EE769 : Introduction to Machine Learning | Course Project

Vehicle Detection for Self-driving Cars

Sandesh Gaikwad | 203190030

Supervisor : Prof. A. Sethi

Department of Electrical Engineering
Indian Institute of Technology , Bombay

Vehicle Detection for Self Driving Cars

1st Sandesh Gaikwad
Mumbai , India

2nd Kshitij Kushwaha
Mumbai , India

***Abstract*—Self-driving cars are slowly becoming the reality of our future. For cars to run autonomously, they first require identifying objects present in their environment and then making appropriate decisions. Electric vehicles have gained massive popularity in recent times due to various reasons. One of them is providing an advanced driving assistance system, which has increased road transport safety by providing vital information of driving environment and suggesting a better decision to user depending on various factors. A precise and robust vehicle detection module is a vital part of such a system. Our project is aimed at addressing this issue. We have used histogram of oriented gradients (HOG) to extract features from dash cam present in front of the car and trained supervised machine learning models to classify the cars and not-cars according to HOG features and use the model to detect the objects present in the driving environment. The primary focus was kept on having maximum precision and accuracy as it is vital to precisely detect the vehicles in front of the car to avoid the collision. The model can take the video file as an input and return the video file with bounding boxes placed on a detected vehicle.**

***Index Terms*—Histogram of oriented gradient(HOG),Vehicle Detection,machine learning,self-driving cars**

I. Introduction

The number of vehicles on the road are increasing rapidly, and with such increasing traffic on the road, traffic safety has become a concern recently. In many road accidents, human errors are the primary reason. Such errors can be avoided by automating some driving decisions or providing vital information to the user about the surrounding environment based on data analysis from various sensors and cameras. To develop such a system vehicle detection module plays a vital role. The module can correctly identify the vehicles in front of the car and warn the user. The module must be fast and precise enough to be implemented in real-time. The use of vision-based vehicle detection system instead of sensors is gaining tremendous popularity due to its richness of information content and low cost [1], and the use of such techniques, with HOG features and machine learning model, has made significant development in vehicle detection systems.

Many factors need to be considered while developing such a system. For example, the number of features is tremendous, and to train models for such high features, the model takes much time, which can make the system ineffective in a real-time world where we need a fast prediction to avoid the collision. Our training images were 64x64 RGB images. If we had used the raw pixel values themselves as a feature, then the feature dimension would have been 12288x1. Not all the dimensions are important, so we need to transform the data to a lower dimension and also, we need to ensure that important information is not lost in doing so. The quality of the image captured by the dash cam is affected by pollution in the air, light conditions, weather, and many other factors that can also affect the decision-making process. While implementing the system, choosing the right features and hyper-parameters is the primary concern. Also, in search of higher accuracy, the model may over-fit and perform poorly on test data. All these factors play a critical role in deciding the applicability of the system in the real-world environment.

The developed system has considered an exclusive feature selection to determine the best features, giving a better estimate of the HOG feature and improved the model accuracy, which is not present in most other work. Instead of focusing on just one model framework, we have experimented with various types of model frameworks proven beneficial in different research articles. Based on accuracy and precision score, we chose the best model framework to be implemented to detect vehicles. We have also carried out k-fold cross-validation to determine the hyper-parameters best suited for our data and avoid over-fitting data. We have tested the system on real-world traffic footage images to see how it performs on real-world systems and got good predictions. Over model achieved more than 97% accuracy on validation data and very few false positive and false negative compared to true positive and true negative. The model predicted precisely 8 out of 9 test images. The labelled data set was also created to help other developers in future while creating such systems, saving time in extracting HOG features.

II. Background and Previous work

A. Feature Engineering

While training a machine learning model, we provide the model with a data set as input, and the output we get is the learned parameters of the hypothesis on which the model was built. The data set consists of no. of examples. These examples can be a vector of size d. Then we say that we have d no. of features in our data set. Generally, the input data set also called as training data set, is written as a matrix where each row indicates an example and each column a feature.

