Amazon Bin Image Dataset(ABID) Challenge

Project Overview

The Huge companies like Amazon all over the world usually move objects from place to another and those companies got many objects to be moved so they move the objects using bins which have many objects inside and each object has a number so the company could keep track of the object and make sure that everything is going well .

The Amazon Bin Image Dataset contains 50,000 images and metadata from bins of a pod in an operating Amazon Fulfillment Center. The bin images in this dataset are captured as robot units carry pods as part of normal Amazon Fulfillment Center operations. This dataset can be used for research in a variety of areas like computer vision, counting genetic items and learning from weakly-tagged data.

Problem Statement

The Amazon Bin Image Dataset contains images and metadata from bins of a pod in an operating Amazon Fulfillment Center. The bin images in this dataset are captured as robot units carry pods as part of normal Amazon Fulfillment Center operations

when we want the bins double-checked. The task is sometimes very challenging because of heavy occlusions and a large number of object categories.

We would like to open a new challenge in order to attract talented researchers in both academia and industry for these tasks. As a starting point, we provide baseline methods and pre-trained models for two tasks, counting and object verification tasks.

This is a simple task where you are supposed to count every object instance in the bin. We count individual instances separately, which means if there are two same objects in the bin, you count them as two.

As we could see, the problem is an Image Classification problem as the image is provided to build the ML/DL model to identify the number of objects in each bin.

Project files

Project consist of multiple files:

- sagemaker.ipynb -- main project file. Entrypoint
- train_model.py -- python script for tuning the network. Can be used from Sagemaker or as standalone application
- inference.py -- python script for running model inference
- file_list.json -- queried for the database to download only part of the dataset

Evaluation Metrics

I would use the evaluation at the end of each epoch and see the progress of the model using both Accuracy and RMSE for Evaluation Metrics. I used it at the end of each epoch to show the improvement of the model .

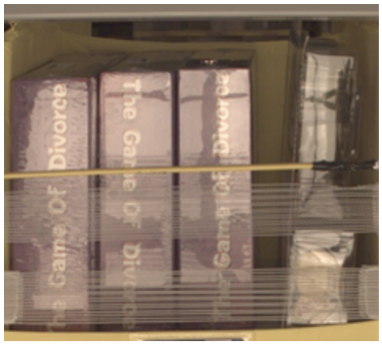
Datasets and Inputs

These are some typical images in the dataset. A bin contains multiple object categories and various number of instances. The corresponding metadata exists for each bin image and it includes the object category identification(Amazon Standard Identification Number, ASIN) which contains more than 500,000 image and metadata, quantity, size of objects, weights, and so on. The size of bins are various depending on the size of objects in it. The tapes in front of the bins are for preventing the items from falling out of the bins and sometimes it might make the objects unclear. Objects are sometimes heavily occluded by other objects or limited viewpoints of the images.

https://www.kaggle.com/datasets/dhruvildave/amazon-bin-image-dataset

The dataset contains 5 classes which identify each object in the picture

.Pictures :









