Capstone Project

Project: Amazon Distribution Center

By: Miguel Mayhuire

Domain Background

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.

timeline and roadmap, data collection and analysis can be done 2 days, just after the kick off the project proposal. Then hyper parameter tuning, and model training can take place in one day and finally deployment and testing can take another day. Before to expose the model to out of sample data, some extra testing would be required that the endpoint would respond effectively with live data.
Some good practices for production solutions can be found in this official documentation from SageMaker: https://aws.amazon.com/blogs/machine-learning/mlops-deployment-best-practices-for-real-time-inference-model-serving-endpoints-with-amazon-sagemaker/.
From this documentation a good practice is to single expose the end point as shadow service. The goal is to evaluate how the model and the endpoint behaves with live traffic and which errors appear during this phase before the recommendations or predictions impact other services.

Finally, my motivation to address this topic is to gain more practice with computer vision models and finetuning pretrained models. I considered the last point quite important because never before power full models are available to the public meaning that there is less and less need to have powerful resources to get powerful models.

References:

- https://aws.amazon.com/blogs/machine-learning/mlops-deployment-best-practices-for-real-time-inference-model-serving-endpoints-with-amazon-sagemaker/
- https://sagemaker.readthedocs.io/en/stable/frameworks/pytorch/using_pytorch.html#deploy-endpoints-from-model-data.
- https://ultralytics.com/yolov5

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.

Solution statement

For this specific use case it is possible to get a quick solution thanks to AWS services and data availability.

- Data: Data is provided in jpg format and is already labeled (which very often is not the case and makes problems very hard to solve). In addition, there is the possibility to get more data from Amazon given that it is a public dataset.
- AWS provide enough computing resources and infra to, train, evaluate and deploy machine learning models. In addition, it provides intuitive configuration for end points such as increase the number of instances in case of high traffic. This is quite critical because it means that AWS takes care of the end points.
- This solution is somewhat replicable. Data is publicly available, code is available, and pretrained model is available too. However, the production part is only replicable in AWS SageMaker environment.

Datasets and Inputs

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 1.8 images

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

first it is processed in the local environment, a splitting job is launched (train/validation/test) and finally uploaded to S3 such that training jobs can ingest it.

Benchmark Model

Now that data is split and available in S3, it is time to train some models starting for hyper parameter tunning. The base model architecture is a Resnet base model that is finetuned for 5 classes. Once hyperparameter tuning job finished a definitive benchmark can be training with the results of the hyper parameter tuning.
The choice model for this use case was to finetune a pretrained model because it is less expensive and faster to train. Training a full architecture from scratch means lots of time in experimenting with the right architecture, training and high exposure to do not finish the training and deployment in time.

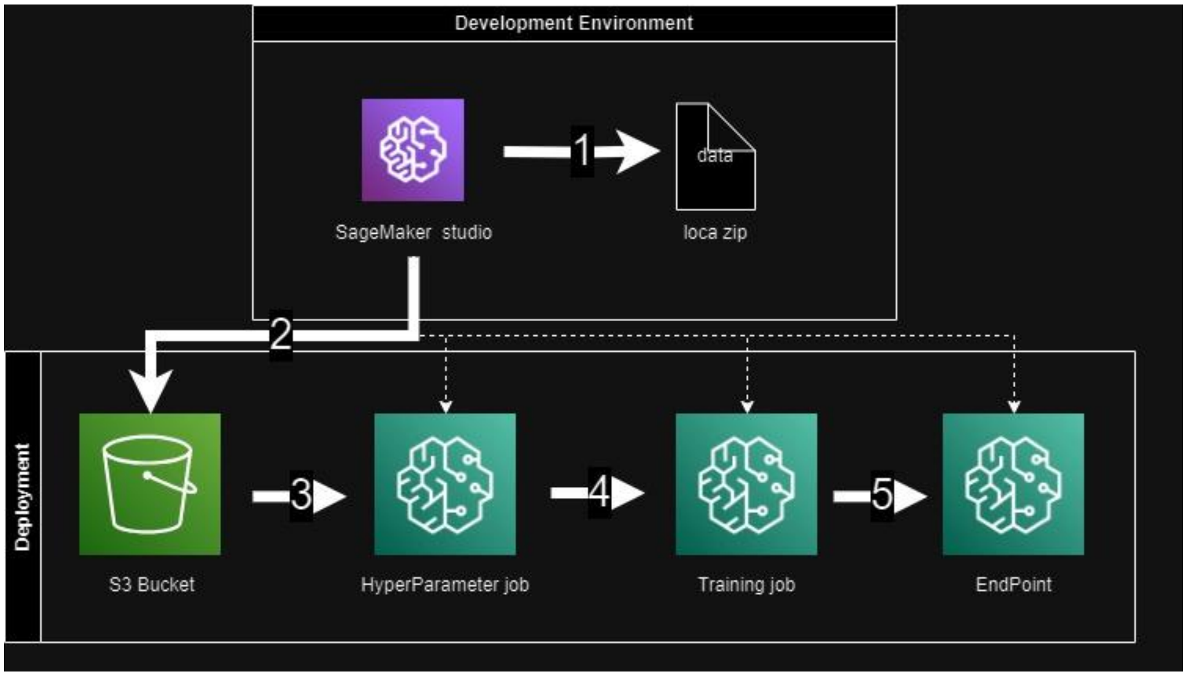
Given that the data is already label, the model can be assessed quickly.

Evaluation 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.

Project Design

The full workflow for this use case is summarized in the following image:



1. treat the data in local – this part is done in Sage Maker studio where the data is downloaded in local, analyzed processed and then sent to an S3 bucket
2. 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

3. search for best hyper parameters – This stage ingest 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
4. train the final model – once the previous job is done. We recover the best parameter to train the official baseline
5. deploy model - once the model is trained and saved, it is time to deployed it, but before, some configuration are needed so that the model can read production data. Some configurations are call the model, resize the images, and validate that the input data is in the right format.