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 .

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 :

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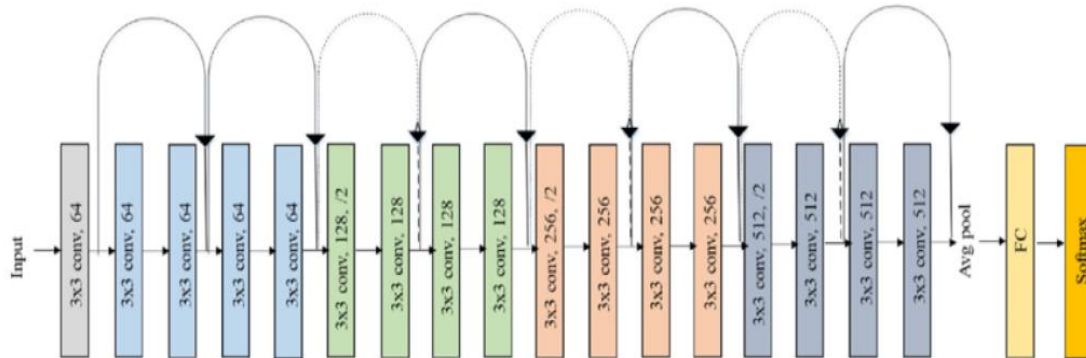
ResNet model is widely used for image classification which is pretrained and can be customized in order to categorize images from different use cases. To adapt this pretrained model to our use case, different training jobs will be launched in AWS SageMaker. In addition, hyperparameters tuning jobs has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

http://cs229.stanford.edu/proj2018/report/65.pdf

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

Implementation

The implementation process can be split into two main stages:

1. The classifier training and Testing stage
2. The Deployment stage

During the first stage, the classifier was trained on the preprocessed training data. This was done in a

Jupyter notebook (titled "sagemaker.ipynp"), and can be further divided into the following steps:

1. Load both the training and testing images into memory, preprocessing them as described in the previous section, then send them to s3 Bucket to pe implemented in Training Phase.
2. implement the aws pytorch image using entrypoint "train_model.py" that retrieve resnet algorithm with hyperparameters = {

   "batch-size": "512",

   "epochs": 5,

   "lr": "0.01",

}

On the training and testing Datasets .

3. Evaluate The Model Using the accuracy and RMSE
4. implement Hyperparameter Training Job to find the best estimator and Hyperparameters
5. deploy the the best estimator