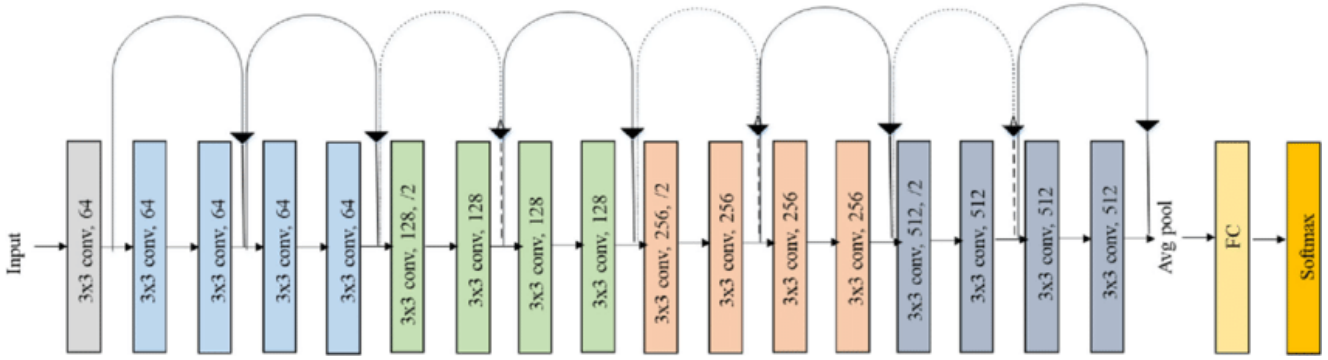
MetaData:


```
{
    "BIN_FCSKU_DATA": {
        "B00CFQWRPS": {
            "asin": "B00CFQWRPS",
            "height": {
                "unit": "IN",
                "value": 2.399999997552
            },
            "length": {
                "unit": "IN",
                "value": 8.199999991636
            },
            "name": "Fleet Saline Enema, 7.8 Ounce (Pack of 3)",
            "normalizedName": "(Pack of 3) Fleet Saline Enema, 7.8 Ounce",
            "quantity": 1,
            "weight": {
                "unit": "pounds",
                "value": 1.8999999999999997
            },
            "width": {
                "unit": "IN",
                "value": 7.199999992656
            }
        },
        "ZZXI0WUSIB": {
            "asin": "B00T0BUKW8",
            "height": {
                "unit": "IN",
                "value": 3.99999999592
            },
            "length": {
                "unit": "IN",
                "value": 7.899999991942001
            },
```

```
            "name": "Kirkland Signature Premium Chunk Chicken Breast Packed in Water,
12.5 Ounce, 6 Count",
            "normalizedName": "Kirkland Signature Premium Chunk Chicken Breast Packed
in Water, 12.5 Ounce, 6 Count",
            "quantity": 1,
            "weight": {
                "unit": "pounds",
                "value": 5.7
            },
            "width": {
                "unit": "IN",
                "value": 6.49999999337
            }
        },
        "ZZXVVS669V": {
            "asin": "B00C3WXJHY",
            "height": {
                "unit": "IN",
                "value": 4.330708657
            },
            "length": {
                "unit": "IN",
                "value": 11.1417322721
            },
            "name": "Play-Doh Sweet Shoppe Ice Cream Sundae Cart Playset",
            "normalizedName": "Play-Doh Sweet Shoppe Ice Cream Sundae Cart Playset",
            "quantity": 1,
            "weight": {
                "unit": "pounds",
                "value": 1.4109440759087915
            },
            "width": {
                "unit": "IN",
                "value": 9.448818888
            }
        }
    },
    "EXPECTED_QUANTITY": 3
}
```


Algorithms

Solution would be to build a Deep Learning model which would help to count the objects in each picture by using pre pre-trained model like Resnet as we used before in the previous project. In my case I used resnet18 .



ResNet model is widely used for image classification which is pretrained and can be customized in order to categorize images from different use cases. To adapt this pretrained model to our use case, different training jobs will be launched in AWS SageMaker. In addition, hyperparameters tuning jobs has been launched in order to find the most appropriate combination of hyperparameters for our use case.

As mentioned in the hyperparameter tuning section below we would fine-tune some parameters like learning rate and batch size as well as the number of epochs .

Benchmark model

There are many who worked on the same dataset and showed great results. I would compare results with them.

https://github.com/silverbottlep/abid_challenge
http://cs229.stanford.edu/proj2018/report/65.pdf

As we could see the results on the repository acc = 55.67 , RMSE = 0.93 :

Data preprocessing

The first step to train a model is to download and process the data which will be used as the input. As stated before, we have decided to focus on pictures with 1 to 5 objects.

