Vehicle detection and tracking

Thesis · July 2021

DOI: 10.13140/RG.2.2.15030.22085

2 authors:

M.N Tondra
Memorial University of Newfoundland
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Ebrahim Karami
Memorial University of Newfoundland
72 PUBLICATIONS **860** CITATIONS
SEE PROFILE

Vehicle detection and tracking

COMP 4301 Project
Final Report
Name: Meheroon Nesa Tondra
Student id: 201555661

Abstract

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class in digital images and videos. Machine learning can be used to detect and classify objects in images and videos. Vehicle detection, also known as computer vision object recognition, is basically the scientific methods and ways of how machines see rather than human eyes. Vehicle detection is one of the widely used features by companies and organizations these days. We can use computer vision to detect different types of vehicles in a video or real-time via a camera. Vehicle detection and tracking finds its applications in traffic control, car tracking, creating parking sensors and many more. We have used the methods discussed in Dimililer et al. (2020) paper of Vehicle detection and tracking to built a system using OpenCV and Python for both images and videos that is able to detect and track the vehicles automatically. We used a dataset consisting of 8792 vehicle images and 8968 non-vehicle images from a combination of the GTI vehicle image database, and the KITTI vision benchmark suite. 5568 features were extracted for training using three feature extraction techniques: Spatial Binning, Color histogram, and Histogram of oriented gradients(HOG). We have used LinearSVC, which is one of the classifiers in SVMs, to predict vehicles in this project. Then we used stratified 10-fold cross validation to assess model performance in terms of the area under the ROC curve (AUROC). LinearSVC accuracy was around 99.37% on the test split of the dataset. Hog sub-sampling was used for implementing the sliding window approach with starting position of the window search from 350px to 656px and cells_per_step to 1. The pipeline video was created by processing each frame of the video images with hog sub-sampling and combining overlapping detections.

Table of Contents

Introduction

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class in digital images and videos. Machine learning can be used to detect and classify objects in images and videos. Vehicle detection, also known as computer vision object recognition, is basically the scientific methods and ways of how machines see rather than human eyes. Vehicle detection is one of the widely used features by companies and organizations these days. We can use computer vision to detect different types of vehicles in a video or real-time via a camera. Vehicle detection and tracking finds its applications in traffic control, car tracking, creating parking sensors and many more.

Dimililer et al. (2020) presented machine learning approach for predicting vehicles in images and videos. Dimililer et al. (2020) have discussed the following steps to write a software pipeline to detect vehicles in a video:

- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier (Dimililer et al., 2020).
- Implement a sliding-window technique and use your trained classifier to search for vehicles in images (Dimililer et al., 2020).
- Run pipeline on a video stream and create a heat map of recurring detections frame by frame to reject outliers and follow detected vehicles (Dimililer et al., 2020).
- Estimate a bounding box for vehicles detected (Dimililer et al., 2020).

In this project, we have used the methods discussed in Dimililer et al. (2020) paper of Vehicle detection and tracking to built a system using OpenCV and Python for both images and videos that is able to detect and track the vehicles automatically. We have used a dataset consisting of 8792 vehicle images and 8968 non-vehicle images from a combination of the GTI vehicle image database and the KITTI vision benchmark suite. We have utilized Histogram of Oriented Gradients (HOG) feature extraction on a labelled training set and trained one of Support vector machines (SVMs) classifier. We have implemented a sliding-window technique and used the trained classifier to search for vehicles in images. From the object detection result, we have assigned an object tracker, the tracker would be following the target by re-detecting it in the sequence frame following the first point of the target. The identification of vehicles is shown by drawing a bounding box around it.

Methods

Data collection
Image datasets were obtained from a combination of the GTI vehicle image database, the KITTI vision benchmark suite, and image frames extracted from "_video.mp4" file (Karunakaran, 2017).

Data processing and other programming scripts were written in Python 3.8.2. The python scripts were run via Slurm jobs in Compute Canada with modules Python 3.8.2 and scipy-stack opencv. The positive dataset consists of 8792 vehicle images and the negative dataset consists of 8968 non-vehicle images. The vehicle images were downloaded to a "vehicles" folder and the non-vehicle images were downloaded to a "non-vehicles" folder in the project directory. Both the positive and the negative datasets are loaded as numpy array to separate lists from their respective folders.