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) .**

Text data

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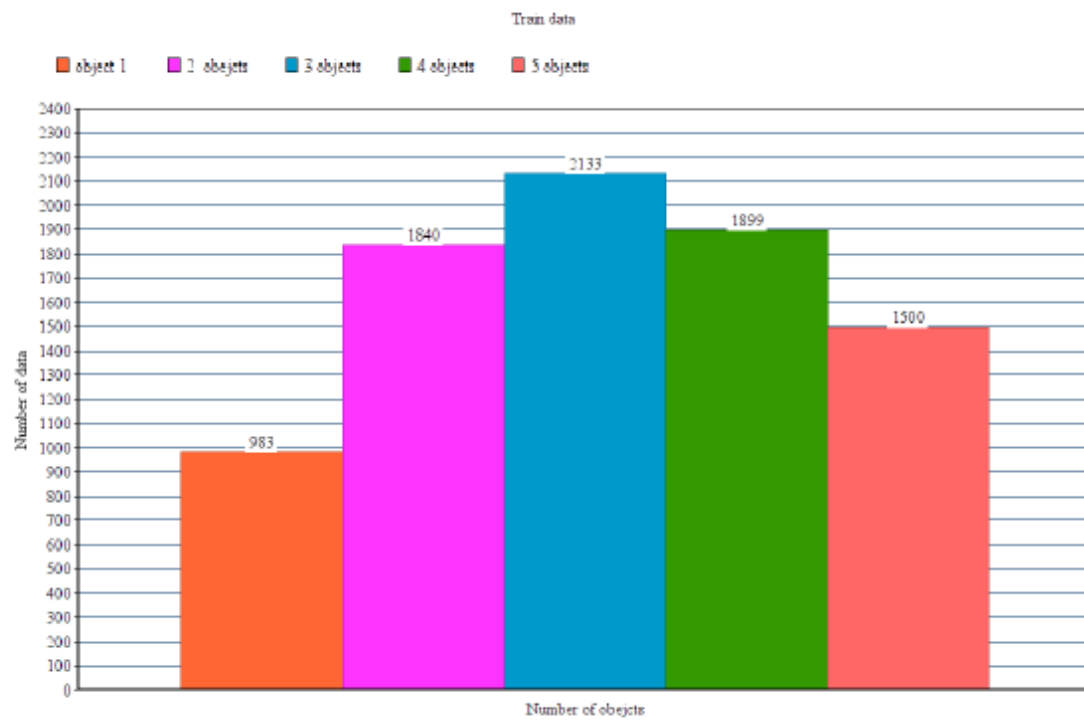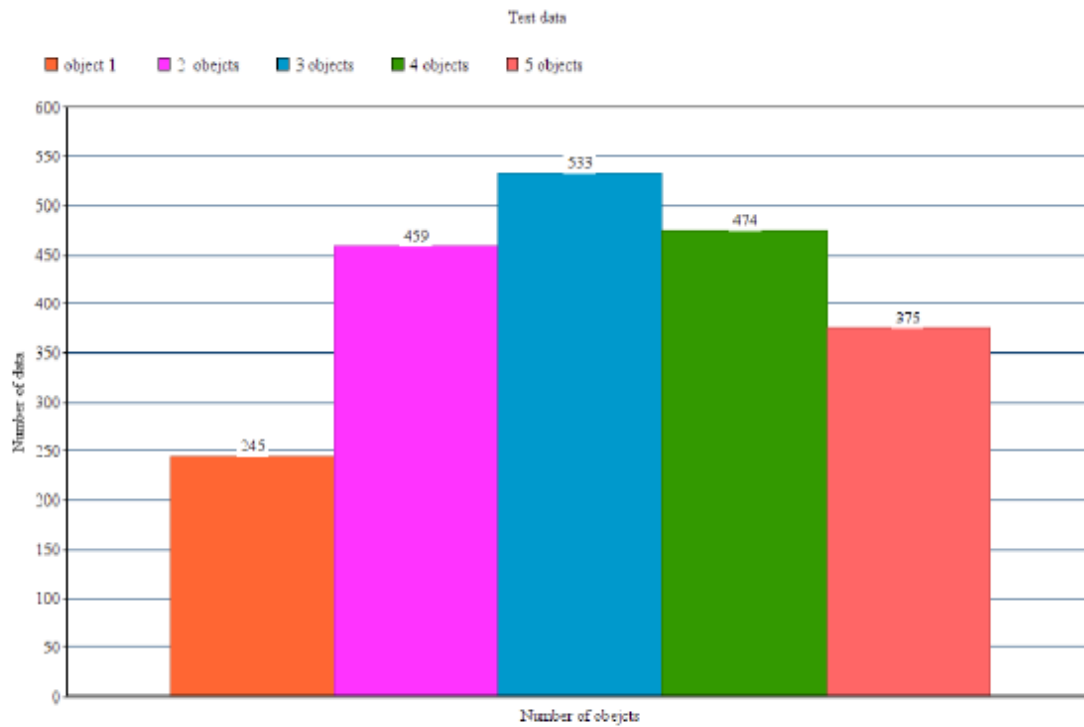
Finally, all these pictures were uploaded to S3, as it is the entry point for data for models being trained on AWS in a new path called new_data.

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 30.20%.

```
8355/8355 images trained...
Accuracy: 32.93836026331538%, Testing Loss: 1.4645368534957912
Accuracy: 30.201342281879196%, Testing Loss: 1.4863957531271594
Downloading: "https://download.pytorch.org/models/resnet18-5c106cde.pth" to /root/.cache/torch/hub/checkpoints/resnet18-5c106cde.p
th
#015  0%|          | 0.00/44.7M [00:00<?, ?B/s]#015 87%|████████  | 38.8M/44.7M [00:00<00:00, 407MB/s]#015100%|████████| 44.7M/
44.7M [00:00<00:00, 406MB/s]
2023-03-17 23:19:01,189 sagemaker-training-toolkit INFO     Reporting training SUCCESS

2023-03-17 23:19:11 Uploading - Uploading generated training model
2023-03-17 23:19:31 Completed - Training job completed
```

First Training job

	pytorch-training-2023-03-17-22-37-08-359	3/18/2023, 12:38:37 AM	41 minutes	⊘ Completed	-	-

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 6 with max jobs 2 so it should run training
jobs with max 2 parallel at each time .

pytorch-training-230317-2322

Stop tuning job

Hyperparameter tuning job summary

Name	**Status**	**Approx. total training duration**
pytorch-training-230317-2322	⊘ Completed	4 hour(s), 35 minute(s)
ARN	**Creation time**	
arn:aws:sagemaker:us-east-1:277157499543:hyper-parameter-tuning-job/pytorch-training-230317-2322	Mar 17, 2023 23:22 UTC	
	Last modified time	
	Mar 18, 2023 01:54 UTC	

Best training job | **Training jobs** | Training job definitions | Tuning Job configuration | Tags

Training job status counter

Completed **6** In Progress **0** Stopped **0** Failed **0** (Retryable: 0, Non-retryable: 0)

Activate Windows

Training job status counter

Completed **6** In Progress **0** Stopped **0** Failed **0** (Retryable: 0, Non-retryable: 0)

Training jobs

Sorting by objective metric value will display only jobs that have metric values.