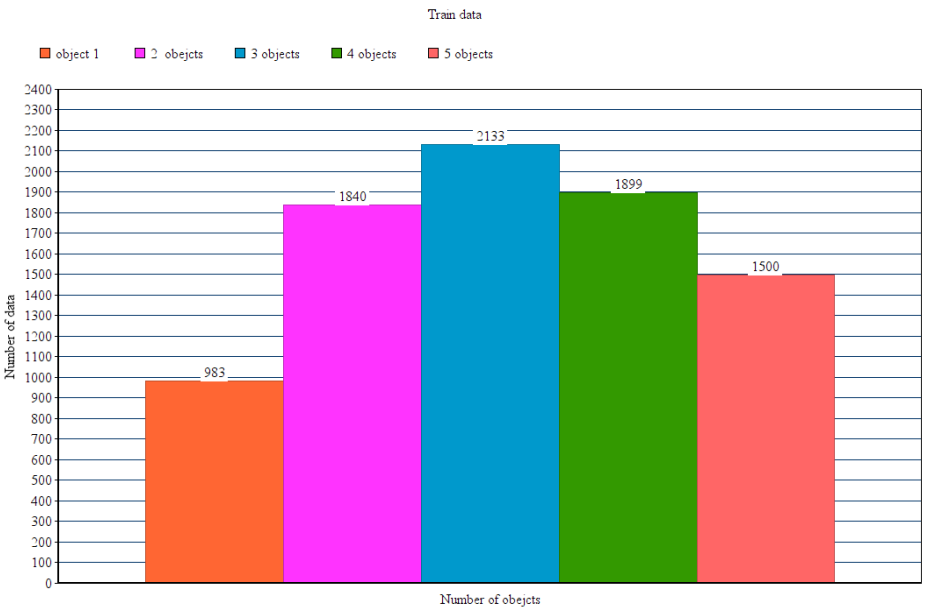
Each picture will be assigned a class according to the following rule:

Class 1 for pictures with 1 object
Class 2 for pictures with 2 objects
Class 3 for pictures with 3 objects

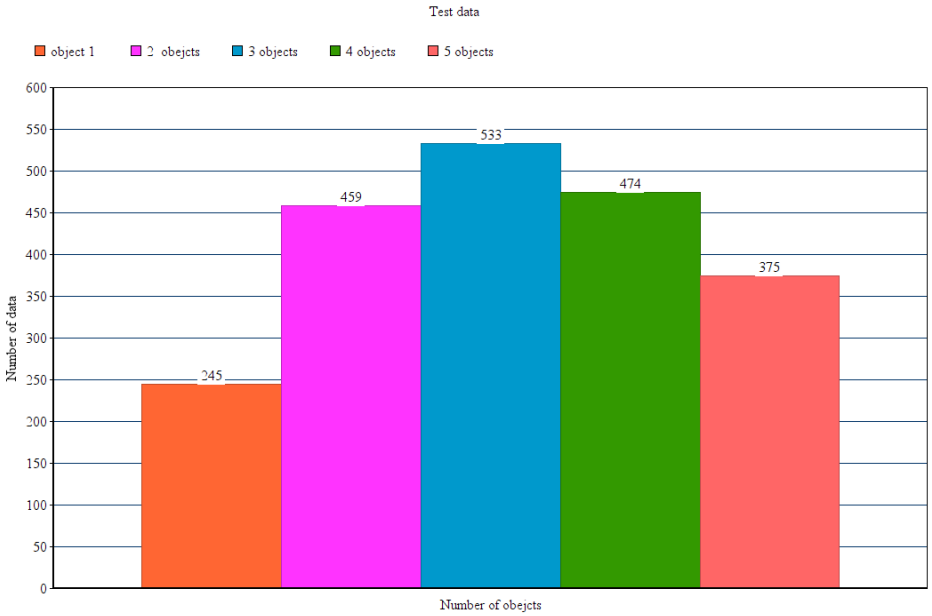
Class 4 for pictures with 4 objects
Class 5 for pictures with 5 objects

For each class, I decided to download all images: 0.8 will be used as input for the training phase, while 0.2 will be used as input for the validation phase as shown on the picture of the distribution for all the train and test dataset

- Number of images are 1228 images with 1 items in it (983 train - 245 test) .
- Number of images are 2299 images with 2 items in it.(1840 train - 459 test)
- Number of images are 2666 images with 3 items in it (2133 train - 533 test).
- Number of images are 2373 images with 4 items in it (1899 train - 474 test).
- Number of images are 1875 images with 5 items in it(1500 train - 375 test) .



.

