Udacity - SDCND

Project 1: Object Detection

by Jose Jesus Cabrera Pantoja

Project overview

In this project, we use data from the Waymo Open dataset in order to create a convolutional neural network to detect and classify objects. This is done by following steps such as data acquisition and processing, Exploratory data analysis(EDA) , cross validation, Data augmentation and creation of animations.

1 Setup

1.1 Requirements

- NVIDIA GPU with the latest driver installed
- docker / nvidia-docker

1.2 Build

Build the image with:

```
docker build -t project-dev -f Dockerfile.gpu .
```

Create a container with:

```
docker run -v <PATH TO LOCAL PROJECT FOLDER>:/app/project/ -ti project-dev bash
and any other flag you find useful to your system (eg, --shm-size).
```

Once in container, you will need to install gsutil, which you can easily do by running:

```
curl https://sdk.cloud.google.com | bash
```

Once gsutil is installed and added to your path, you can auth using:

```
gcloud auth login
```

download the Pretrained model and extract it in the training/pretrainedmodels/ directory of the project

1.3 Debug

Follow this tutorial if you run into any issue with the installation of the tf object detection api

1.4 Structure

- Dockerfile.gpu: The docker file used to build the projet's docker image.
- README.md: contains instructions for the project setup.
- requirements.txt: contains the dependencies to be installed with the docker image.

  project_folder/experiments

- exporter_main_v2.py: This file is used to create an inference model
- mode_main_tf2.py: This file is used to launch training
- project_folder/training/pretrained-models/ This folder contains the checkpoints of the pretrained models.
  project_folder/
- create_splits.py: Creates the test, val and train folders and partitions the datasets into them.
- download_process.py: Downloads data sets from Waymo Open Dataset and processes them into acceptavble format.
- edit_config.py: this file is used to generate configuration files used for model training.
- Exploratory Data Analysis.ipynb: Used to analyse the image and annotation data by displaying them.
- Explore augmentations.ipynb: Used to test various augmentation methods.
- filenames.txt: contains the file names of the data sets to be downloaded.
- inference_video.py: This file is used to generate animation videos of our model's inference for any tf record file.
- label_map.pbtxt: Contains the defined object classes
- pipeline_new.config: The training configuration file.

2 DataSet

After exploring the dataset the following analysis where made. In Figure 1 is shown a sample of teh data set with its corresponding bounding boxes.

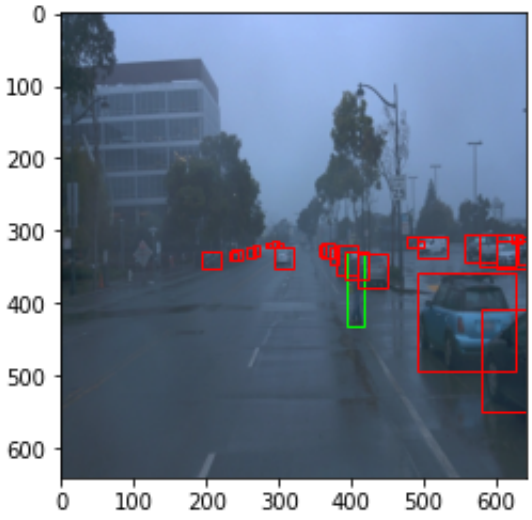

Figure 1: Sample

2.1 Dataset Analysis

It can be seen from the Figure 2 that the images were taken in different places sucha as in town and contains a vehicles, pedestrians and a cyclist. Furthermore, some objects are close enough to the camera while other are very far away and almost can't be seen with the human eye. In some cases the images are blurred and it is dificult to distinguish the object.

