UDACITY

‹ Return to Classroom

Object Detection in an Urban Environment

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Awesome work Udacian ⭐!

You have done an excellent job in covering all the required specifications for this project, congratulations on passing the project. 🎉 🎉

Great work done in performing the required exploratory data analysis and augmentation by using the provided dataset, the code written in create_splits.py works great to perform the data split according to the defined cross validation strategy.
The required functions have been implemented in Exploratory and augmentation notebooks. ✌️
Your write-up about the project work is very detailed, thank you for sharing the GitHub link to your code repository as well-https://githubmate.com/repo/khaled-98/nd013-c1-vision-starter. 👏

P.S. In this project you have developed an Object detection based on SSD. Remember that single-stage architectures are preferred for autonomous vehicles because they provide a higher frame rate than other architectures (e.g. Dual-stage encoders such as R-CNN).
However, SSD is not the only single-stage encoder network. YOLO is another great single-stage network that is very popular in the automotive industry. For instance, you can see a comparison of both in the following paper from a couple of years ago. A relevant part of this paper is the comparison chart between YOLO and SSD.

Thus, in general, SSD is a good approach to balance between FPS and mAP performance. Whereas, YOLO focuses more on FPS performance providing a better real-time method. You can find a good introduction to YOLO in here.

Github and Code Quality

All the code generated for this project lives in a Github repository and a link to this repository has been provided to the reviewer.

The student made at least five commits to this repository.

Great work here!
✓ The source code for this project is within a Github repository.
✓ The link to the Github repository was provided: https://githubmate.com/repo/khaled-98/nd013-c1-vision-starter
✓ Commits have meaningful messages.
✓ Separate folders are used to organize files.

A very good tutorial on git control can be found in Udacity's free courses here.

All the code written for this project should follow the PEP 8 guidelines. Objects have meaningful names and syntax. Code is properly commented and organized. Imports are correctly ordered.

Good job!

- The syntax/names are meaningful with respect to the context.
- Code is properly commented with proper ordering of imports in each of these files.

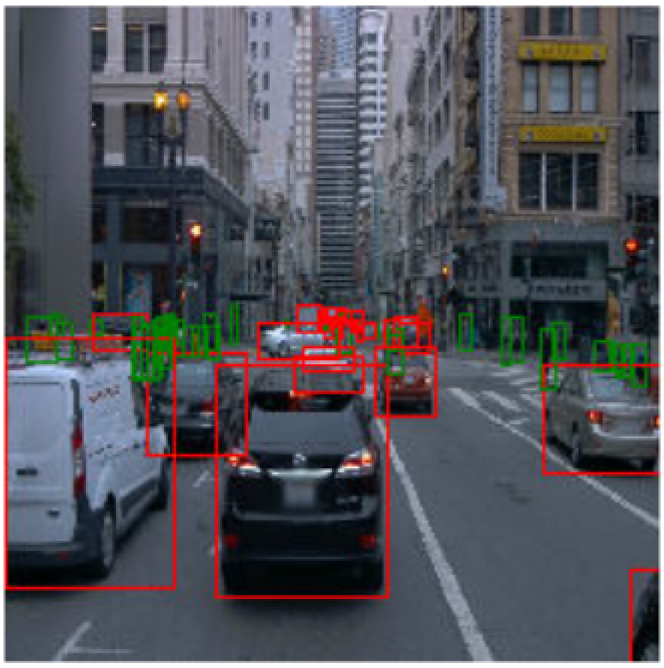
The project contains either a requirements.txt file or a Dockerfile. The instructions to install the dependencies are clear. The file contains a summary, a build section as well as a detailed description of the different functions and files. Someone should be able to run the code by reading the README.

Write-up is holding required information to run the solution and instructions are clearly written. It also details about the implementation being done and strategy used for experimentation. 👌

Exploratory Data Analysis and Split Creation

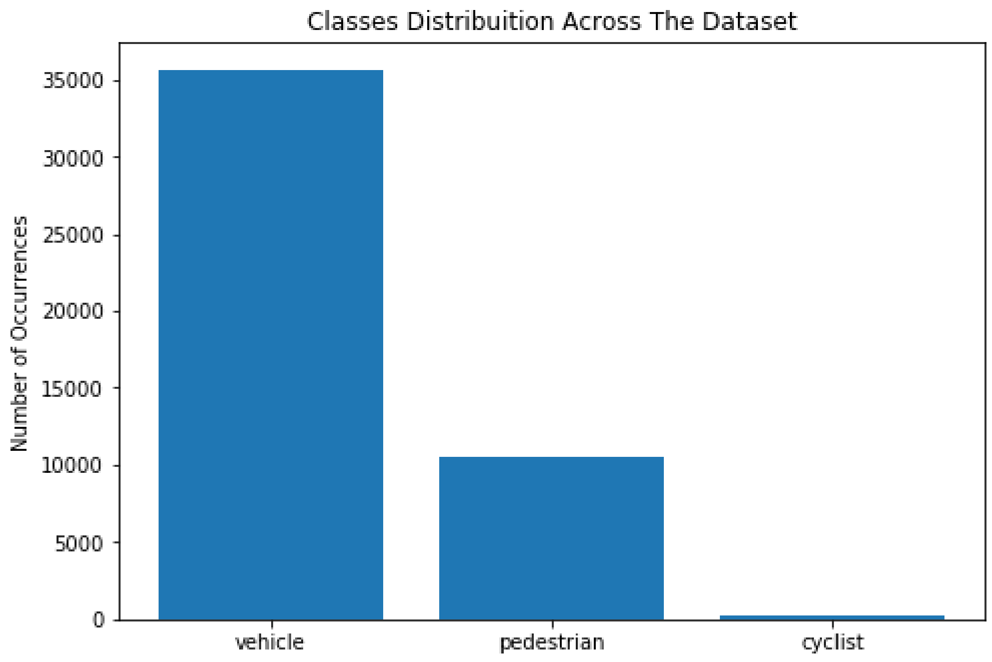
The figures should display all the meaningful information. They are big enough, it is easy to grasp the message that the figure is conveying. When plotting charts, the axis should be labeled and the figure should be named. When plotting bounding boxes, the different classes should be color coded.

