INVENTORY MONITORING AT DISTRIBUTION CENTRES

Project Overview

Distribution centres all over the world use robots to move objects from one place to the other in bins which contain multiple objects. Determining the number of objects in each bin can be valuable in order to check that the process is working as expected.
The goal of this project is to build a machine learning model that can count, with good enough accuracy, the number of items in each bin in order to track inventory and check that the bins have the appropriate number of items in order to reduce stock mismatches.

Problem Statement

Based on the background, it can be seen that the problem to be resolved here is related to image classification. A ton of images have been provided by our client (Amazon) as our dataset. A machine learning model will be built to classify bin images by the number of items in them.

Metrics

For this project, we used model accuracy (images that are correctly classified versus the total number of images to classify). In further work on this project, better and more appropriate

model metrics such as precision and RMSE can be used. These metrics are used to evaluate the model's performance.
Metrics are calculated at the end of every training epoch to observe the model's improvement or otherwise.

Analysis

Data Exploration

The dataset used for the project will be the Amazon Bin Image Dataset which contains more than 500,000 images and metadata from bins of a pod in an operating Amazon Fulfillment Center. Both images and metadata will be used to develop the model. Images will be used as the main input in the training phase. Metadata will be used in order to arrange the pictures in a way that the ML model will identify the number of possible classes.
Amazon provides us with these images via S3. Specifically, both, images and metadata, are stored on the aft-vbi-pds bucket. Images are stored in the bin-images folder. Each image is a different JPG file whose name is similar to <X>.jpg being <X> a number.
Metadata is stored in the metadata folder. Each image has a corresponding metadata file whose format is JSON. The most important field of these JSON files is EXPECTED_QUANTITY as it contains the number of objects which can be found on the specific image. For the scope of this project, we will focus on pictures containing 0 to 4 objects

Below is an example of a bin with three items in it:

And another with just one item in it:

Algorithms and Techniques

In this project, an ML model has been built to identify the number of objects in each image. In order to build this ML model, SageMaker has been used, training a model using a resnet34 neural network. ResNet model is widely used for image classification which is pre-trained and can be customized in order to categorize images from different use cases. To adapt this pre-trained model to our use case, different training jobs will be launched in AWS SageMaker. In addition, hyperparameters tunning jobs have been launched in order to find the most appropriate combination of hyperparameters for our use case.

Benchmark

Others have worked on the same data to build their own models. Specifically, we can find two GitHub projects related to this problem:

1. Amazon Bin Image Dataset (ABID) Challenge by silverbottlep
2. Amazon Inventory Reconciliation using AI by Pablo Rodriguez Bertorello, Sravan Sripada, Nutchapol Dendumrongsup

As can be seen in the conclusions of both projects, the obtained accuracy is 55% approximately with an RMSE of 0.94.

