AWS MACHINE LEARNING ENGINEER

CAPSTONE PROPOSAL

Domaine background

User experience and satisfaction are some of the main obsessions of contemporary retail/delivery companies. Any existing delivery company is expected to preserve, or better, to enhance the speed and consistency of their services.

Distribution centers can face large volumes of orders and strict time constraints to fulfill them. Those situations can lead to numerous errors, sometimes hard to notice. As a consequence, distribution centers need to be equipped with multiple means for visual/sensorial quality verification and, when required, restoration.

Due to the advancements in machine learning models and their promising results for labeling tasks, automated quality assessments (or anomaly detections) are getting increasingly adopted in retail/production companies (Redi et al., 2013). Images or weight/length measurements of the product are fed to a model in real-time. If the model sees an anomaly or its predictions are in a mismatch with the expected content, it would create an alarm so that the problem can be properly mitigated.

Problem statement

In this project, we will address one of the problems Amazon fulfillment centers can face.

Amazon fulfillment centers use robots to carry clients' orders in bins. Each bin can contain multiple items. Occasionally, items are misplaced due to human errors, resulting in a mismatch: the recorded bin inventory, versus its actual content.

One basic error that could easily be noticed is if the count of every object instance in the bin is greater or inferior to the expected number.

In this project, we will investigate how can computer vision/machine learning help recognize this error. In other words, we will investigate how we can count the number of objects in a bin with a simple picture of its content.

Dataset

To complete this project we will be using a subset of the Amazon Bin Image Dataset. This subset contains 10441 bins images of different sizes containing one or more objects. For each image, there is a metadata file containing information about the image like the number of objects, their dimension, and the type of object.

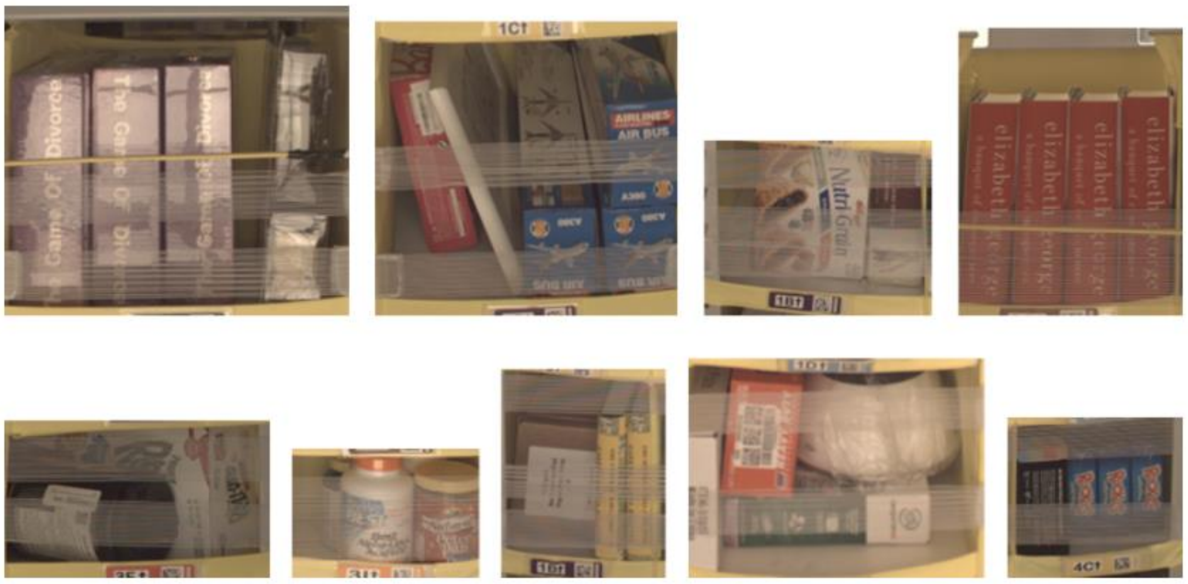

Figure 1: Sample images of the Amazon Bin Image Dataset.

The dataset is distributed within the following classes…

-1 object: 1228 images (~12%)

-2 objects: 2299 images (~22%)

-3 objects: 2666 images (~25%)

-4 objects: 2373 images (~23%)

-5 objects: 1875 images (~0.17%)

…which makes it relatively balanced.

Because our goal is to simply classify the images in one of the five categories, much of the images metada won't be used. Instead, they will simply be placed in folders with the name of the class to be easily fed to a Pytorch datalogger.

To look out and possibly mitigate overfitting, the dataset will be split into train, validation, and test dataset (accounting for respectively 70/20/10%).

Solution statement

1) Data preparation

The images are stored in a public S3 bucket which can be accessed via a list of addresses in a JSON format (provided by Udacity). Those images will be downloaded locally and uploaded to our default bucket.

2) Model choice and hyperparameter optimization

Using Autogluon, several classification models and hyperparameters will be investigated. Autogluon automatically trains many models with different hyperparameter configurations and returns the model that achieved the highest level of accuracy.

Under the hood, several models such as alexnet, resnet or vgg, are trained using transfer learning.

The result of this training (or "exploration") job will be the best performing model architecture and its training hyperparameters (batch-size, learning rate…).

3) Model Training

The previously obtained model and its associated hyperparameters will then be trained, from scratch, using a Pytorch estimator.