Say we consider the raw pixel values of an image as input to our model, then the no. of features equals pixels along width $\times$ pixel along with height $\times$ channels For example, Fig(1) has 12288 ($64\times64\times3$) no. of features. What features to use when? This depends on the problem. Say you want to classify an image as to whether or not it contains a circular object. So the colour of the image is not important. So convert your RGB image to grey scale and use an edge detector filter. So

the no. of features is reduced from 12288 (64×64×3) to 4096 (64×64×1)! Note not all the pixel values are of importance to us. So we need to reduce the no. of features. Generally, the process of extracting the important features is referred to as 'Feature Engineering' or 'Feature Extraction. During feature engineering, we perform some arithmetic operations on raw pixel value. The output contains only the minimal sufficient information that could be used for further operation without compromising the quality of the results of that operation. In simple words, we find a feature descriptor representing an image (here) that simplifies the image by extracting only useful information. A feature descriptor converts an image of shape width×height×channels to a vector of shape n×1.The most popular feature descriptor used in image processing is the Histogram of Oriented Gradients (HOG) features [2]. Apart from this, other features are Spatial Features and Color Histogram Features.

B. Colour Spaces

For the classification of car vs non-car, the structural information such as edges and corners are important, so is the colours. A high saturated red colour is typically not a part of natural environment, but most cars are red. So, we need to carefully choose the colour space to train our machine learning model. Apart from RGB colour space, other colour spaces are HSV (hue, saturation, and value), YCrCb (Y-brightness, Cr-red difference, Cb-blue difference), and YUV (Y-brightness, U-blue luminance, V-red luminance).

C. Histogram of Oriented Gradient (HOG)

In the HOG, the histograms of directions of gradients are used as features. Region where there is a sharp change in raw pixel values like at corners and edges the value of gradients are high there which gives some indication of shape of the object. Steps to calculate HOG feature

1) Calculate the Gradient: To calculate a HOG descriptor, first calculate the horizontal and vertical gradients gx and gy. This is done by filtering the image with the following kernels as in fig.(1).The magnitude and direction of the gradient is calculated using [2] :

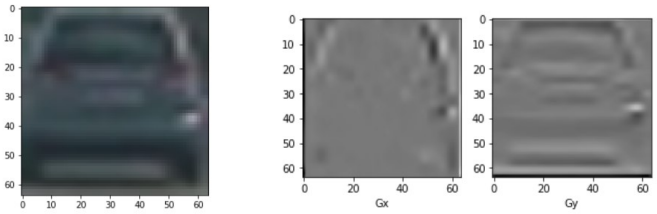

Fig. 1. Image example (left-original image, right- Sobel operator used)

$$g = \sqrt{g_x^2 + g_y^2} \tag{1}$$

$$\theta = arctan\frac{g_x}{g_y} \tag{2}$$

Use Sobel operator in OpenCV with kernel size 1. At each pixel, the gradient has a magnitude and a direction. For a multi-channel image like for RGB, the image gradient is calculated for all the channels. The magnitude of the gradient at a given pixel is the max gradient among all the channels, and the angle corresponds to the max gradient.

2) Calculate the Histogram of Gradients in cells: We have used a car image that has a size 64×64 RGB. As an example, divide this image into cells where each cell has size 8×8 non overlapping. If we used all raw pixel value of a cell; then dimensionality would be 192 (8×8×3). HOG uses only two values (magnitude of gradient and its direction) on a pixel. So, the dimensionality is now 128 (8×8×2). The size of a cell is a hyper parameter, and it is denoted as pixels_per_cell in OpenCV.

