DLE305
Assessment 3 - Object Detection and Tracking

Introduction

This project involves the detection of road vehicles in a range of scenarios. The goal is to accurately detect, categorize and count vehicles from videos. This project is important as it has real world applications such as traffic monitoring, autonomous driving and even urban planning. Detecting vehicles is also applicable to other things like objects. There are challenges associated with this project such as processing the videos into readable data for the model and also classifying vehicles into categories like cars, trucks and buses.

Project Brief

The primary objective of this project is to develop an application that can easily take in videos, process them into frames then detect the vehicles in them. Afterwards, analysis can be done to see how well they performed. I found that the model can struggle with accuracy at times and being consistent.

Programming Environment

For my IDE I used Jupyter Notebook as it is well suited and known for being a good IDE. I used Python to develop the whole application as it has a lot of good packages that are easy to use, also my experience lends me more to Python. The project uses a lot of packages such as, pandas, tkinter, matplotlib, numpy, torch and cv2. Numpy and Pandas are standard packages and used in most projects. tkinter is used for the GUI's and displaying videos/results. matplotlib is used in the project to plot the results of the vehicle counts and cv2 is used to manage and load csv files. I also used CUDA for the ability to utilize my GPU for processing, this sped up the process greatly.

Dataset Description

The dataset given and used in this project is a list of 42 videos of vehicles along with corresponding txt files containing tracking data. This is a great dataset as it has a lot of real-word examples of traffic on the road which is perfect for what the project calls for. Some of the videos are a bit wobbly in the way they are filmed but this does not become a problem. In addition to those videos, I personally filmed 5 videos of traffic in my local area for testing. When extracting the frames of these videos for processing I used OpenCV. I extracted every 2nd frame to improve the processing time speed.

Implementation Methodology

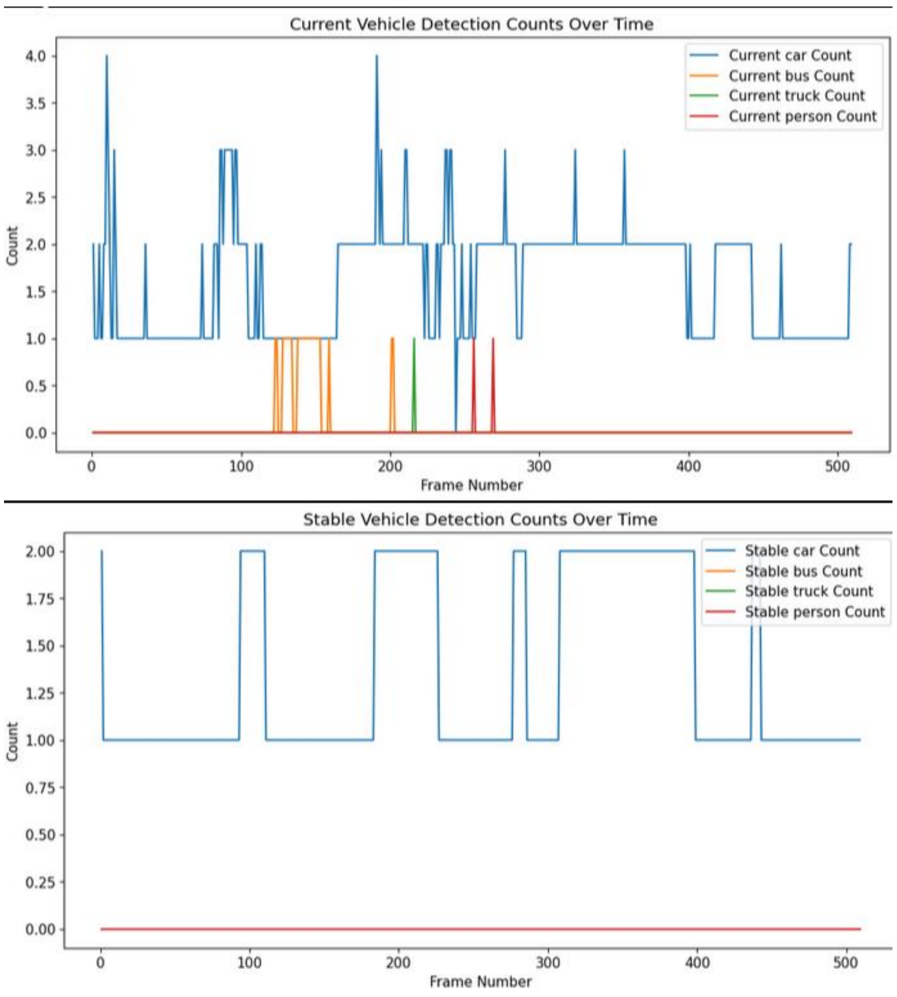
The key driver of this project is the YOLO (You Only Look Once) model. I chose this model because of its great speed and accuracy. Along with the wide range of detectable objects it can be applied to multiple problems, so it is good to learn and use now, for potential future projects. I configured the model to only detect cars, buses, trucks, people and fire hydrants. The first steps of this project are to load the dataset and track the tracking data into the videos without the use of any model, simply using the provided tracking data to show what the final result should look like. I used pandas, cv2 and os libraries for this process. I loaded all the video files and then looped over all of them. During the loop I also loaded the corresponding tracking data which was in a .txt file, I drew the bounding boxes over the video frames to get a sequence of tracking for the car. This showed what accurate tracking of a single car would look like and what I would aim to replicate with a deep learning YOLO model. The tracking however only showed a single car and didn't show multiple cars or vehicle types.