Training from scratch modern image models can be computationally demanding. A p3 instance will be used to increase training speed. P3 instances come with several GPUs, to exploit those resources as efficiently as possible, data distributed training will be performed.

A profiler configuration will be also attached to the estimator to detect any major training errors/complications.

4) Model deployment

Finally, the model will be made accessible via a deployed endpoint.

A data capture configuration will be attached to the model to save inference information (input, output, and other metadata). It will help keep track of the model prediction's confidence over time.

Amazon Elastic Inference will be used to optimize inference speed cost-effectively. A low-cost GPU-powered acceleration will be attached to the deployed EC2 instance. This configuration tends to reduce costs up to 75% compared to traditional GPU instances.

Additionally, a lambda function will be set up to receive an image in bytes format, invoke the endpoint, and return the prediction.

A few predictions will finally be performed (invoking the lambda function) to ensure that the created structure is working properly.

Evaluation metrics

Autogluon models can be ranked on the accuracy or a mixt between accuracy and inference speed. As we don't have any latency restrictions, we will focus exclusively on the validation set accuracy.

Our final model (trained from scratch) will be evaluated on its accuracy on the test set.

Benchmark

Other researchers/ML engineers already proposed solutions to the problem. In his approach, silverbottlep obtained a 55% accuracy score (on test set) using a resnet-34, trained from scratch.

Those results seem quite poor (especially when the model was trained on the full dataset), and may already hint at the limitations of image classification models for counting tasks (in opposition to object detection models).

Nonetheless, Silverbottlep's score will be chosen as a benchmark to assess our model performance.

Project design

All steps that will be performed to select, train, assess, and deploy the model are described in the solution statement section.

Those steps will be performed within a SageMaker studio notebook instance (except for the model training part).

Screenshots of the different deployed microservices will be provided in the final report.

Bonus step:

To automate future model training, assessment, and deployment, a SageMaker pipeline will be created.

When executed, our selected model will be trained from data located on a specific S3 address. Then, its accuracy will be calculated on a test dataset. If the new model performed better than the currently deployed model, the endpoint will be updated with the new model.

Here is a schematic of the pipeline:

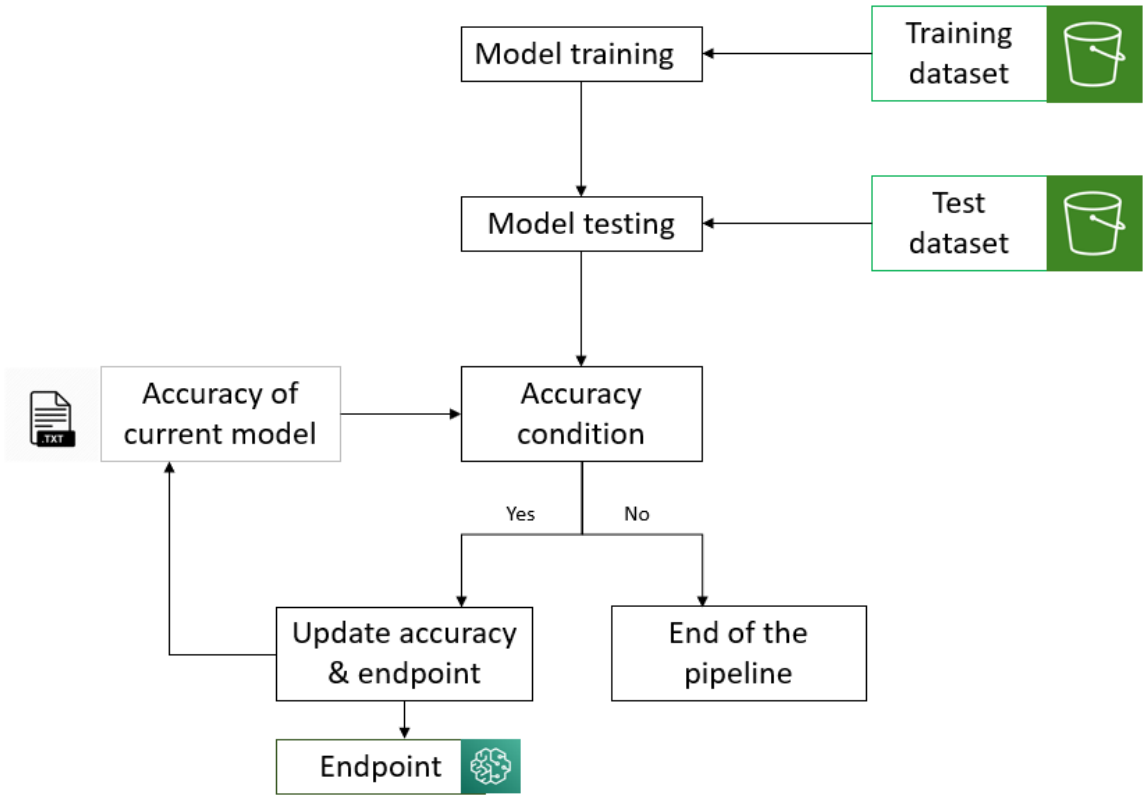

Figure 2: SageMaker pipeline schematic.

Redi, J., Gastaldo, P., & zunino, r. (2013). Supporting visual quality assessment with machine learning. *EURASIP Journal on Image and Video Processing, 2013*, 1-15. https://doi.org/10.1186/1687-5281-2013-54