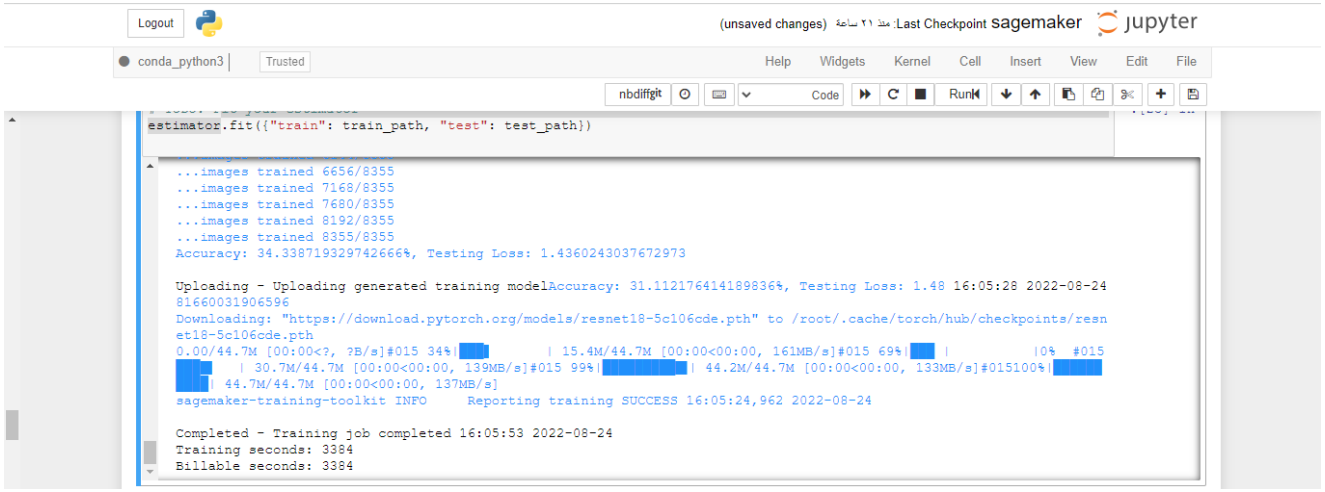


Finally, all these pictures were uploaded to S3, as it is the entry point for data for models being trained on AWS in a new path called new_data.

Refinement

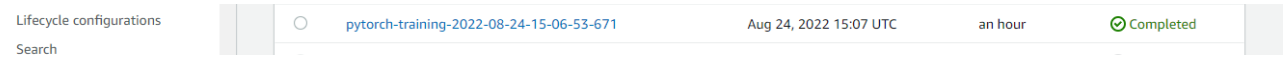
As stated before, I planned to use a ResNet neural network to train the model As a base I used this Python training script, which is the one I implemented for the "Image Classification" project of this course (file train_model .py). I adapted the number of classes (from 133 to 5) and configured the transformation part in order to deal with the new set of images (i.e. resizing).

Then I launched this script through the Jupyter Notebook. However, the results, as can be seen on this screenshot, not very promising, with a RMSE of 1.48 and an accuracy of 31,11%.



This script used a pretrained ResNet18 neural network for training. In order to obtain a more accurate model, I tried the same script but with different ResNet networks. I tested with ResNet50 and the results were pretty similar. so I decided to move my project to this implementation (ResNet18) but with a pretrained network. With this implementation, accuracy was 31.11% and RMSE to 1.48, which was not perfect but better than the results obtained on the other attempt.

However, reading the project done by Pablo Rodriguez et al., I discovered they were using a ResNet34 non-pre- trained network which I think would get better results than I got.