View logs ⧉ | View instance metrics ⧉ | Stop | Create model

Search training jobs

‹ 1 › ⚙

|	Name	Status	Final objective metric value	Creation time	Training Duration
○	pytorch-training-230317-2322-006-2a1d2546	⊘ Completed	13.260614395141602	3/18/2023, 3:19:15 AM	32 minute(s)
○	pytorch-training-230317-2322-005-b2e5b688	⊘ Completed	3.4645771980285645	3/18/2023, 3:01:48 AM	40 minute(s)
○	pytorch-training-230317-2322-004-f9898712	⊘ Completed	1.503721833229065	3/18/2023, 2:10:33 AM	1 hour(s), 8 minute(s)
○	pytorch-training-230317-2322-003-ba4f5010	⊘ Completed	1.5338243246078491	3/18/2023, 1:58:16 AM	56 minute(s)
○	pytorch-training-230317-2322-002-39c52b8b	⊘ Completed	1.4996742010116577	3/18/2023, 1:23:03 AM	45 minute(s)
○	pytorch-training-230317-2322-001-c800af0b	⊘ Completed	1.5330885648727417	3/18/2023, 1:23:02 AM	34 minute(s)

Activate Windows

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

● **Epochs: 5**

● **Batch Size: 32**

● **Learning Rate: 0.1**

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.
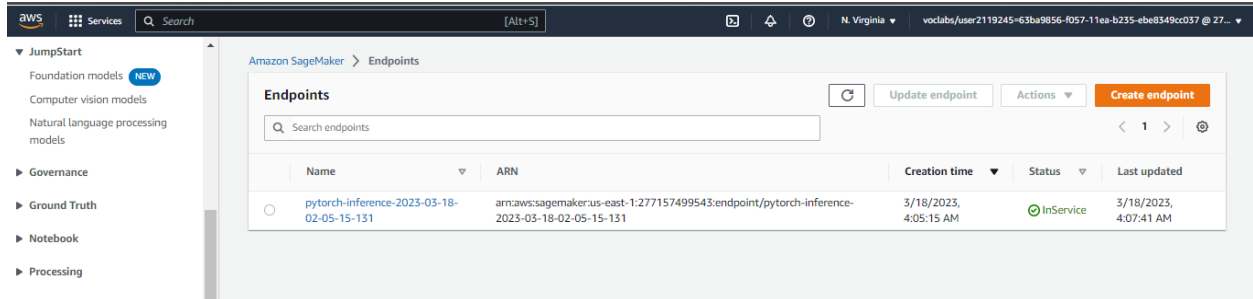
Results

Model Evaluation and Validation

During development, a test set was used to evaluate the model.
The final architecture and hyperparameters were chosen because they performed the best among the tried combinations in The Hyperparameter Training Job, with the best Hyperparameters

● Epochs: 5

● Batch Size: 32

● Learning Rate: 0.1

Then deploy the best estimator that has the best accuracy and test the endpoint in some test images .

Accuracy

```
In [73]: correct = 0

for i in range(1, 6):
    partial = len(list(filter(lambda x: x == i, model_responses[i])))
    correct += partial
    print(f"Accuracy for {i} objects: {partial/200}")

print(f"Mean Accuracy: {correct/1000}")
```

```
Accuracy for 1 objects: 0.0
Accuracy for 2 objects: 0.7
Accuracy for 3 objects: 0.32
Accuracy for 4 objects: 0.59
Accuracy for 5 objects: 0.0
Mean Accuracy: 0.322
```

Rmse

```
In [75]: import math
rsme = {}

for i in range(1, 6):
    rsme[i] = sum(list(map(lambda x: (x - i) ** 2, model_responses[i])))
    print(f"RSME for {i} objects: {math.sqrt(rsme[i]/200)}")

print(f"Total RSME: {math.sqrt(sum(rsme.values())/1000)}")
```

```
RSME for 1 objects: 0.0
RSME for 2 objects: 1.5362291495737217
RSME for 3 objects: 2.2494443758403984
RSME for 4 objects: 3.0298514815086235
RSME for 5 objects: 3.5735136770411273
Total RSME: 2.4236336356801123
```