Developing the model to take any video and track the cars in it was the next step. I started by loading and configuring the YOLO model. The model can be configured to detect certain objects from a list of choices. [0, 2, 3, 10, 5, 7] is the configuration I chose, 0=person, 2=car, 3=bus, 5=motorcycle, 7=bicycle, 10=fire hydrant. I found these to be a good combination as the videos showed a lot of cars and some of the rest. The main process involved tweaking and testing the settings of YOLO. Initially, a skip rate of 2 was selected, this meant that every 2nd frame would be analyzed instead of all of them. With every 2nd frame being processed this sped up the overall processing time and didn't have any side effects. When comparing the accuracy of the tracking it was found that due to it skipping every 2nd frame the accuracy was low, after changing it back to every frame the accuracy was showing as it should but for testing purposes every 2nd frame worked well. Applying a buffer to the model allowed for the integration of a stable count, this made analyzing the data trends much easier as it made the count much more consistent and easier to read. I still kept the raw unsmoothed data so I could analyze the data in rawer and unedited form.

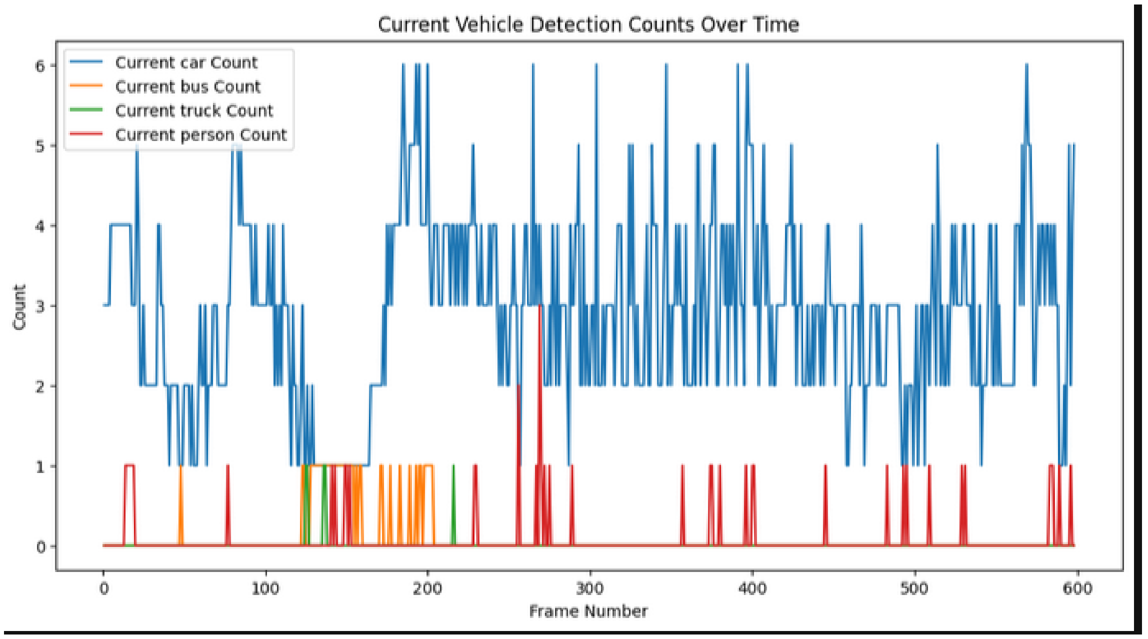
Results Analysis and Key Insights

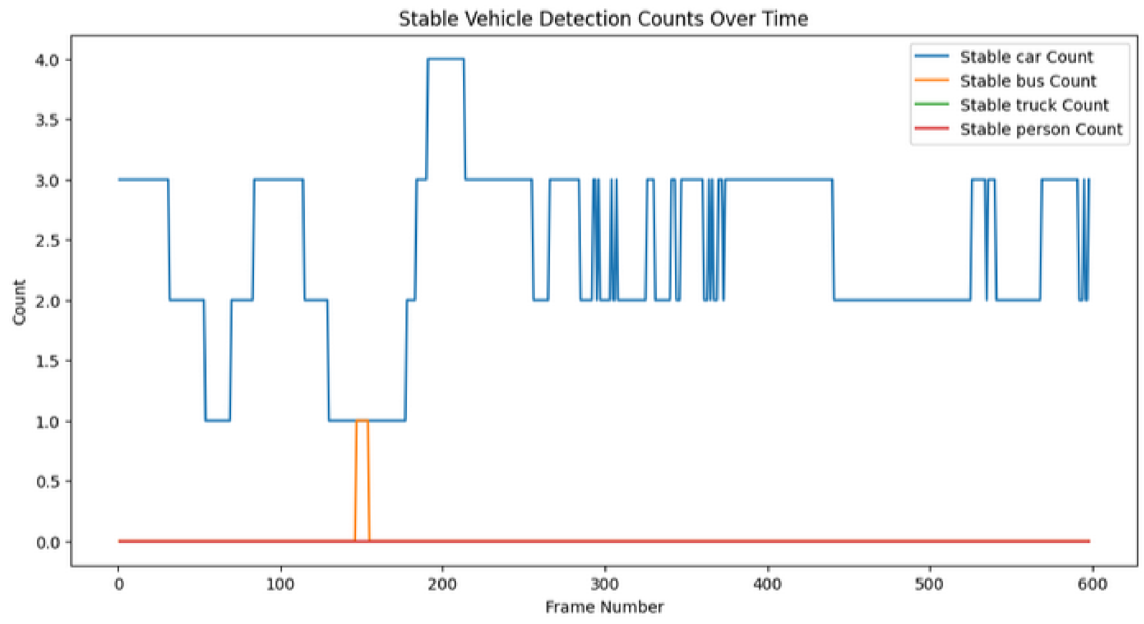
Results showed a very positive and accurate tracking of most vehicles in the scene. A key detail is that I set the threshold of detection to 0.5, this key detail changes the detection to only show objects with a high confidence. This could be edited to be lower or higher depending on the project goals, I settled for a mid-choice so that not every single object

was being detected and to ensure that what was being detected was accurate.
honda_civic_2018_purple_02:



This graph shows detection with a threshold of 0.5. Accuracy of IOU was 65.270%





This graph shows detection with a threshold of 0.25. Accuracy of IOU was 65.276% Testing with this setting shows a big difference in model results. It shows a threshold of 0.5 has a more stable number count but doesn't detect smaller vehicles in the distance, whereas the 0.25 threshold shows that it does detect the far away vehicles but is a lot more variable and unstable. Both of these have their use cases for different problems.

Conclusion

In conclusion, I used the YOLOv5 model to detect vehicles within videos. A lot of changes were made and edited to find a good balance for testing. I found that depending on how sensitive you want the model to be you can configure the model to be better for different use cases depending on how accurate you would prefer the model to be.