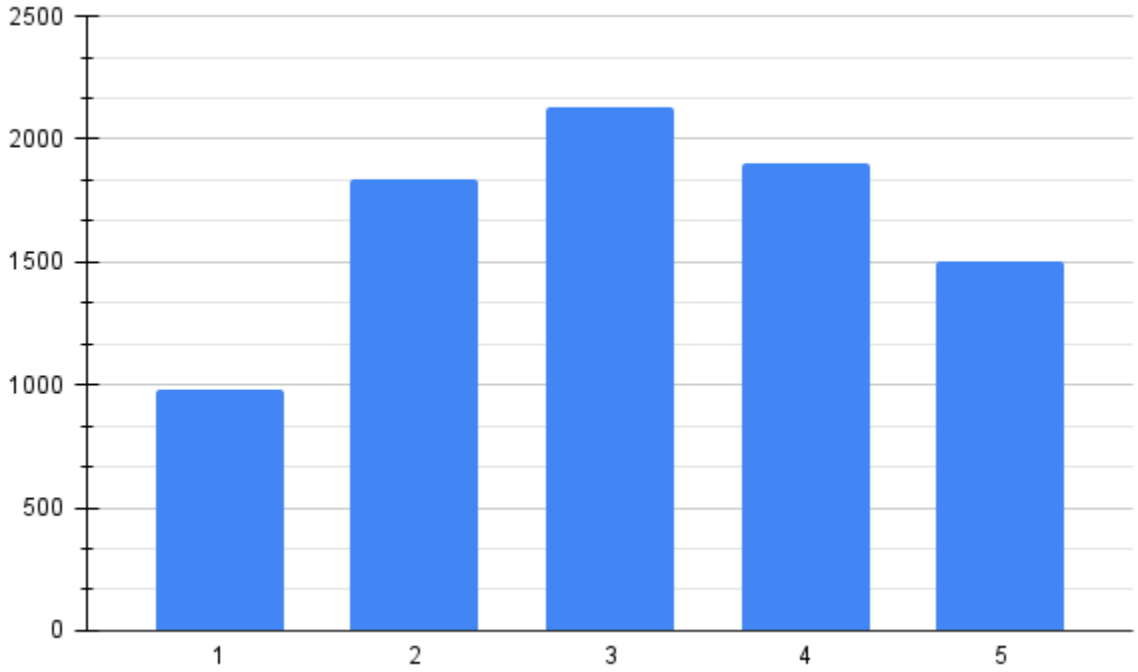
Methodology

Data preprocessing

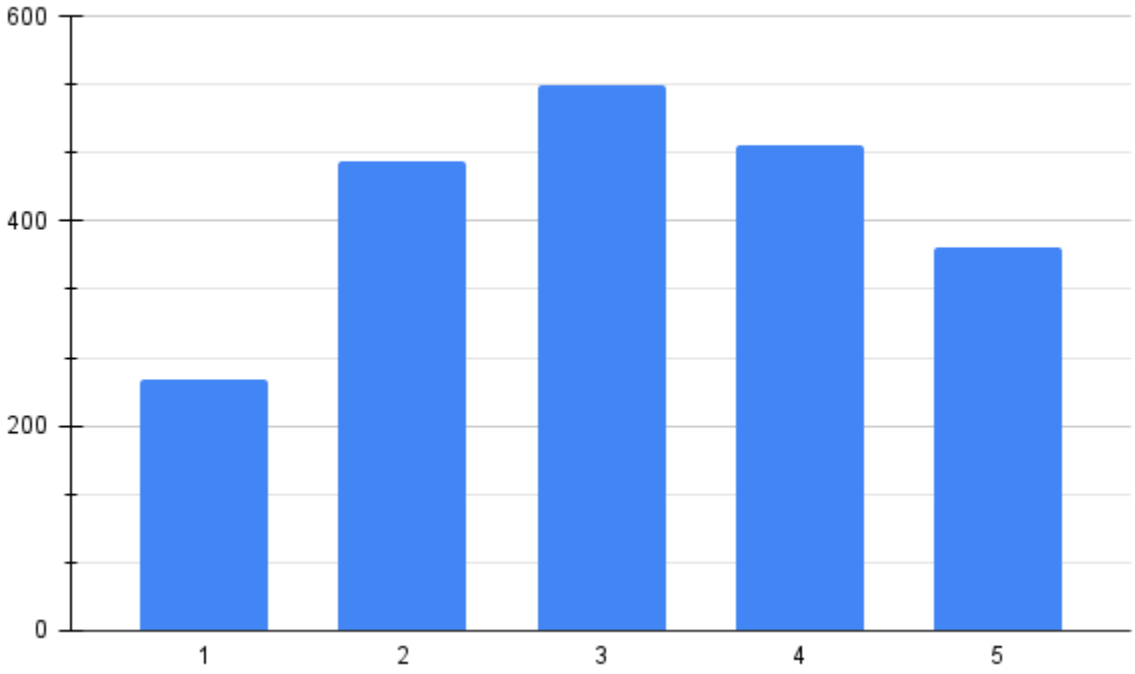
The first step to training a model is to download and process the data which will be used as the input. As stated before, we have decided to focus on pictures with 0 to 5 objects. Each picture will be assigned a class according to the following rule:

- Class 1 for pictures without objects
- Class 2 for pictures with 1 object
- Class 3 for pictures with 2 objects
- Class 4 for pictures with 3 objects
- Class 5 for pictures with 4 objects

Below is the distribution of the classes of the training data:



Similarly, for the testing data:



In order to download these pictures, a Python script has been created. The script can be found in the project Jupyter Notebook (file sagemaker.ipynb). Specifically, the script will iterate over the JSON files from the Amazon dataset and will download the picture if it contains 0 to 5 objects.
Finally, all these pictures were uploaded to S3, as it is the entry point for data for models being trained on AWS.

Implementation

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.py). I adapted the number of classes (from 133 to 6) 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 first results were not very promising, with

an accuracy of 31,60%. This first script used a pre-trained ResNet18 neuronal network for training.
In order to obtain a more accurate model in future works, it would be a good idea to train with the same script but with different ResNet networks.
However, reading the project done by Pablo Rodriguez et al., I discovered they were using a ResNet34 network which is not pre-trained, so it would be a good idea to implement my project (which used ResNet18) but with a pre-trained network. With this implementation, accuracy could be about 38% and RMSE could also drop considerably, which is not perfect but better than the results obtained on the first attempt.

Refinement

The performance of the model can be refined by tuning the hyperparameters in order to find a better combination.
I suggest the following hyperparameters:

- The number of epochs epochs
- The batch size (the number of images being trained on each iteration) batch-size
- The learning rate lr

In addition, other hyperparameters such as early stopping can also be tuned.

Results

Model Evaluation and Validation

Results were very poor if compared with the ones obtained on other projects such as the one developed by Pablo Rodriguez et al. where an RMSE of 0.94 is obtained in the testing phase. At this point, I decided to launch their training script against my dataset.

As can be seen, results with the same input were pretty close to what I obtained, with an RMSE of 1.27 and an accuracy of 43,8%. It's important to note that while their script requires 40 epochs to obtain these results, my script only requires 14.

Justification

As stated before, both training scripts, the one I developed and the one developed by the team Pablo Rodriguez, generated pretty close results differing by about 3 points (40,9% to 43,80%). My work does not demonstrate a substantial increase in performance but shows that more can be done to improve the performance of these models. For example, through hyperparameter tuning, the model can be made to perform better.