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

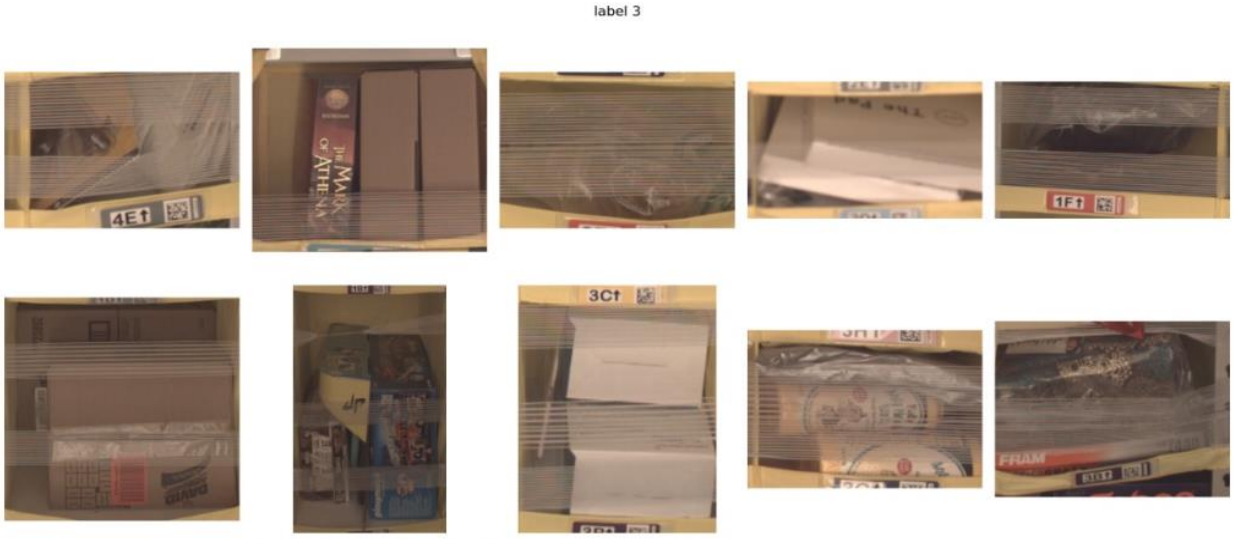
Data Exploration

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

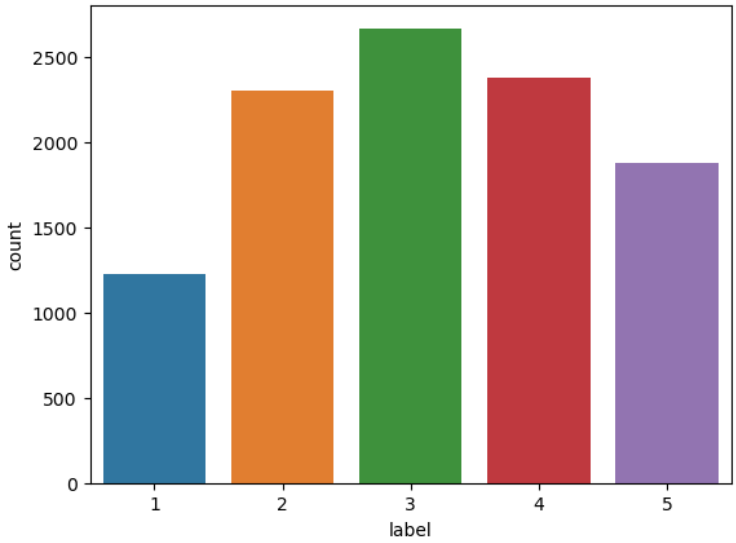
Here I share a sample of images with label 3 or 3 items in a bin:



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:

