor weight initialization was found
- Validation
    - Validation accuracy is low even though log loss is low

Refinement

- Model performance can improve if more training data is provided and further arechitectures are tested
- Model training time can be improved if GPU resources are used
- Maybe vanishing gradient hooks are making the task to take longer, (because the machine is assessing if the model weight optimization is doing well) I would suggest to remove this hook and see if the model trains faster

A new architecture was set to compete with the baseline model:



The changes:

- Further data augmentation layers in the input layer. Random rotation (goal generalize better the item distribution inside a bin), random grey scale (increase the samples with low light contrast)
- The custom layer is shortened and just a single layer is used Nx128 meaning that just a layer of 128 nodes are trained to make this block of layers less complex
- Epocs for hyper parameter tunning is set 4 and 20 epochs goal is to train the model many more times compered to the baseline model

After hyperparameter tunning using the new architecture, the best parameters for this model are:

'batch_size': 64,
'epochs': 20,
'lr': 0.09980513226333607

Using 100 random images from the test data, the accuracy score is 33.8%

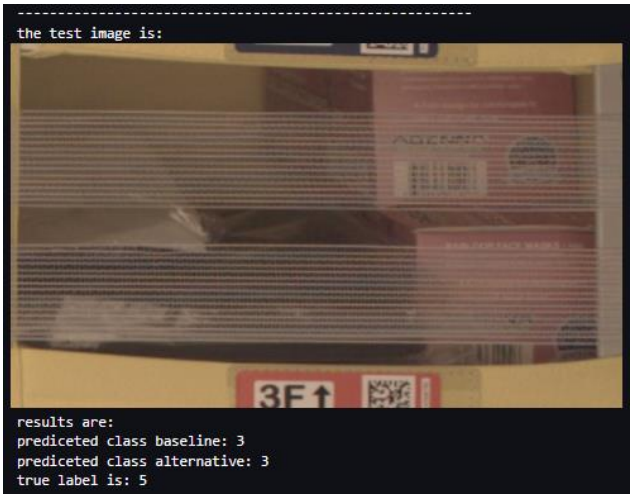
Results

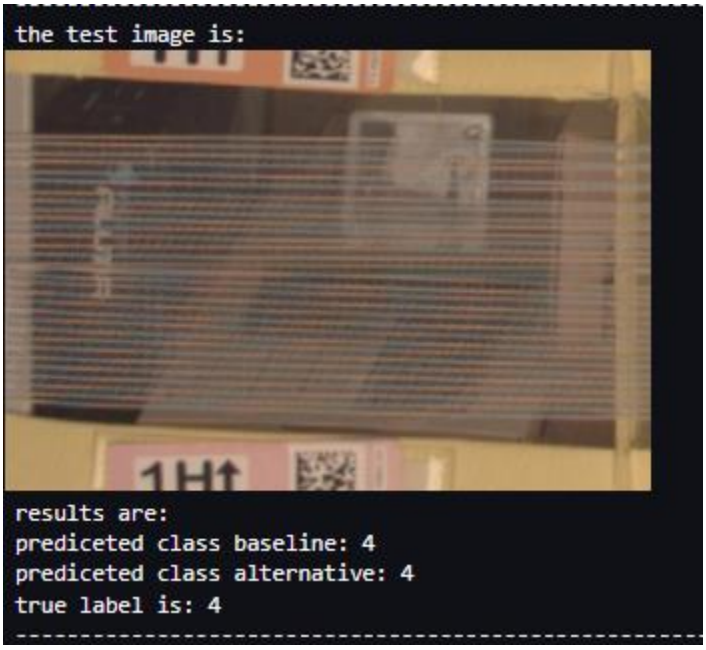
Model Evaluation and Validation

Comparative table:

Items	**Baseline**	**Alternative**
Data augmentation	Horizontal Flip	Horizontal Flip, rotation and grey color sclaing
Epochs	2	20
Learning rate	0.065	0.0998
Custom Layers	3	1
Training time	1 hrs	6.3 hrs
Sample Accuracy	0.297	0.338

It seems that the alternative model is better than the base line model, lets do some inspections:





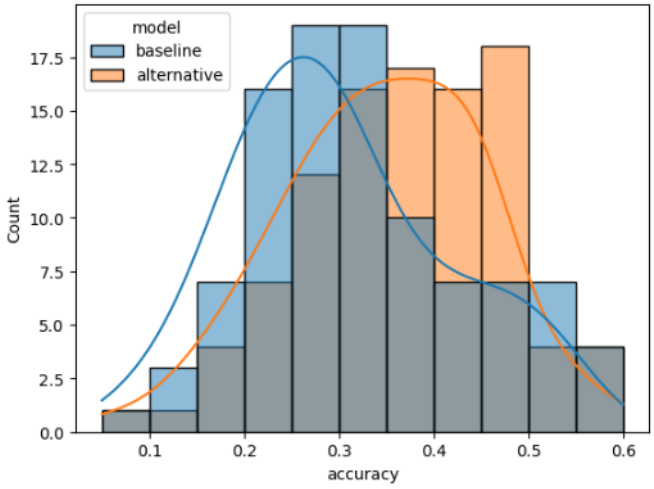
What can be spotted in both models:

- Still general accuracy is low
- Dissimilar predictions
- Endpoint response time is fairly fast

To be more rigorous with the evaluation of both models I will run some tests in the following section.

Justification

Here I share some statistical analysis of both models, baseline and alternative model. I took 100 random samples with replacement of 10 images (bootstrapping) for both models and the results are:



Metric	baseline	alternative
Mean accuracy	**0.30**	**0.345**
Standard deviation	**0.115**	**0.106**

T test p value: 0.00516 (alpha 0.05) therefore means are different

Thanks to bootstrapping and t test for two population over accuracy metrics on different sample test data we see that:

- Alternative model has better accuracy
- And t test demonstrate that this value is statistically different than the baseline accuracy 0.30

Therefore we can conclude that the alternative model is a best model than the baseline.