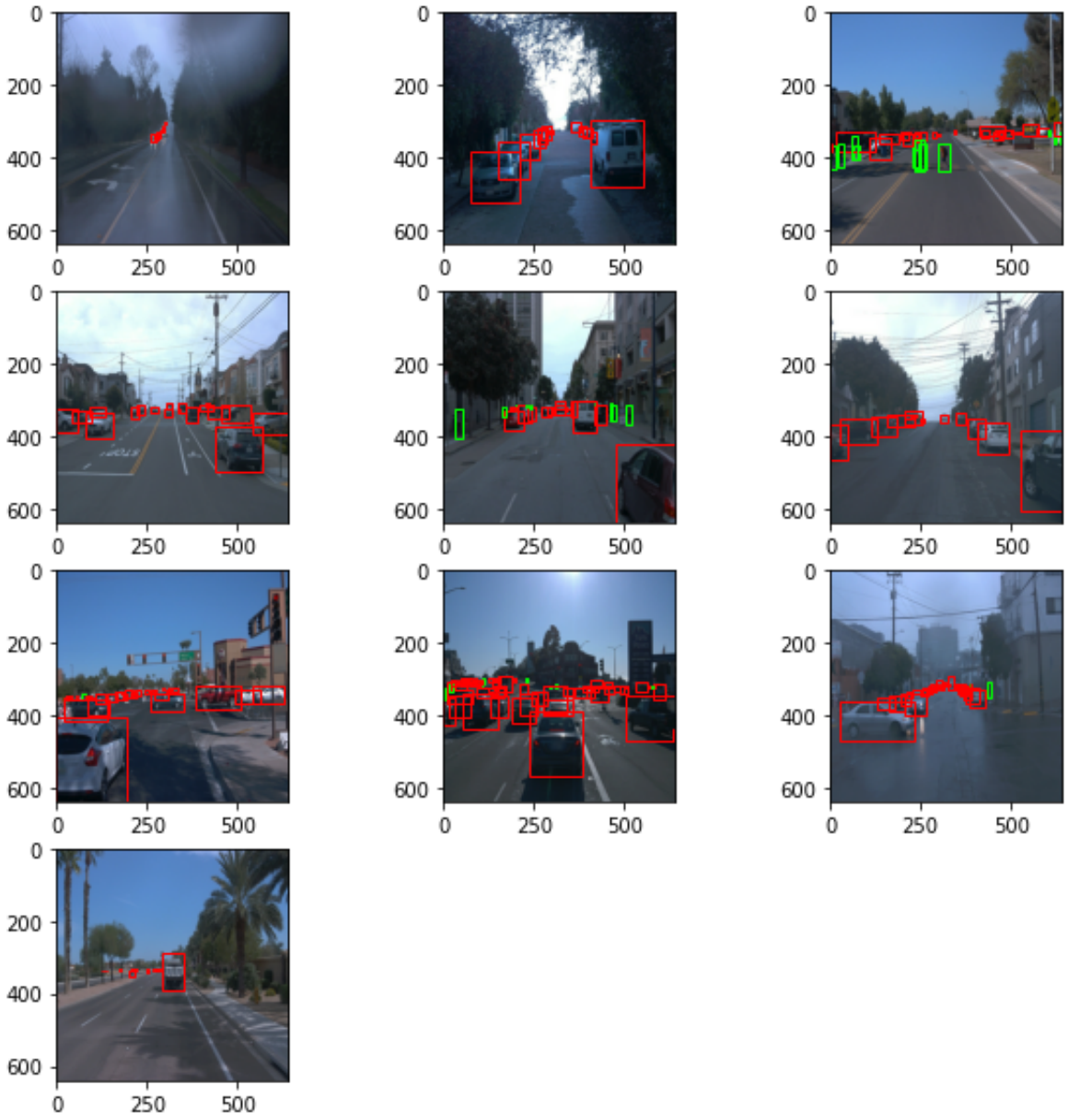

Figure 2: Samples

Some images do not contain all the class objects, some only have cars, others have cars and pedestrians only. Furthermore, in some images the environment is bright and the lighing is clear enough, but in some other images where taken at night and the environement is quit dark as well as due to climate changes some images

are blured by fog. The overall class Overall, there are more cars that pedestrians and cyclists. Moreover, some Images are taken during the day, some at night, some are blured due to fog or darkness. Please, refer to the Figure 2.