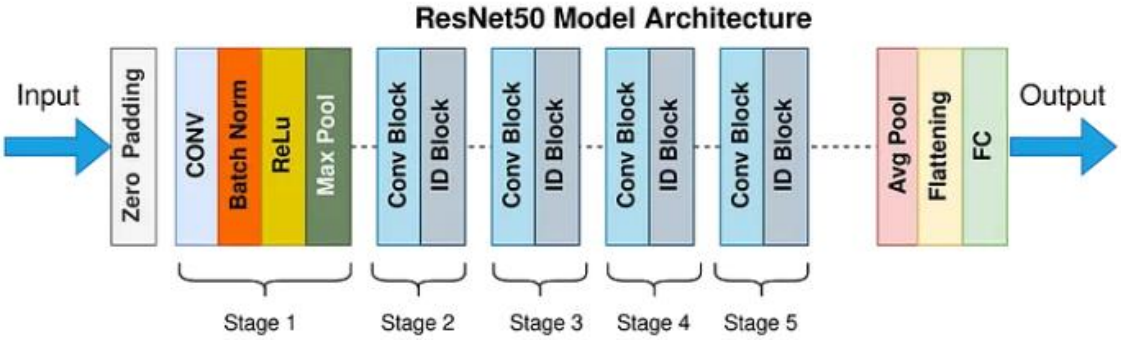


For the purpose of this use case, a stratified sample is taken 10% of validation data and 10% of testing data

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

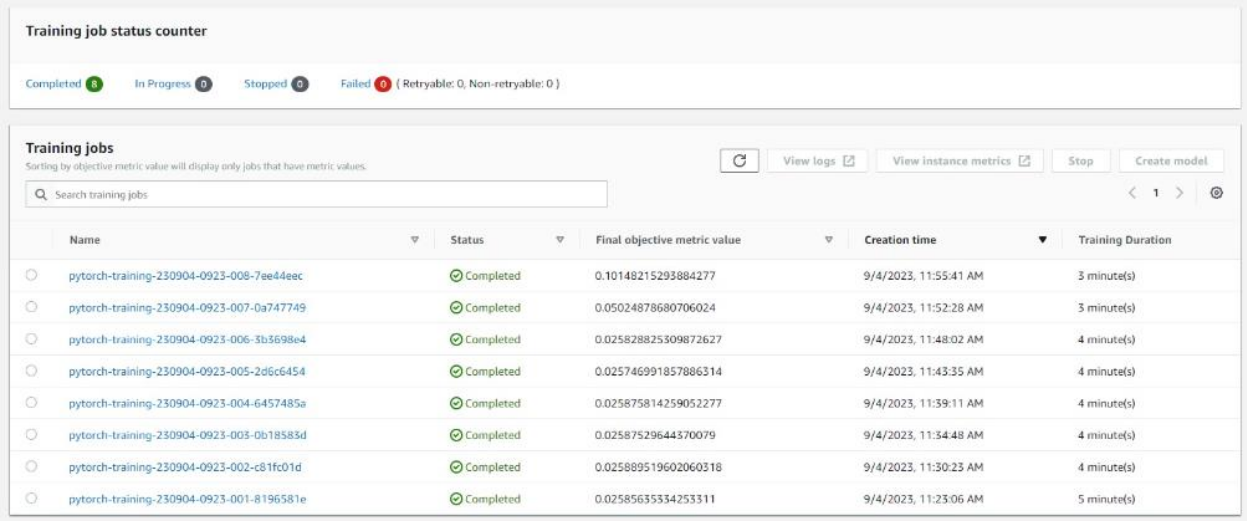


for hyperparameter tunning I used:

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

Benchmark

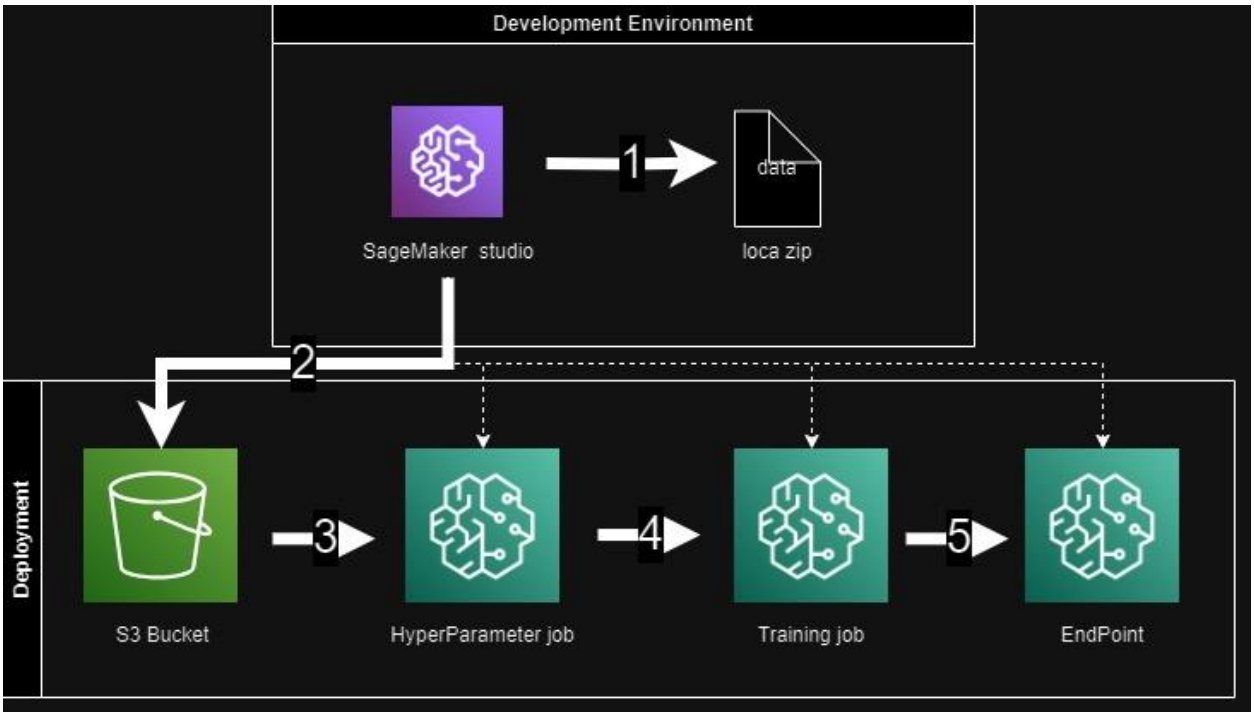
Thanks to hyper parameter tunning job it is possible to train many models and to evaluate then completely:
For this use case I set 8 jobs or models