Feature extraction
Three following feature extraction techniques have been used to train the classifier to detect the cars efficiently:

- Spatial Binning: We performed spatial binning on each image by using OpenCV (Bradski, 2000). We have used OpenCV's function 'cv2.resize()' to scale down the resolution of the image and numpy's 'ravel()' function was used to convert to a 1D feature vector (Dimililer et al., 2020).
- Color histogram: A color histogram is a simply a histogram that shows the color level for each individual RGB color channels (Dimililer et al., 2020). Here, we have computed histograms of the three colour channels (Red channel, Green channel, Blue channel), and concatenate the histograms into a single feature vector.
- Histogram of oriented gradients(HOG): The HOG is a feature descriptor used in computer vision and image processing for the purpose of object detection. This feature descriptor is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information (Dimililer et al., 2020). The technique counts occurrences of gradient orientation in localized portions of an image. We have used the 'hog' function from Python extension skimage.feature to extract features and are included in the Result section (Figure 1).

A sample size of 8750 from both the positive and the negative datasets were taken to extract 5568 features for training using the three feature extraction techniques mentioned above.

Model training and assessment
Support vector machines (SVMs) (Cortes & Vapnik, 1995) are a set of supervised learning methods used for classification, regression and outlier detection. LinearSVC is one of the classifiers in SVMs which was used as one of the models in Dimililer et al. (2020) paper; therefore, we have also used LinearSVC for this project to detect vehicles.

The following parameters on Table 1 were found to be the optimum parameters for prediction (Karunakaran, 2017) and we have chosen these parameters to train the classifier to predict vehicles :

Table 1: Parameters for classifier

Parameter name	Parameter value
color_space	'YCrCb'
spatial_size	(16, 16)
orient	8
pix_per_cell	8
cell_per_block	2
hog_channel	'ALL'
hist_bins	32
scale	1.5
spatial_feat	True
hist_feat	True
hog_feat	True

The StandardScaler() function assumes data is normally distributed within each feature and will scale them such that the distribution is now centred around 0, with a standard deviation of 1; 'x' values are transformed using the function and get the output scaled_X (Dimililer et al., 2020). 'train_test_split' from Python extension sklearn.model_selection was utilized to help split the dataset into train and test data for the LinearSVC classifier.

LinearSVC was fitted on the training set. After training was completed, test dataset was used to check the accuracy of the classifier from the 'score' function of the classifier. To assess the model, we have utilized StratifiedKFold from Python extension sklearn.model_selection. StratifiedKFold cross-validation object is a variation of K-fold that returns stratified folds, and the folds are made by preserving the percentage of samples for each class (Buitinck et al., 2013). We used n_splits=10 for 10-fold stratified cross-validation on the model. To evaluate the output of the cross-validations, we utilized the area under the ROC curve (AUROC) from Python extension sklearn.metrics. Below we define the Receiver Operating Characteristic (ROC) curve.

Receiver Operating Characteristic (ROC) metric is used to evaluate classifier output quality using cross-validation. ROC curves typically feature true positive rate on the Y axis, and false positive rate on the X axis (Buitinck et al., 2013). This means that the top left corner of the plot is the "ideal" point - a false positive rate of zero, and a true positive rate of one (Buitinck et al.,

2013). There is a red diagonal line in the ROC curve (Figure 2 in Result section) that represents a random classifier. From the curves described in the Results section, it is possible to calculate the mean area under curve, and see the variance of the curve when the training set is split into different subsets (Buitinck et al., 2013). This shows how the classifiers' performance is affected by changes in the training data, and how different the splits generated by K-fold cross-validation are from one another (Buitinck et al., 2013).

Sliding-window technique

In the context of computer vision, a sliding window is rectangular region of fixed width and height that "slides" across an image (Dimililer et al., 2020). For each of these windows, the window region was taken and the trained LinearSVC classifier was applied to determine if the window has vehicles in the images extracted from the video (Dimililer et al., 2020). The Figure 3 below shows the positive detections reported by LinearSVC on the test images extracted from the video.

Figure 3. Six test images with overlapping positive detections reported by LinearSVC.