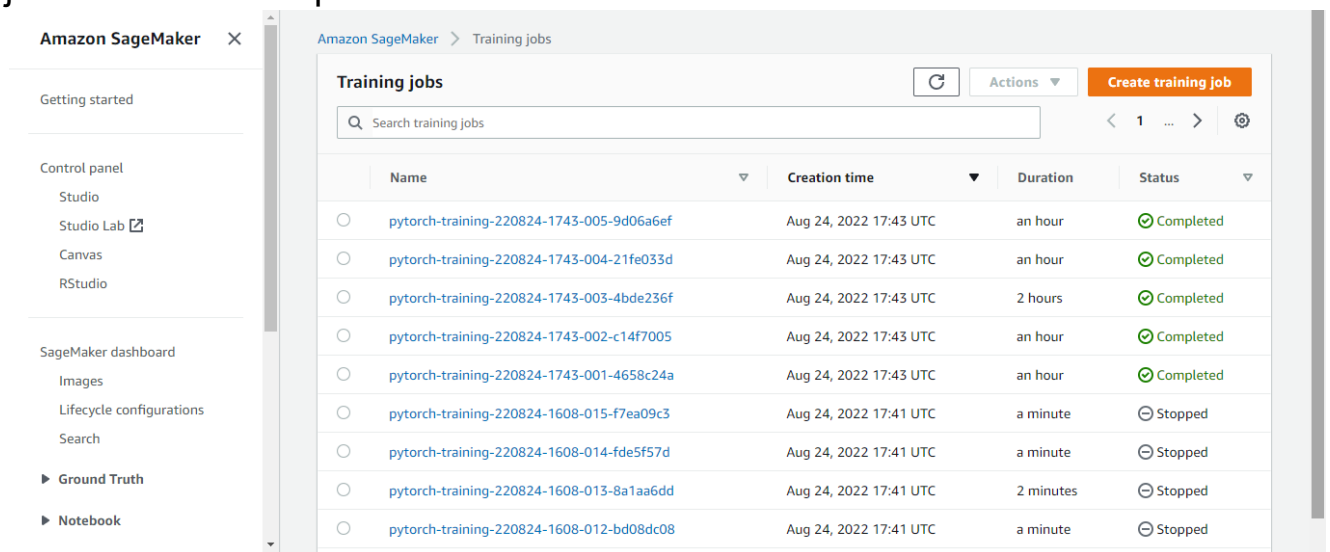


Hyperparameter Tuning

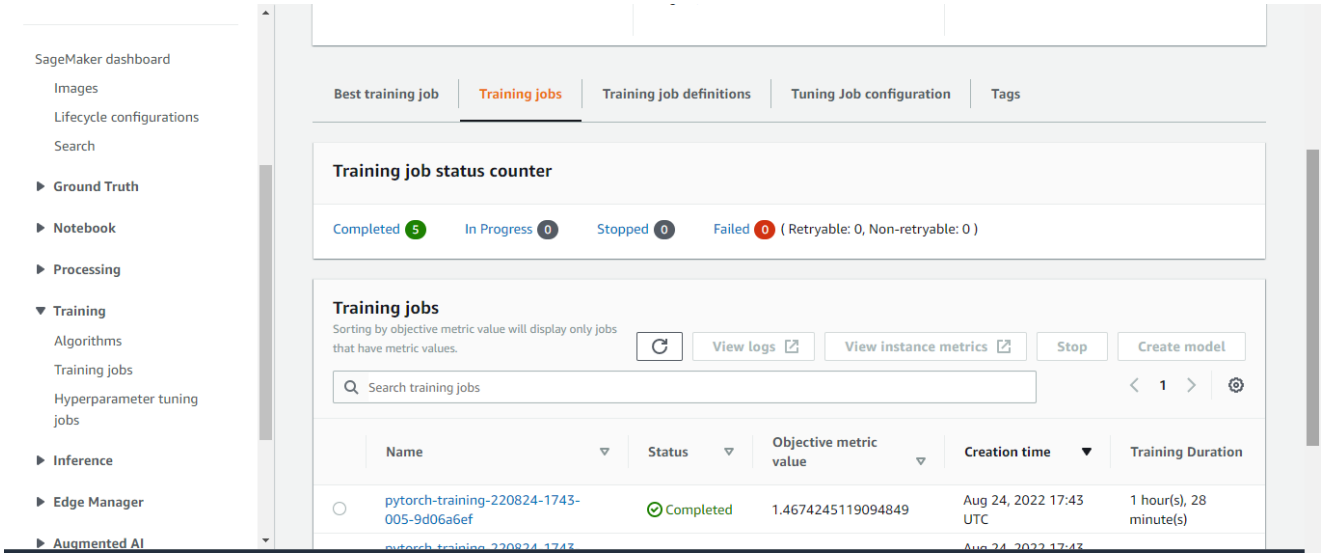
At this state, I decided it was high time to tune the hyperparameters in order to find a better combination of them which allow me to obtain a more precise model. Specifically, these hyperparameters were tuned:

- The number of epochs (epoch) between (4,10)
- The batch size (the number of images being trained on each iteration) (32, 64, 256)
- The learning rate (lr) (0.001,0.1)

I made max parallel jobs equal to 5 with max jobs 5 so it should run training jobs with max 5 parallel at each time .



you can see, as shown in the screenshot, the hyperparameter tuning.