The criteria to choose the best model was the one with the least log loss error and and which log loss error was lower than 0.003.

Methodology

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



Data Preprocessing

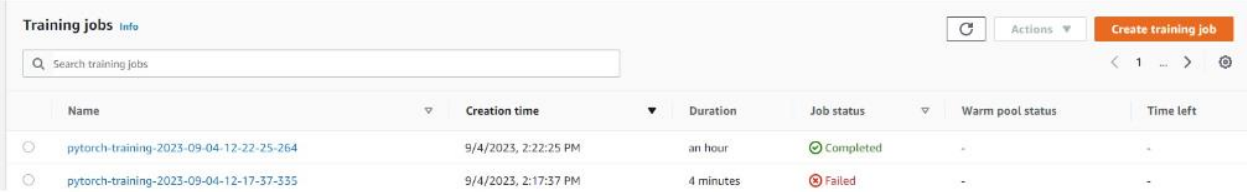
the images are jpg and some characteristics are:
images are somewhat dark
some images seems to be difficult to classify (even using image inspection)
some images seems to have incorrect labels
export the data to s3 such that execute training jobs – In order to run training jobs, data must be in an S3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

Implementation

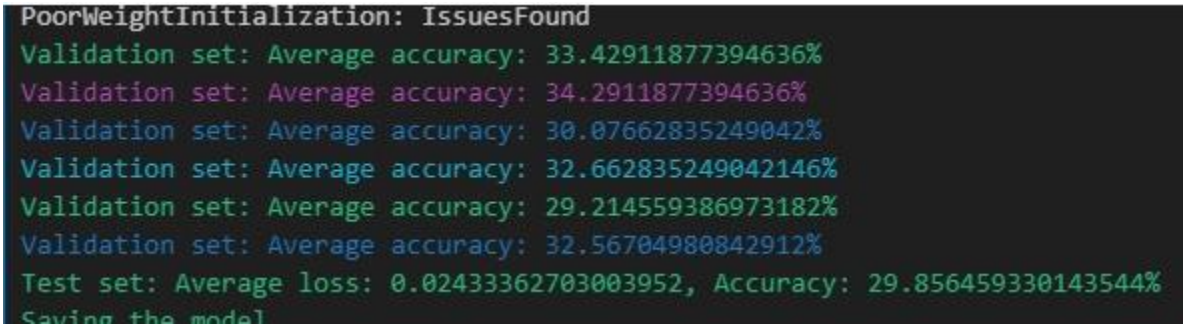
Search for best hyper parameters – This stage ingests the data from s3 and trains different models using different parameter settings. Note that to improve execution time, multi-instance was set and it helped to improve execution time

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:



- 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

Results

Model Evaluation and Validation

After some further training, an acceptable model was reached an then deployed:

The modifications that were done:

- Reduce the number of layers to be finetuned.
- Initialize weights using distributions.
- Add an additional layer in the data augmentation step.

Justification

The new model improved general accuracy in 7 points (from 33 baseline model to 40) in validation data.