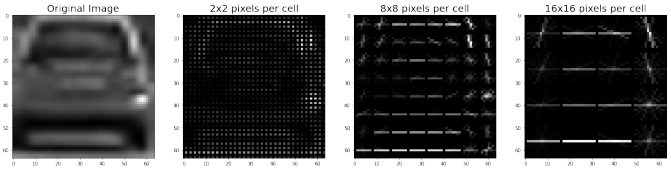

Fig. 2. Hog at different pixel per cell

Now we create a histogram of gradients in these cells. The histogram contains nine bins corresponding to 0, 20, 40 … 160 degrees. The number of bins is a hyper parameter, and it is denoted as orient in OpenCV. The number of bins less than 8 gives poor result. A bin is selected based on the direction of

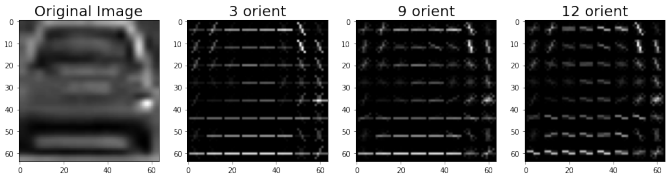

Fig. 3. HOG at different orientation

the gradient, and the value that enters the bin is selected based on the magnitude of the gradient. Say a pixel has direction 20 and value of 2, the entry in the bin corresponding to angle 20 will be 2. Now suppose the gradient direction was 10 and magnitude 2, then there will be two entries, one in bin 0 and another in bin 20, each having entry 1.The contributions of all the pixels in the 8x8 cells are added up to create a 9-bin histogram.

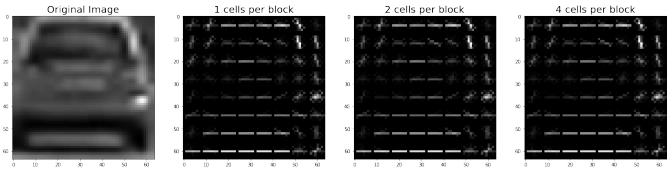

Fig. 4. HOG at different cell per block

3) Block Normalization: Now we have 64 cells each of size 8×8 and corresponding histograms. In this step, we can combine the histograms of neighbouring cells and then normalize them. This combination even outs the variation due to lighting. For example, if we select neighbouring one cell on the left side and downside of the current cell, we have 16x16 block normalization, and we refer to this as cells_per_block = 2 in OpenCV. This normalization will have 36x1 elements as normalized histogram vector. This is repeated by moving to the next cell, that is, by moving 8 pixels to the right. Note cells_per_block is a hyper parameter for HOG.

4) Calculate the Histogram of Oriented Gradients feature vector: To get the final feature vector for the entire image, the 36×1 vectors are concatenated into one giant vector. With 8×8 cell size, nine bins, and 16×16 block normalization and 64x64 original image size no. of features =1764

D. Previous work

Dalal and Triggs (2005) have proposed using HOG features and linear SVM for Human detection [3]. This made use of HOG features for image classification very popular. Huan Wang and Hiachuan Zhang (2014) proposed a two-step approach to develop a vehicle detection system. In the first step Harr like features along with AdaBoost is trained to detect car features, then HOG and SVM based approach is used to verify the vehicle detection, this way was proved to be useful for real-time vehicle detection [4].Arunmozhi, Ashwin, and Jungme Park (2018) has compared the three feature descriptors widely used in object detection, i.e. Histogram of Oriented Gradients (HOG), Haar-like features and Local Binary Pattern (LBP) and found that local binary pattern features work best on real-time vehicle detection system [5]. Bougharriou, S., F. Hamdaoui, and A. Mtibaa (2017) have proposed the Linear SVM trained using HOG features to design a vehicle detection system [6].

III. Datasets

The data set used for the project is a combination of data sets from various sources. These sources are the GTI Vehicle Image Database (GTI) [7], KITTI Vision Benchmark Suite [8] and additional images extracted from Udacity's dataset [9] and some traffic videos from YouTube [10]. GTI database contains images from different viewpoints. Udacity's project video images are used as test images along with some random images from traffic videos from youtube [11], which were shot from the car's dashcam. The datasets are open-sourced and open to use for analysis by all. While assembling the data set, we ensured that the number of vehicle images is equal to non-vehicle images to better train the model. Also, the characteristics (number of pixels, colour space) of all images are similar or not is checked. We wanted to analyze the model's performance in real-life scenarios, for that some traffic videos available from YouTube [10] , are used. Before selecting such videos, various factors were needed to be checked, like the quality of the video, file size, file format and existence of vehicle in the footage. We have trained the model for the car data set, so we needed the video where only cars are present. No other vehicles, such as trucks, bicycles, motorcycles, or buses, need to be present.