After completing, the best hyperparameters combination was the following one:

- Epochs: 8
- Batch Size: 256
- Learning Rate: 0.00216189132009066

With this combination, testing accuracy spiked to 33% and RMSE was 1.46, as can be seen on the following screenshot.

Model Evaluation and Validation

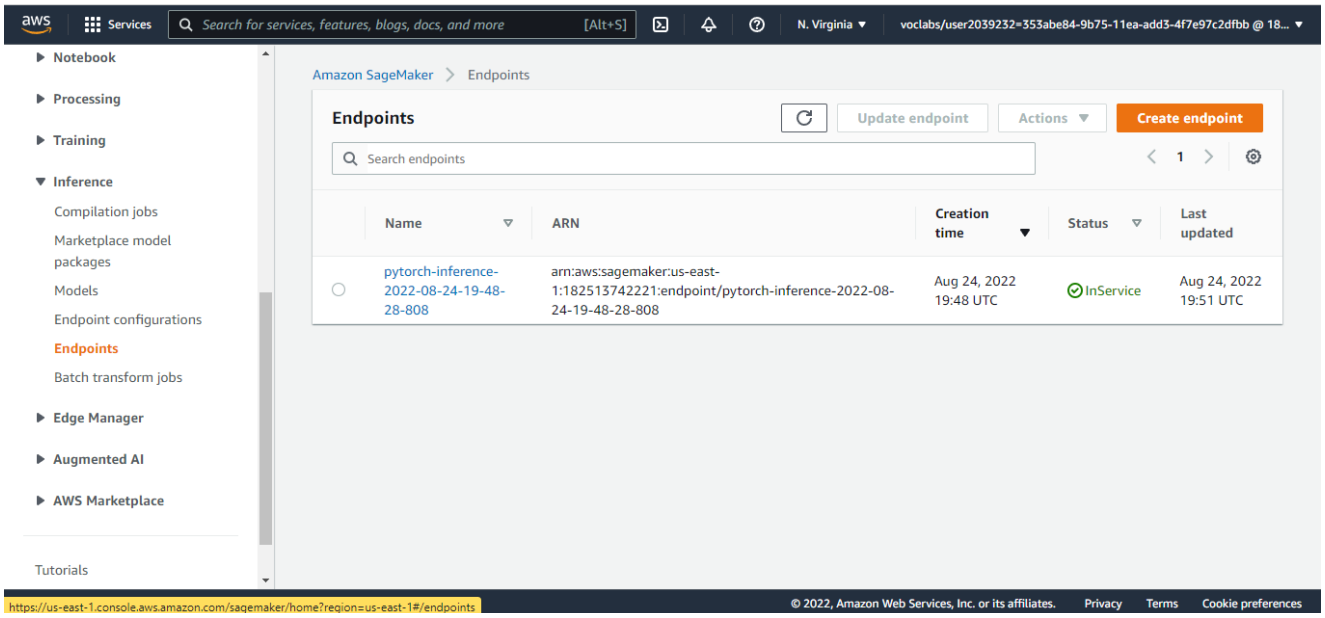
As can be seen in the previous picture, results were very poor if compared with the ones obtained on other projects such as the one developed by Pablo Rodriguez et al. where a RMSE of 0.94 is obtained on the testing phase. At this point but mine was worse as it was 1.46 with lower accuracy than developer by Pablo Rodriguez I think the difference between my work and his that he modified the train_mode.py in a professional way with using Resnet34 make the result higher and better .

I tried to modify his file and see the result it was better but not very close to it with 36% accuracy because I used the resnet 18 still .

Unfortunately, I deleted the logs because I finished the free tier.

Model deployment

After training model can be deployed and used from different AWS services. Deployment procedure is presented in notebook sagemaker.ipynb creating an endpoint to predict throw it then delete it to avoid cost using interfec.py file for entry point.



Example code for model inference:

```
from PIL import Image
import io

with open("test_image.jpg", "rb") as image:
    f = image.read()
    img_bytes = bytearray(f)
    Image.open(io.BytesIO(img_bytes))

response=predictor.predict(img_bytes,
initial_args={"ContentType": "image/jpeg"})
```