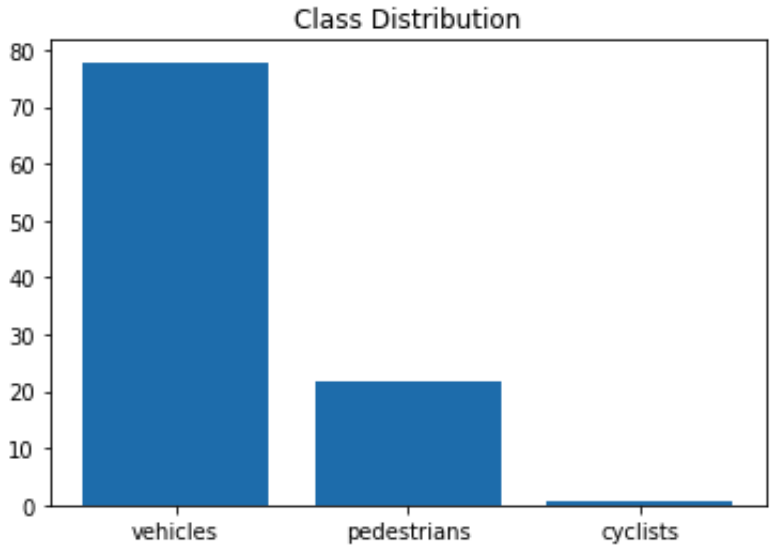

Figure 3: Distribution

In Figure 3, it is shown a bar chart for the object distribution. This show that the images contain a higher number of vehicles than pedestrians and cyclists so the class distribution is not balanced. We can see from these charts that the images contain a higer number of vehicles that pedestrians and cyclists, where cyclists are the least numerous objects.

2.2 Cross validation

The cross validation strategy is to randomly split the dataset into three partitions for trainign test and evaluation in the ration 8:1:1 or percentages 80%,10% and 10%. This because the training set need to be large enough in such a way that the model gets well trained because from the results of the analysis, we can see that the percentage of cyclist objects per images is very low so if the training data is no much the model might not be well trained to recognise cyclists.

It can also be seen that the climate and brightness can change from image to image so the training data should be large enough for the model to be able to recognize object classes in any kind of image.

3 Training

After monitoring the training through the tensorboard, the following was discoverd. In Figure 4 and 5 are shown the validation loss which is higher that training loss. This shows that there is some over fitting in the data set though it is not that much. This behaviour was expected and it should be enhanced with data augmentstion which will give a better recognition of unkonwn data to the model.