IV. Procedure and Experiments

Machine learning model with HOG features The below flowchart describes the steps in our model:

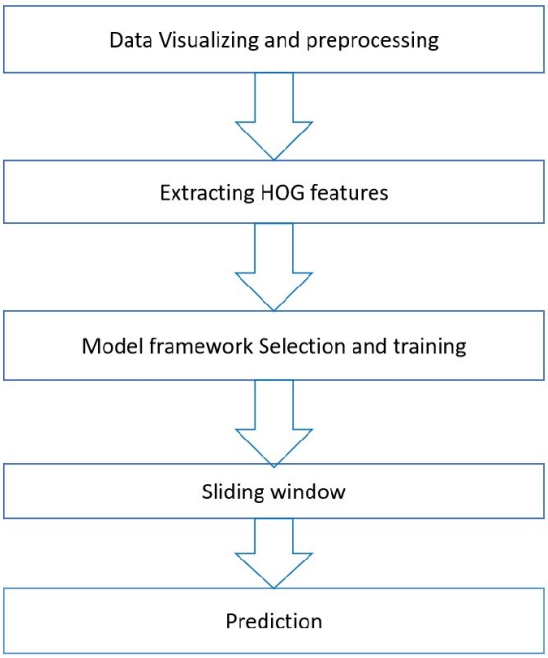

Fig. 5. Flow chart of procedure followed

1) Data visualization and pre-processing: The dataset contains two zip files containing vehicle images and non-vehicle images. The files are first extracted and then stored in separate folders for use in other process. The images from folders are plotted, and image size, pixel, shape information is obtained.

2) Extracting HOG features: For extracting HOG features, we have used the hog module from scikit-images [12]. The module returns HOG features for one channel at a time. As evident from data visualization, three channels are present. We create a function that stacks the feature from each channel together to obtain the necessary hog features for a single image with three channels or colour spaces. For all images, such hog features are obtained, and they are vertically stacked to form a feature vector.

3) Model Framework selection and training: After extracting feature and assigning binary labels to data, we shuffled and split data into training and validation data set .

- Our initial hypothesis was that perceptron could be a good model for our data as the problem is simple binary classification. We trained a perceptron classifier on training data with loss as hyperparameter and performed 5-fold cross-validation to verify this hypothesis.
- As Bougharriou, S., F. Hamdaoui, and A. Mtibaa (2017) [6], Dalal and Triggs (2005) [3] have proposed an approach of using Linear SVM with HOG features,

and we also explored the suitability of SVM for our problem. However, instead of using linear SVM, we took kernel and C value (the penalty for misclassification) as hyperparameters and performed 3-fold cross-validation to select the best hyperparameters for SVM.

- Huan Wang and Hiachuan Zhang (2014) [4] proposed a two-step approach with the first step as AdaBoost with Harr like features; this gave good results in vehicle detection. We tried the AdaBoost classifier with the HOG feature to see if it can give better accuracy than other models.
- To check the performance of the tree-based method in our classification problem, we trained a Random Forest classifier with the number of trees as a hyperparameter. The time taken to train the models and accuracy score is recorded for all models. The model framework which gave the best result is used for the final training of data.

4) Sliding Window approach: To detect the vehicle in an image containing various objects, we have used the sliding window approach. The basic idea is that we create small windows having shape more miniature than the original image and slide them over the entire image. We fix the window whenever it predicts a car in the entire image. There were many false positives in the prediction on test images. To counter that, we applied a certain threshold to eliminate them, and we verify it is a car only if the value at that window is above that threshold value.

5) Prediction: After training the model, the sliding window approach is iteratively run for all test images. Final predictions are made, including windows with car predictions and having pixel value more than the threshold value. After prediction, a heatmap and image with windows on the car are plotted to detect the location and size of the car. The nearer the car, the higher the window size will be.