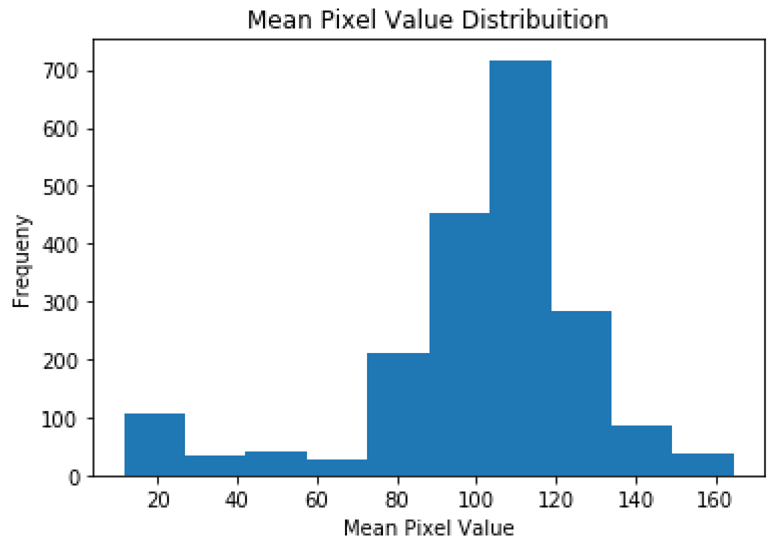
The color spread in dataset is shown in the plotted chart 👏. The bounding boxes with suggested colors are drawn for vehicles and pedestrians. Good job!



The write up and the code capture the dataset variability (classes distribution, images variability).

The write up and the code captures the dataset variability.





The write up justifies the cross-validation strategy and the code reflects this strategy.

Necessary cross validation strategy details have been shared.

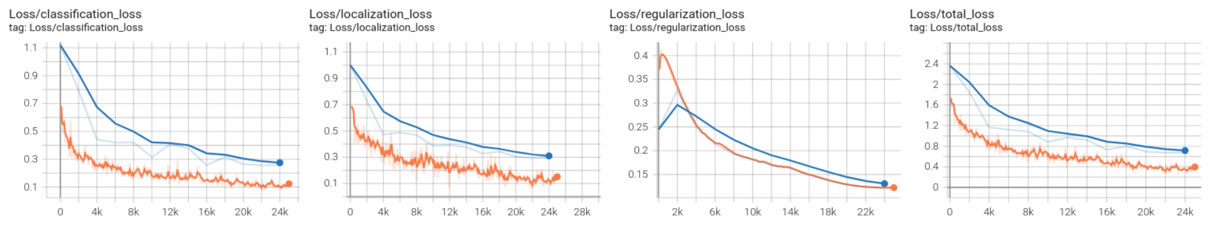
"Given the relatively small number of images provided in this dataset, 75%, 15% and 10% of the images were allocated to the training, validation and testing sets, respectively. A larger share for the training set would have been more ideal; however, that would make the validation results less reliable.

The split was made based on the files, as opposed to combining all the images and randomly allocating them to different sets. This ensured that images from a given journey were only allocated to one set, rather than split between multiple sets. This was done to minimise the chance of overfitting and to ensure that the model generalises well for different journeys/conditions."

Training

The write up contains a screenshot of the different Tensorboard charts and the write up describes these charts. For example, how does the validation loss compare to the training loss? Did you expect such behavior from the losses / metrics?

Very well done in performing multiple experiments, it shows how hard working you are. ✌🏽
The charts look great!



Model Improvements

A new version of the config file is created and contains modifications to improve the model performances. A new config file is created with meaningful modifications.

The write up details why these modifications were made. New augmentations are visualized and displayed in the writeup.

Great job in performing detailed experimentation and trying out different augmentations, new config files have been shared and experimentation details are also added in the write-up document.

I am sharing a very nice paper on multispectral object detection for your learning purpose.

The write up demonstrates that the student investigated at least two modifications of the config that were not discussed in the course and referenced the corresponding Tf Object Detection API code documentation.

Good job here!
A very nice read on Improving small object detection in YOLO, hope you enjoy it!

DOWNLOAD PROJECT