Hog sub-sampling is more efficient method for doing the sliding window approach as it computes individual channel HOG features for each entire image and uses the scaled features to make prediction. Each window is defined by a scaling factor where a scale of 1 would result in a window that's 8 x 8 cells then the overlap of each window is in terms of the cell distance; this means that a cells_per_step = 2 would result in a search window overlap of 75% (Dimililer et al., 2020). The hog sub-sampling helps to reduce calculation time for finding HOG features and thus provided higher throughput rate (Dimililer et al., 2020).

Hog sub-sampling was used for implementing the sliding window approach with starting position of the window search from 350px to 656px and cells_per_step to 1 as mentioned in Dimililer et al. (2020) and in Karunakaran (2017). In order to combine overlapping detections

and remove false positives, heatmap and threahold limit were used (Dimililer et al., 2020). A heat map of recurring detections was created frame by frame to reject outliers.

Pipeline on a video stream

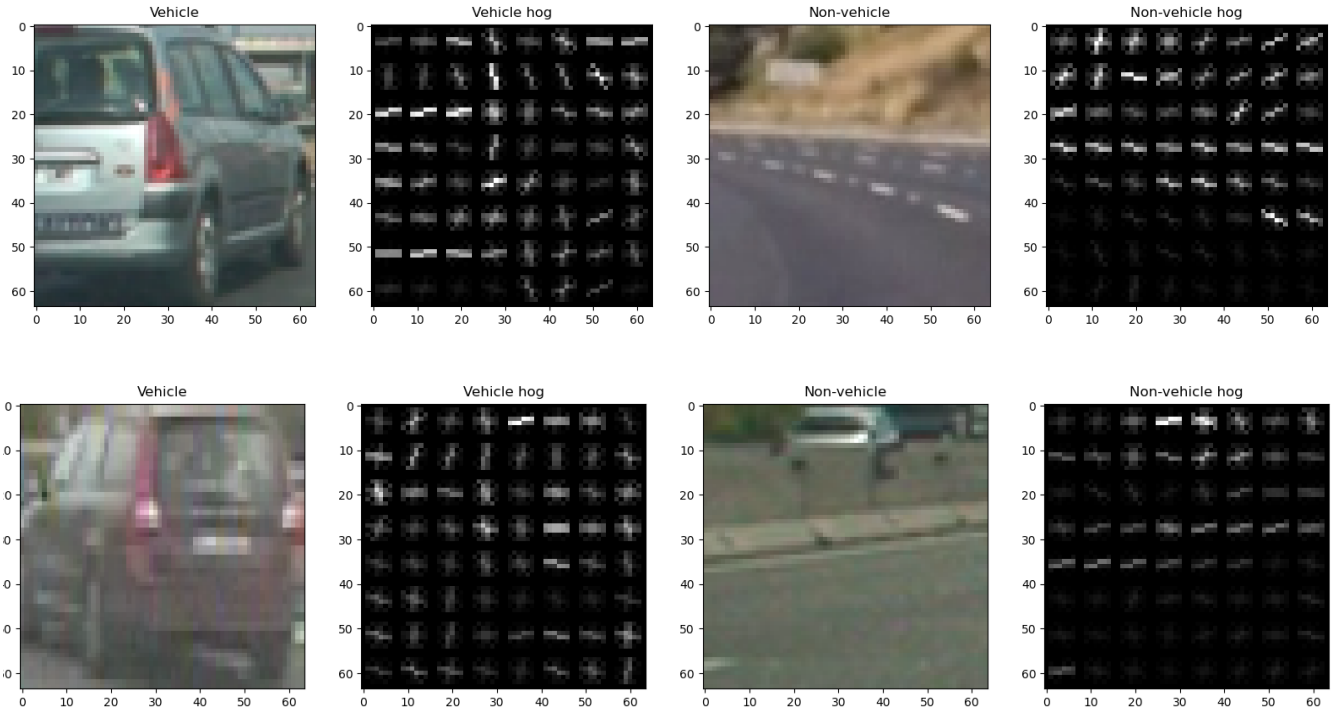
The pipeline video was created by processing each frame of the video images with hog sub-sampling discussed previously.