V. Results

A. HOG parameter selection

TABLE I
HOG parameters and Accuracy

Colour Space	*Pixel per cell*	*Cell per Block*	*Accuracy*
YUV	8	1	0.9611
YUV	16	1	0.9659
YUV	8	2	0.9811
YUV	16	2	0.9783
HSV	8	1	0.9623
HSV	16	1	0.9662
HSV	8	2	0.9783
HSV	16	2	0.9797
YCrCb	8	1	0.9561
YCrCb	16	1	0.9707
YCrCb	8	2	0.9801
YCrCb	16	2	0.9772
RGB	8	1	0.9364
RGB	16	1	0.9505
RGB	8	2	0.9682
RGB	16	2	0.9721

Pixel per cell, Cell per Block and colour space are the hyper parameters for the HOG module used to extract HOG features from images. Efficient tuning of these value can improve the accuracy of the whole model. For various values of these hyper parameters, we have calculated the accuracy on validation data and found that colour space = YUV, pixel per cell = 8, cell per Block = 2 and orientation fixed at 9 gives the highest accuracy.hence these parameters are used to extract HOG features from all images.

B. Model Selection

We have trained 4 different machine learning model frameworks on our dataset and after 3-fold cross-validation and hyperparameter tuning found out the below results as stated in TABLE II and TABLE III.

TABLE II
Best hyperparameters for different model framework

Model Framework	*Best hyperparameter)*
Perceptron	penalty : L1
Support Vector Classifier	(C : 10 , kernel : "rbf")
Random Forest Classifier	$n_estimator$: 150
AdaBoost Classifier	$n_estimator$: 50

TABLE III
various Model frameworks performance on validation data

Model Framework	*Time to train (in seconds)*	*Accuracy*
Perceptron	8.5369	98.45
Support Vector Classifier	182.73	99.52
Random Forest Classifier	143.030	98.09
AdaBoost Classifier	438.0988	97.13

From the tables, we can see that we achieved the highest accuracy when we used the Support Vector Classifier with kernel = RBF and C=10. The value of C is the penalty for miss classification, and kernel indicates kernel trick used for data classification.The Gaussian kernel is giving better estimate than linear for accuracy measure.Hence we can use this model framework for prediction on test data set.

C. Performance of SVC

From the fig.6 Confusion matrix, we can see that number of true positives and true negatives detected by SVC are more than false negative and false positive. Hence we can say the model has very high accuracy and high precision

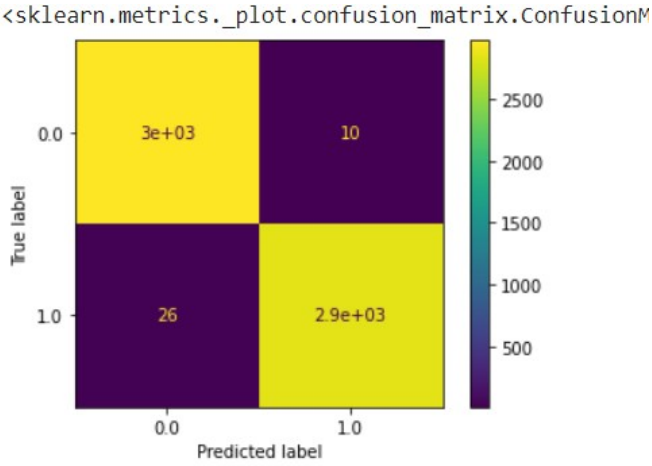

Fig. 6. Confusion matrix

D. Sliding window

As seen in fig.7 , We have generated 18041 windows on the test image, which are of different shapes starting from 24×24 to 96×96 with overlapping of 80%, and only 35 windows identify cars correctly. We have chosen only 60 % of the area of the image, which is closer to the car, to save time in generating windows and identifying cars closer to our vehicle.