Figure 4: Tensorboard 1

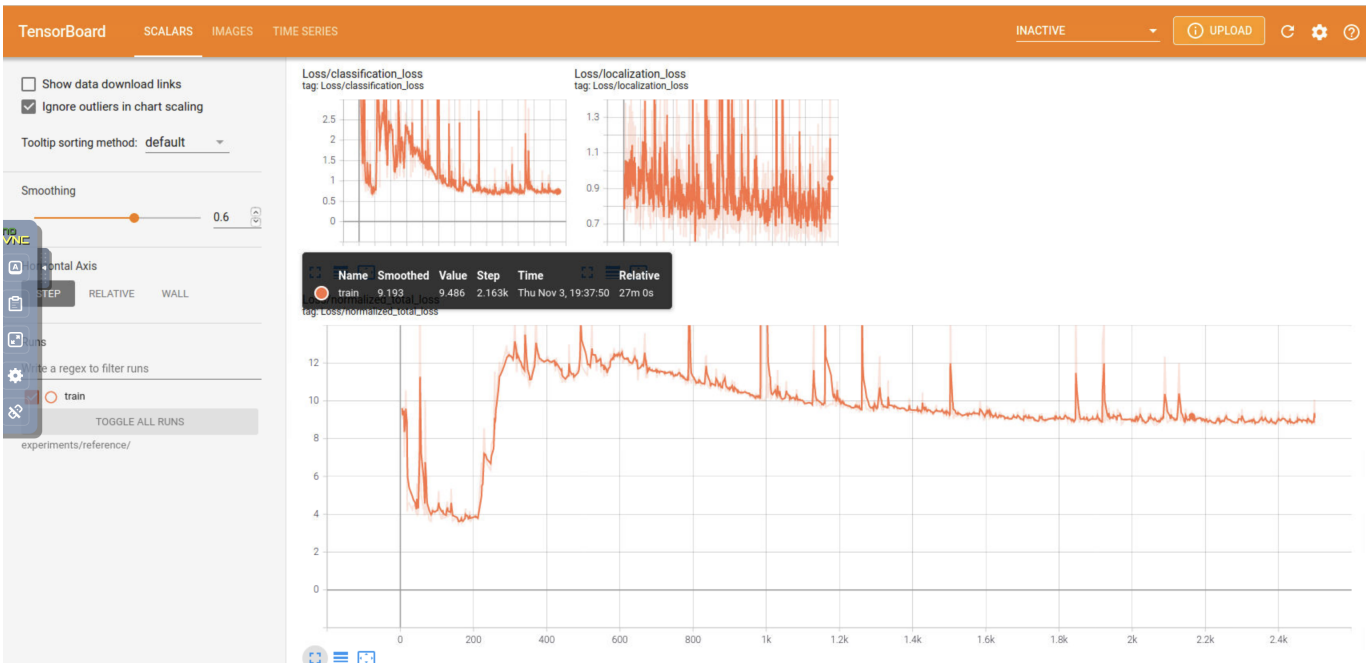

Figure 5: Tensorboard 2

3.1 Reference experiment

3.2 Improve on the reference

Note that the images were taken from different places having different climatic conditions therefore the images are very different. Then, it is needed to take into account this broad spectrum, therefore, the data augmentation method was applied which would yield the best results are those that provide some variation in environment. For instance, scalling up and down, changing the color, contrast, brigthness and intensity in order the permit the model to detect objects from environments with any type of climate and bightness. The Augmentation options chosen here are;

1. random_image_scale: Since there are cases where the vehicles can vbe close enough to the camera or too far from it, this augmentation was chosen in order to simulate that by randomly enlarging or minimizing images. This should enhance the training.
2. random_horizontal_flip: This flips the position of objects in images this will help to better train the model to recognize objects from any direction.
3. random_rgb_to_gray: This augmentation is used so the the model can better recognize objects even in the dark or in zones where the view is blured due to fog.
4. random_adjust_brightness: This randomly adjusts the brightness of the images and enhances the object recognition of the model in daylight or in the night in torch light.
5. random_adjust_contrast: This modifies the contrast of the images to make it more dark and trains the model to recognize images in dart climates.
6. random_distort_color: With image colors randomly distorted, the model will better recognize vehicles with any typre of paint, and it will also enhance image recognition when moving at high speed.
7. random_crop_image

After applying this method the following images were obtained:

Figure 6: Augmentation

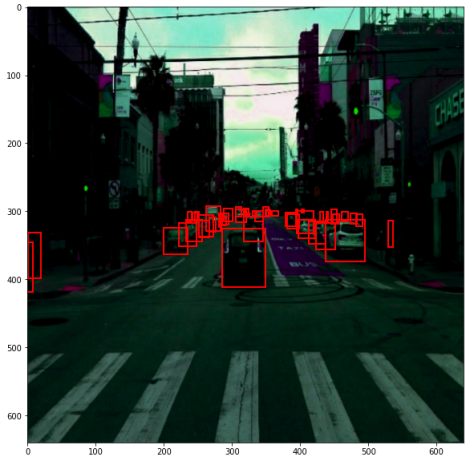

Figure 7: Augmentation

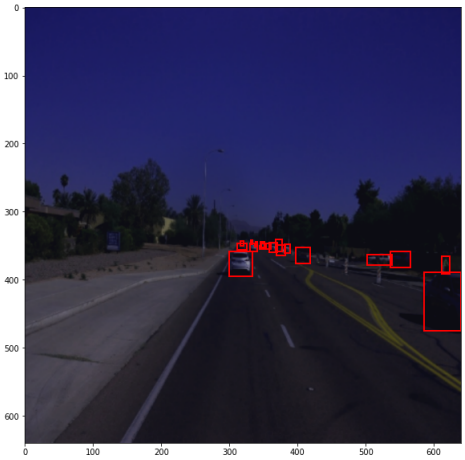

Figure 8: Augmentation

After that there was the option to change the optimizer. Throught the different optimizers which could be used here like Mini batch, Momentum and AdaGrad and RMSSProp,Adam optimizer was tested but the results were bad. Therefore, the momemtum optimizer was kept and the batch size was increased to 8.

After running the training with these augmentations applied, it can be seen from the new loss chart that the loss with augmentation is less than the loss that we had without augmentation. This then results to a less significant error from the model during optimization.

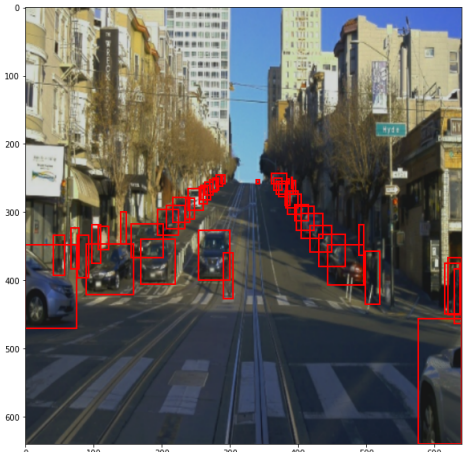

Figure 9: Augmentation

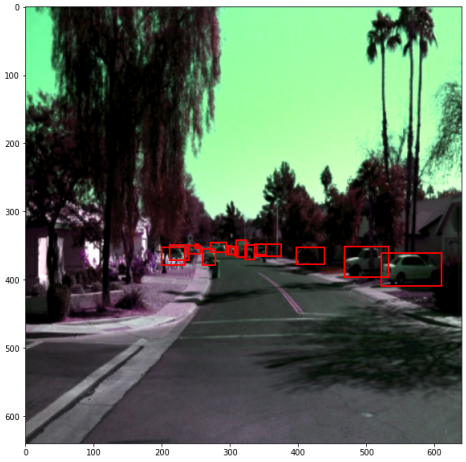

Figure 10: Augmentation

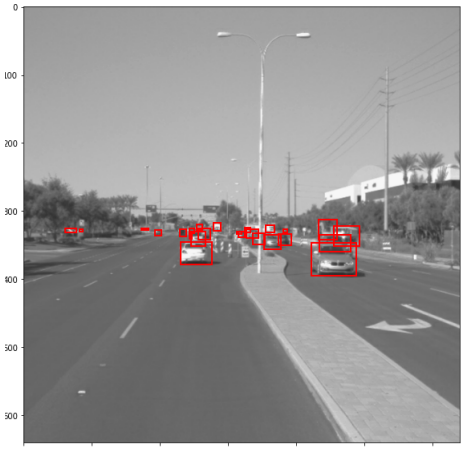

Figure 11: Augmentation

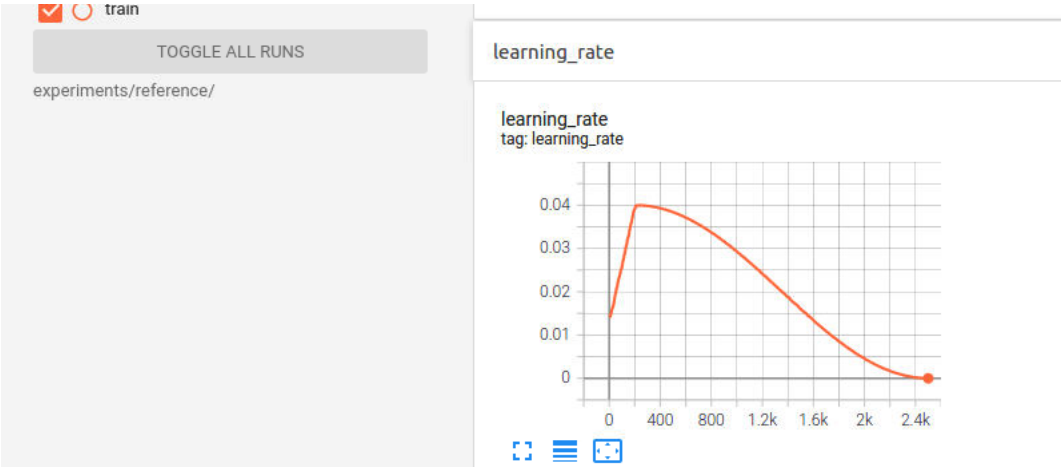

Figure 12: TensorBoard

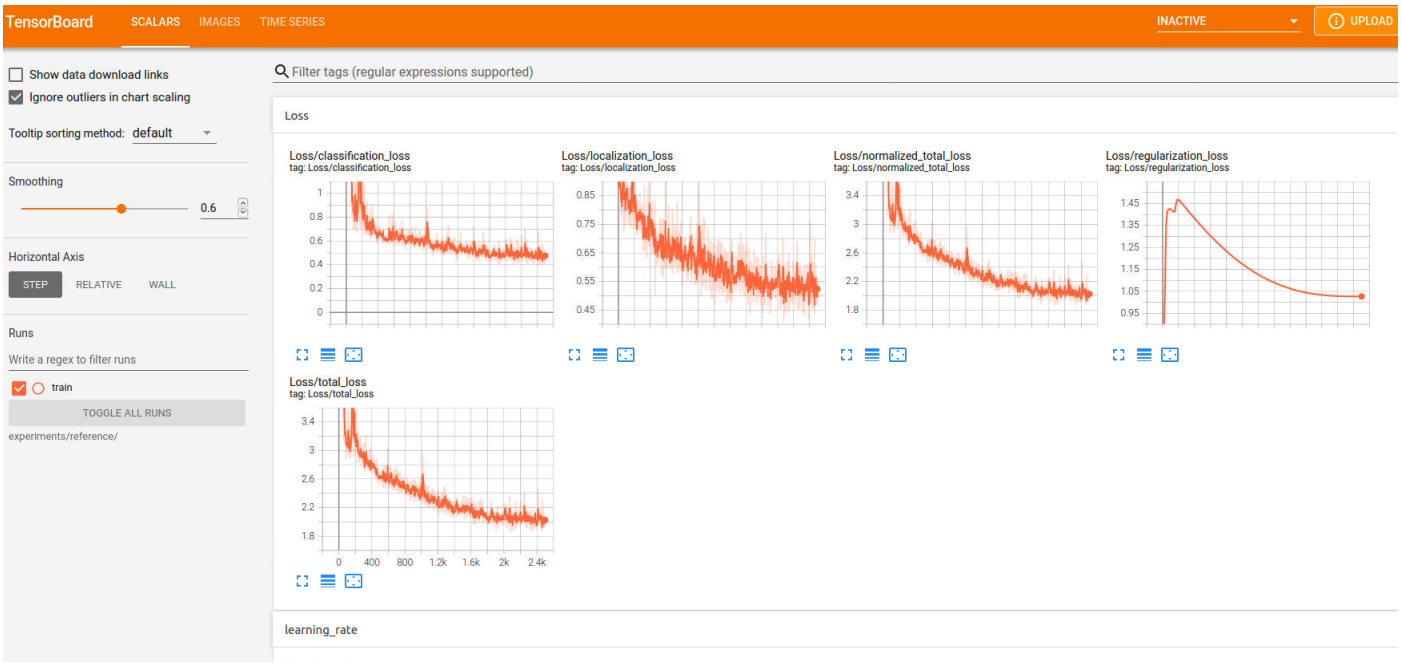

Figure 13: TensorBoard