All the bounding boxes detected were extracted for the cars in the image, and ‘heat_threshold’ function was used to combine overlapping detections and remove false positives as mentioned in Dimililer et al. (2020). The output video, called ‘processed_video.mp4’, was produced with bounding box added to the image and was submitted in the project dropbox along with other materials.

Results and discussion

In the HOG feature descriptor, the distribution of directions of oriented gradients are used as features. Figure 1 below show the following:

- A vehicle image was randomly selected from the positive dataset and is shown on the right with its extracted hog feature.
- A randomly selected non-vehicle image was taken from the negative dataset and is shown on the left with its hog feature.



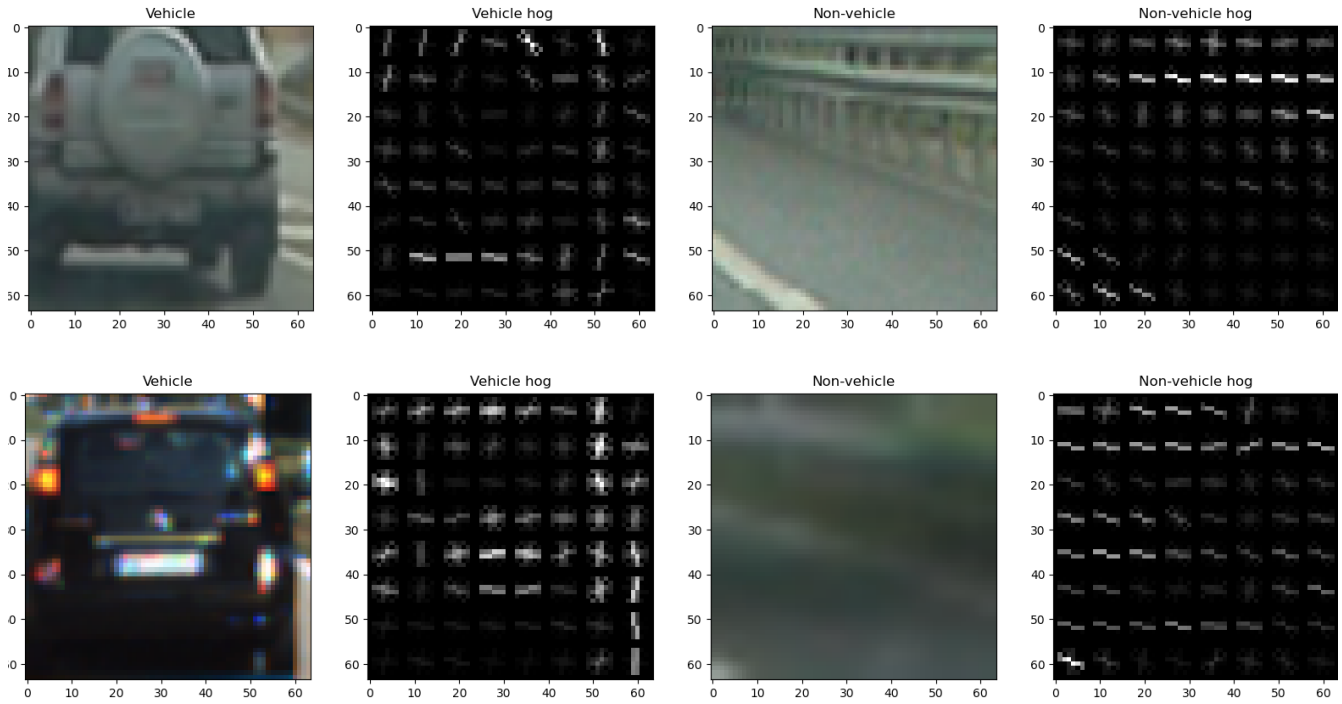


Figure 1: HOG features for vehicle and non-vehicle images.

The score obtained after training LinearSVC as the test accuracy was 0.9937, i. e. 99.37%. After running StratifiedKFold cross-validation, we have acquired values for the area under ROC curve (AUROC) for the model shown in Figure 2. We can observe that LinearSVC has a great True poositive rate and a mean AUROC of 0.99 ± 0.01.