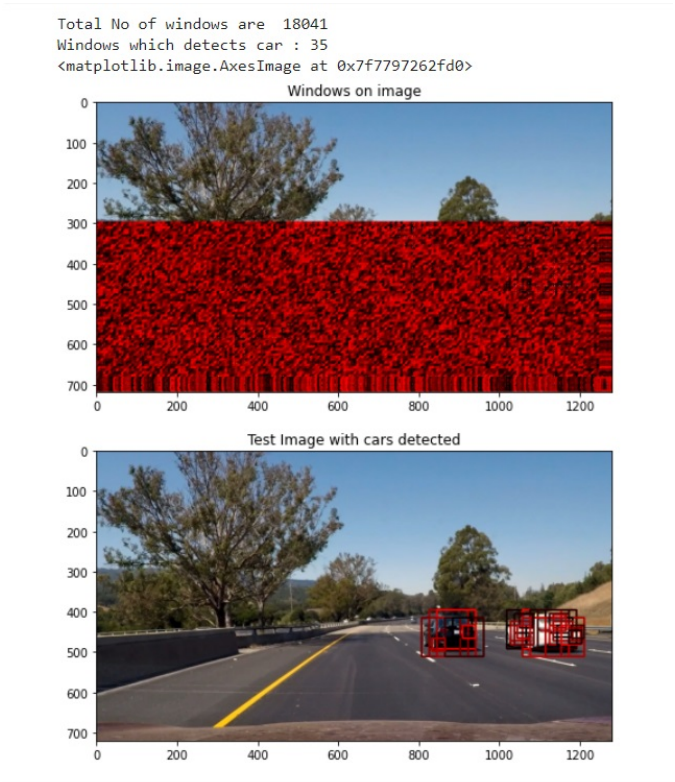

Fig. 7. Windows generated and cars predicted on test data

E. Prediction on test images

Some images of prediction using Support vector classifier with Gaussian kernel and C=10 on test data.