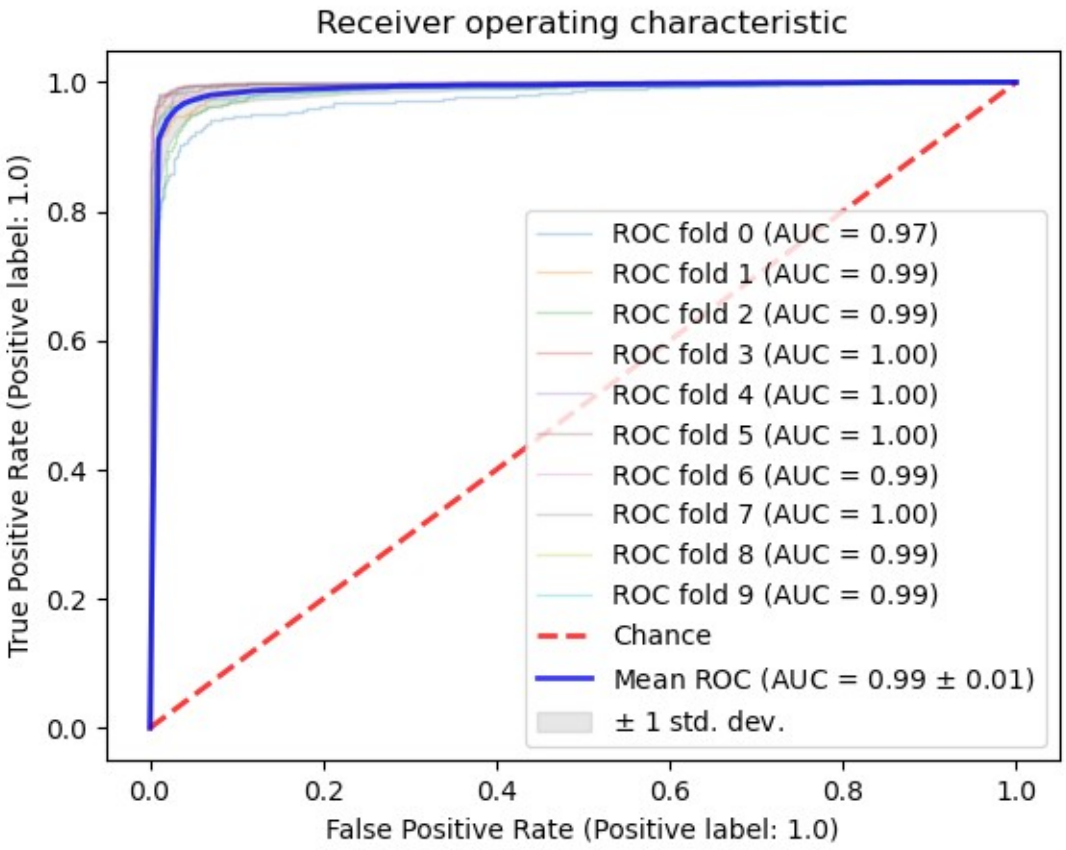
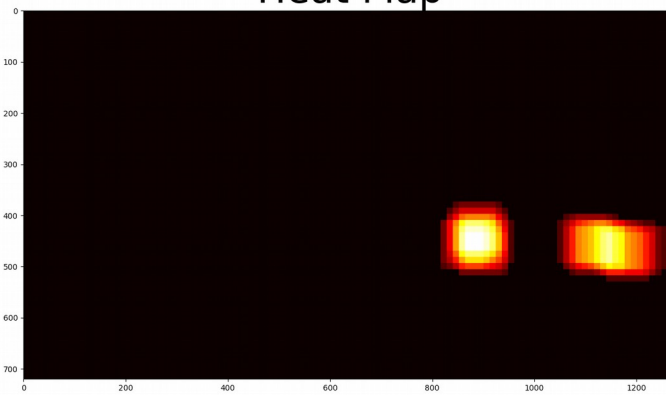


Figure 2. ROC curves for LinearSVC. The blue line indicates the average ROC across all CV folds and the grey area indicate 1 standard deviation. Numbers between brackets are the AUROC.

The results of heat map and thresholding for the sliding window technique after combining together is faster with hog sub-sampling. Figure 4 shows the cars found using hog sub-sampling search window with it's heat map after thresholding. We can observe from the figure that heat map is black (with no heat) when there is no car on the road.

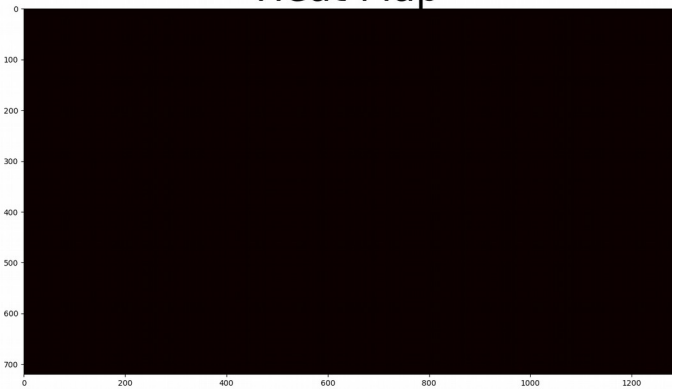
Car Positions

Heat Map



Car Positions

Heat Map



Car Positions

Heat Map

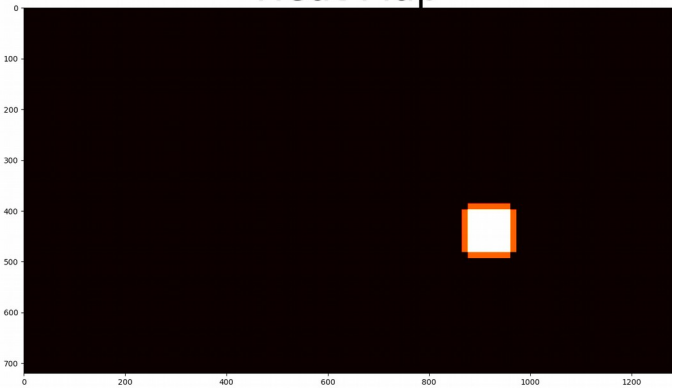


Figure 4. Hog sub-sampling for 4 test images with heat map.

Conclusion

We used the methods discussed in Dimililer et al. (2020) paper of Vehicle detection and tracking to built a system to detect and track the cars.
Using SVM, we have obtained a good result, but this cannot be generalized. So, this project could be done with other machine learning algorithms.
As a future work, this can be extended to detect lanes on the road, and can be used to find distance or speed of the vehicles.

References

1. Dimililer K., Ever Y.K., Mustafa S.M. (2020) Vehicle Detection and Tracking Using Machine Learning Techniques. In: Aliev R., Kacprzyk J., Pedrycz W., Jamshidi M., Babanli M., Sadikoglu F. (eds) 10th International Conference on Theory and Application of Soft Computing, Computing with Words and Perceptions - ICSCCW-2019. ICSCCW 2019. Advances in Intelligent Systems and Computing, vol 1095. Springer, Cham. https://doi.org/10.1007/978-3-030-35249-3_48

2. M. V. G. Aziz, H. Hindersah and A. S. Prihatmanto. (2017). Implementation of vehicle detection algorithm for self-driving car on toll road cipularang using Python language. *2017 4th International Conference on Electric Vehicular Technology (ICEVT).* Sanur, 2017, pp. 149-153. doi: 10.1109/ICEVT.2017.8323551.

3. Bush, I.J., Dimililer, K.(2017). Static and dynamic pedestrian detection algorithm for visual based driver assistive system. *ITM Web Conf, 9*(03002). doi: https://doi.org/10.1051/itmconf/20170903002

4. Karunakaran, D. (2017, December 10). *Vehicle Detection and Tracking: Udacity's Self-driving Car Nanodegree*. Medium. https://medium.com/intro-to-artificial-intelligence/vehicle-detection-and-tracking-udacitys-self-driving-car-nanodegree-ca02330820ee

5. Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's* Journal of Software Tools.

6. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

7. Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., Vanderplas, J., Joly, A., Holt, B. And Varoquaux, G. (2013). *API Design For Machine Learning Software: Experiences From The Scikit-Learn Project*, 108–122. https://arxiv.org/abs/1309.0238

8. Hortovanyi, N. (2017, February 24). *Vehicle Detection and Tracking.* Towards Sata Science. https://towardsdatascience.com/vehicle-detection-and-tracking-6665d6e1089b