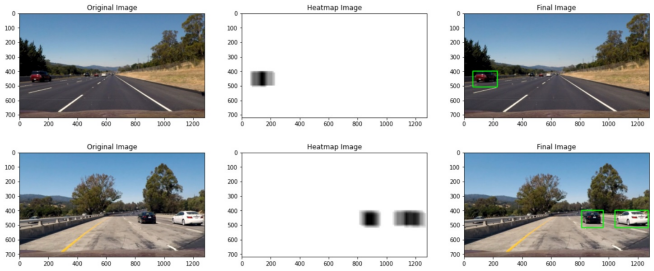

Fig. 8. prediction on test data

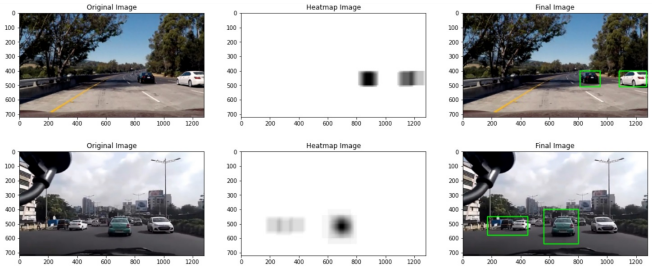

Fig. 9. correct prediction on test data

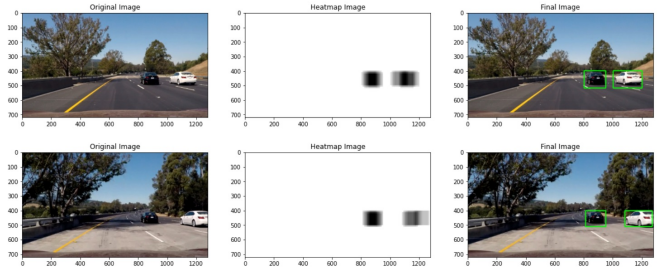

Fig. 10. correct prediction on test data

Fig. 11. incorrect window size in real world traffic image

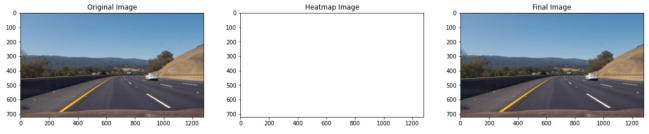

Fig. 12. False negative on test image

VI. Conclusion

We used 8×8 cell size, two cells per block and nine bins for HOG features to get the highest accuracy. From HOG parameter selection, we can claim that the HOG feature descriptor depends on the colour space, cells per block, and pixel per cell of the image. If these parameters are not correctly chosen, they may lead to reduced precision. We tested our model on nine test images, and the model correctly identified cars on eight images and drawn a perfect box on the predicted car. We also ran our model for real-world traffic videos. We observed that the model predicts correctly for certain cars, which are closer but sometimes fails to draw an appropriate box on the car leading to incorrect detection of the car's shape. Car detection using vision-based methods is highly affected by external factors such as driving environment, adequate exposure to light, humidity, the pollution level in the surrounding, quality of the camera used. Prediction also depended upon the distance of the car from the camera. Although it took very little time to train on training data, this model could not be used for a real-time car detection system because it took more than 2 hrs. to process a video of 6 sec through the video pipeline as we need to process video frame by frame and depending on speed of car the frames move too fast, and such low speed is not helpful in real-world scenarios. prediction time increases as the image size increases, as we need to generate more windows to cover the image, time increases as we need to predict and check for each and every window generated. This project can be used for a better understanding of image classification using supervised machine learning models. This project also provides a good insight into the HOG feature descriptor. One can improvise the results using Deep Learning models instead of basic machine learning models.

VII. Statement of contributions

Sandesh Gaikwad worked on : Understanding of HOG descripter , OpenCV and image classification,Literature review to study past approaches and deciding model frameworks to be used ,Hyperparameter tuning , coding of data visualization ,creating labelled dataset ,implementing machine learning model framework,Analysing final result and drawing insights,Final report compilation from individual members draft report,Video making. Kshitij Kushwaha contributed in : Studying Image classification , Feature descripters(HOG), Optimizing parameters of HOG , coded pipeline design to perform testing, testing on real world traffic videos-finding suitable videos and performing prediction. ,report making on HOG and feature selection , Video making . Bikram Majumdar worked on : Data collection , help in report making , helped in video making.

References

[1] X. Li and X. Guo, "A hog feature and svm based method for forward vehicle detection with single camera," in *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 1. IEEE, 2013, pp. 263–266.

[2] S. Mallick. Histogram of oriented gradients explained using opencv. 06/12/2016. [Online]. Available: https://learnopencv.com/histogram-of-oriented-gradients

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[4] H. Wang and H. Zhang, "A hybrid method of vehicle detection based on computer vision for intelligent transportation system," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 9, no. 6, pp. 105–118, 2014.

[5] A. Arunmozhi and J. Park, "Comparison of hog, lbp and haar-like features for on-road vehicle detection," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*. IEEE, 2018, pp. 0362–0367.

[6] S. Bougharriou, F. Hamdaoui, and A. Mtibaa, "Linear svm classifier based hog car detection," in *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*. IEEE, 2017, pp. 241–245.

[7] M. N. J. Arróspide, L. Salgado. Video analysis based vehicle detection and tracking using an mcmc sampling framework. 2012. [Online]. Available: http://www.gti.ssr.upm.es/data/Vehicle_database.html

[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[9] A. R. Udacity, "Udacity self-driving car dataset 2-1." [Online]. Available: https://github.com/udacity/self-driving-car

[10] N. Patil. Driving in mumbai,india (bandra-worli sea link). 2020. [Online]. Available: https://www.youtube.com/watch?v=fGRlf5q_rg4&list=WL&index=2&t=186s

[11] D. C. O. Australia. Dash cam owners australia april 19 2019. 19/04/2019. [Online]. Available: https://www.youtube.com/watch?v=v4qIbFDgFvg&t=351s

[12] Histogram of oriented gradients. 18/05/2021. [Online]. Available: https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog