ON ROAD VEHICLE DETECTION USING *Classical Machine Learning* TECHNIQUES

A PREPRINT

Pratyush Kumar
Department of Computer Science
Ashoka University
Sonipat, Haryana
pratyush.kumar_ug21@ashoka.edu.in

Vedansh Priyadarshi
Department of Computer Science
Ashoka University
Sonipat, Haryana
vedansh.priyadarshi_ug21@ashoka.edu.in

Shekhar Chaterjee
Department of Computer Science
Ashoka University
shekhar.chaterjee_ug20@ashoka.edu.in

December 1, 2019

ABSTRACT

On road vehicle detection has been an interesting and challenging area of research in Machine Learning and Computer Vision. Over the years, the field of machine learning has grown from using conventional techniques to using modern Deep Learning techniques like Recurrent Neural Network[1] and semantic image segmentation[2] techniques. Approaches described in papers like YOLO [3], are the current state-of-the-art in object recognition. In this project, we use Histogram of Gradients(HOG)[4] and run experiments to explore the nuances and features this method uses to differentiate between objects. We apply this to detecting cars and train four models namely Multi-Layers Perceptron[5], Decision Trees[6], Naive Bayes[7] and Support Vector Machines[8]. To test the model, we draw boxes in frames where ever a vehicle is detected and qualitatively decide if a particular model is performing well.

***K*eywords** HOG

1 Related Work

There are extensive literature on object detection, but here we mention just a few relevant papers on vehicle detection. Ding et al.[9] presented an accurate approach to estimate vehicle pose using CNNs by extracting vehicles' semantic keypoints. Satar et al.[10] proposed to combine an SSD (Single Shot Multibox Detector) model with a CNN (Convolutional Neural Network) model to train on the database. There are lots of work that has been done on vehicle re-identification by using deep learning techniques. Alfasly et al.[11] presented two-fold framework by using variational feature learning and long short-term memory (LSTM) to learn the relationship among different viewpoints of a vehicle. Oliveira et al.[12] proposed Two-Stream Siamese Neural Network for vehicle re-identification. Sheeny et al.[13] proposed a method of vehicle detection using polarised long wave infrared sensors and Faster-RCNN. Hu et al.[14] presented state-of-the-art SINet, which is a scale-insensitive Convolutional Neural Network (CNN) for fast vehicle detection. Moghimi er al.[15] presented a method for moving vehicle detection using AdaBoost and Haar-Like features in surveillance videos. Recently, Fu et al.[16] presented low-cost LIDAR based vehicle pose estimation and tracking.

2 Approach

In our approach to solving this problem, we started with a statement that in order to indentify vehicles, our model must be capable of differentiating vehicles from other non-vehicle objects present on road. In order to obtain such a model, a

dataset from GIT[17] and KITI[18] consisting of vehicle and non-vehicle images was used. The images in the dataset are of size 64×64. Below is the visualization of a random sample from the dataset:

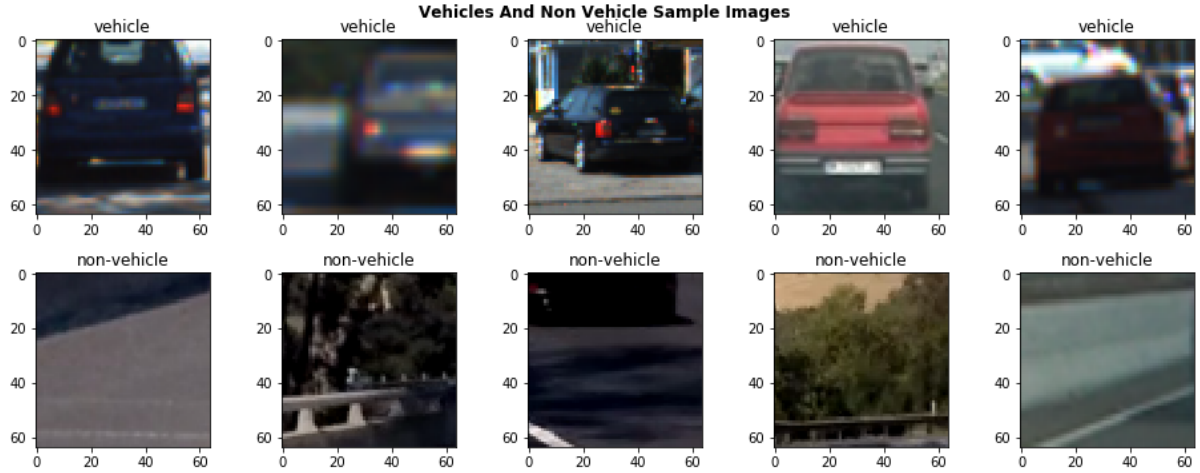


Figure 1: Dataset Visualization

Images contain information about the color, orientation among other assortment of data and the question arises as to how can the best features be extracted from the images so that our models are able to best learn to classify between vehicles and non-vehicles. A viable approach will be to extract features that contain information about the shape and size of the car. For this we use the Histogram of Gradients (HOG) [4] algorithm.

2.1 Histogram of Gradients

Calculating the histogram of gradients of an image involves the following steps:
1.) Normalization and equalization of whole image. This is performed by either either computing the square root or the log of each color channel[4]. This step helps reduce the effects from local shadowing and illumination variations.
2.) Calculate the gradients along **x** and **y** axis: This is done by first filtering the image with the help of two kernels shown in Fig. 2

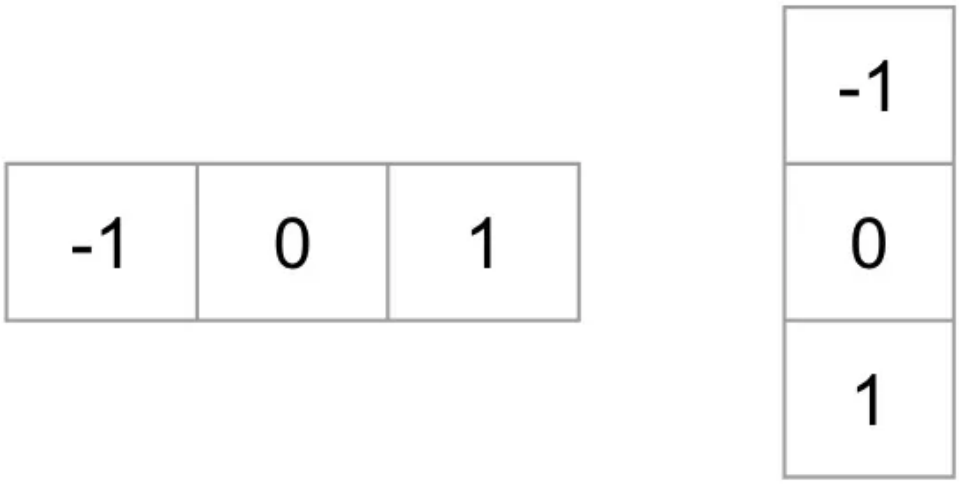
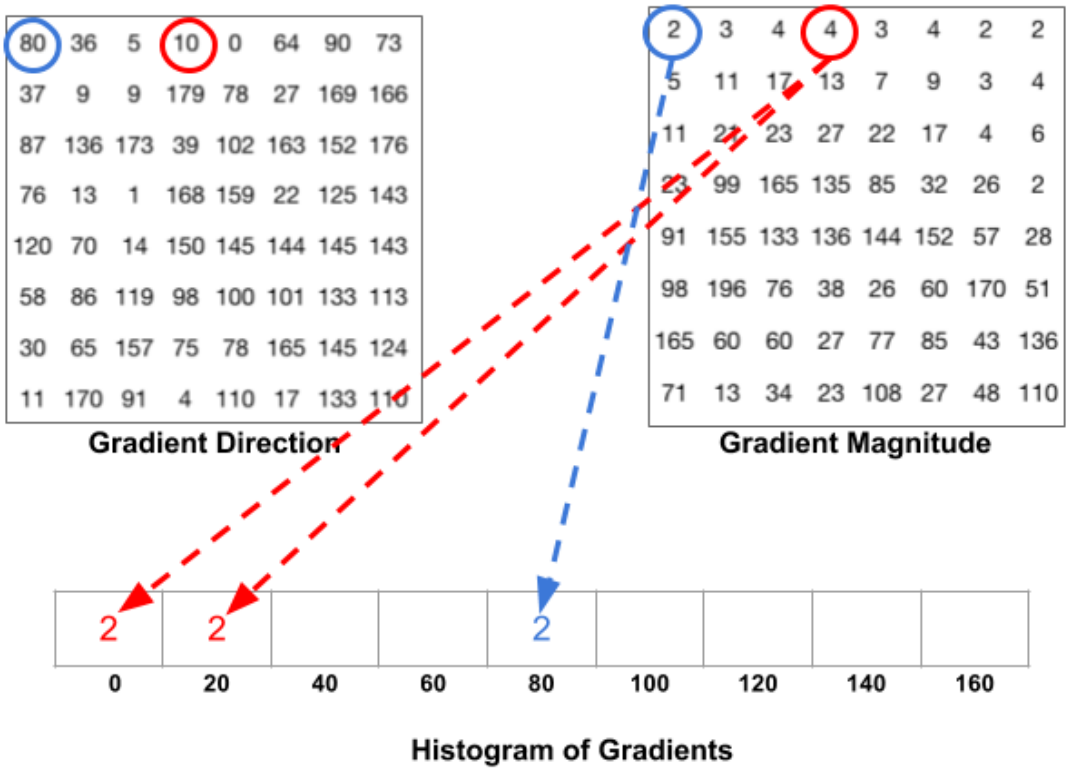


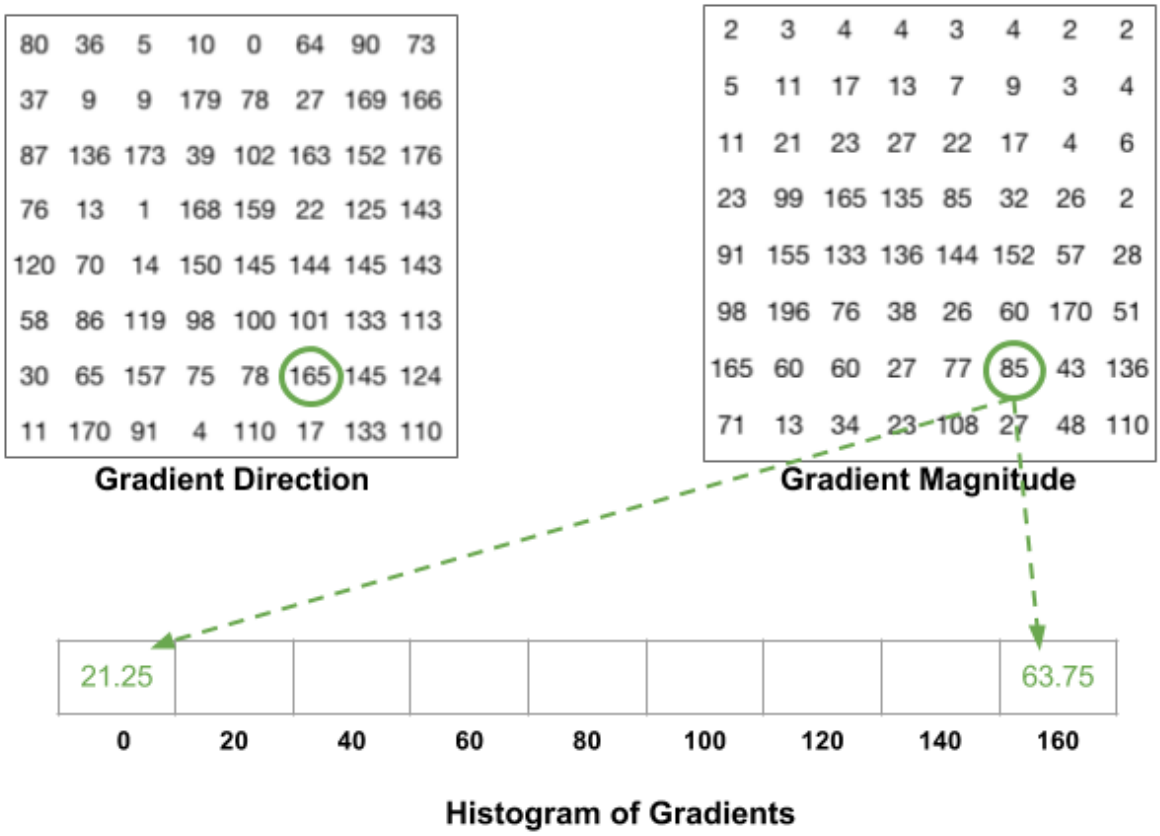
Figure 2: Sobel operator filter[19]

This technically employs the Sobel operator[20] to detect edges. Following this, the magnitude and direction of gradient is calculated using the formulas $Gradient = \sqrt{g_x^2 + g_y^2}$ $\theta = arctan\frac{g_y}{g_x}$ The output is an image that highlights all the edges of an object and removes a lot of non-essential information like constant colored background.

3.)Next, this algorithm calculates gradients across a patch of image of size $height \times width$ This is called the **cell-size** The cell-size can be of any shape but using a small cell-size is recommended as it draws out a compact representation and is thus robust to noise. Consider a cell-size of 8×8 which will have $8 \times 8 \times 2 = 128$ numbers represented as an array of 9 numbers (in case of unsigned gradients) and 18 numbers (in case of signed gradients). Here, w multiply 8×8 with 2 as we will have two 8×8 matrices one representing gradients direction and the gradient magnitude.

4.) Make the histograms: The way we make histograms is as follows: For unsigned gradients, the nine element array is used where each element represents the division of angles from $0 - 180$ degrees as : $0, 20, 40, ..., 180$. The image[19] shown below explains this better:





The gradient direction decides, what cell of the 9 available cells does the gradient magnitude goes into. If the direction value is equally placed between two cell values, the magnitude equally divides between the two cell. Another case is visualized above[19]:

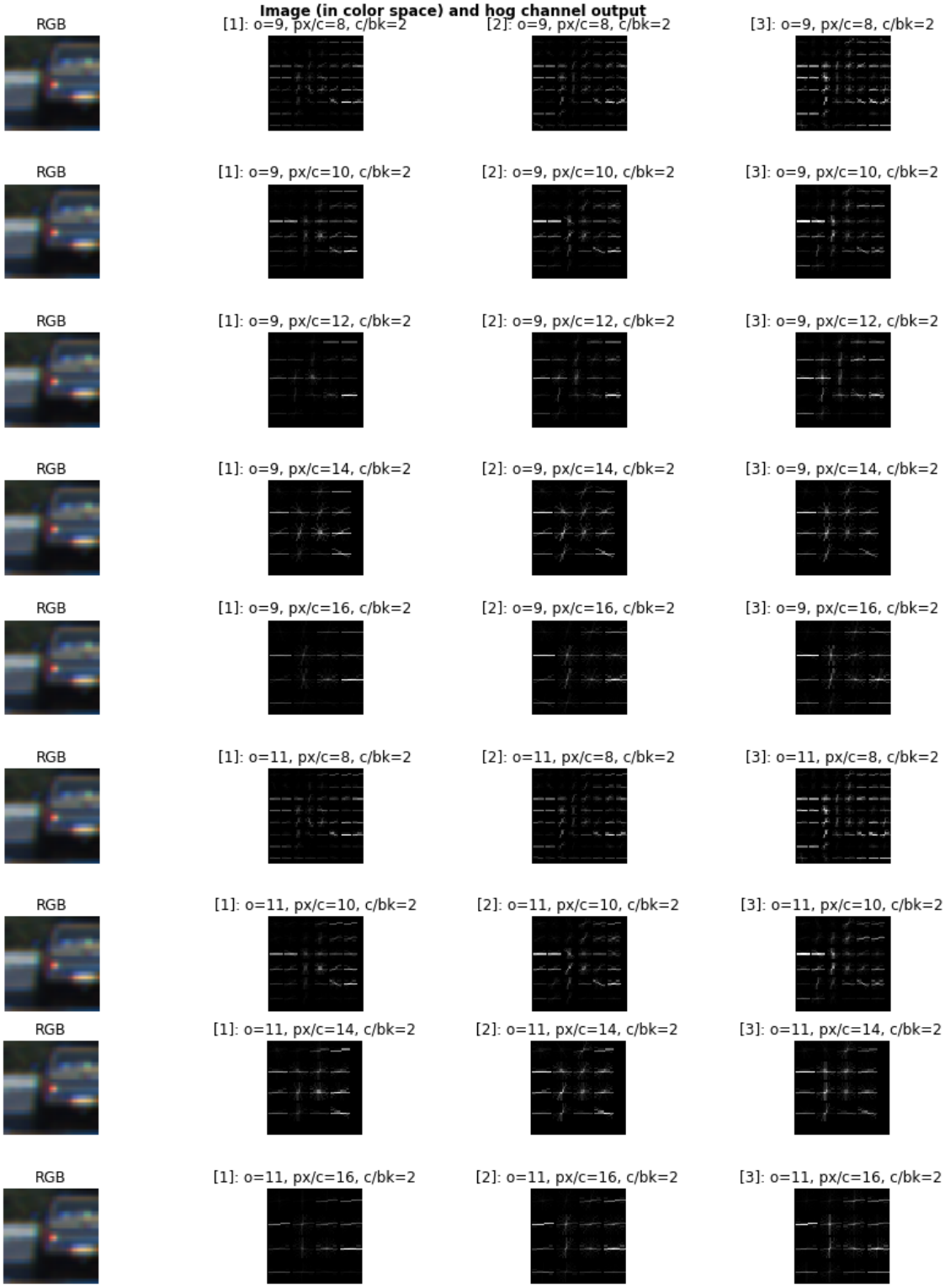
If the direction is not mid-way between two values, the value is proportionally divided between the two cells.

5.) Block Normalization: Finally, we take a block size larger than the cell-size say 16×16 for a cell-size of 8×8. We are just keeping the block-size larger than the initial size as this was the general practice followed in other implementations of the code we studied. In this method, we calculate the length of the histogram vector for each cell and place them one after the other horizontally to get a feature vector.

2.2 Data Analysis

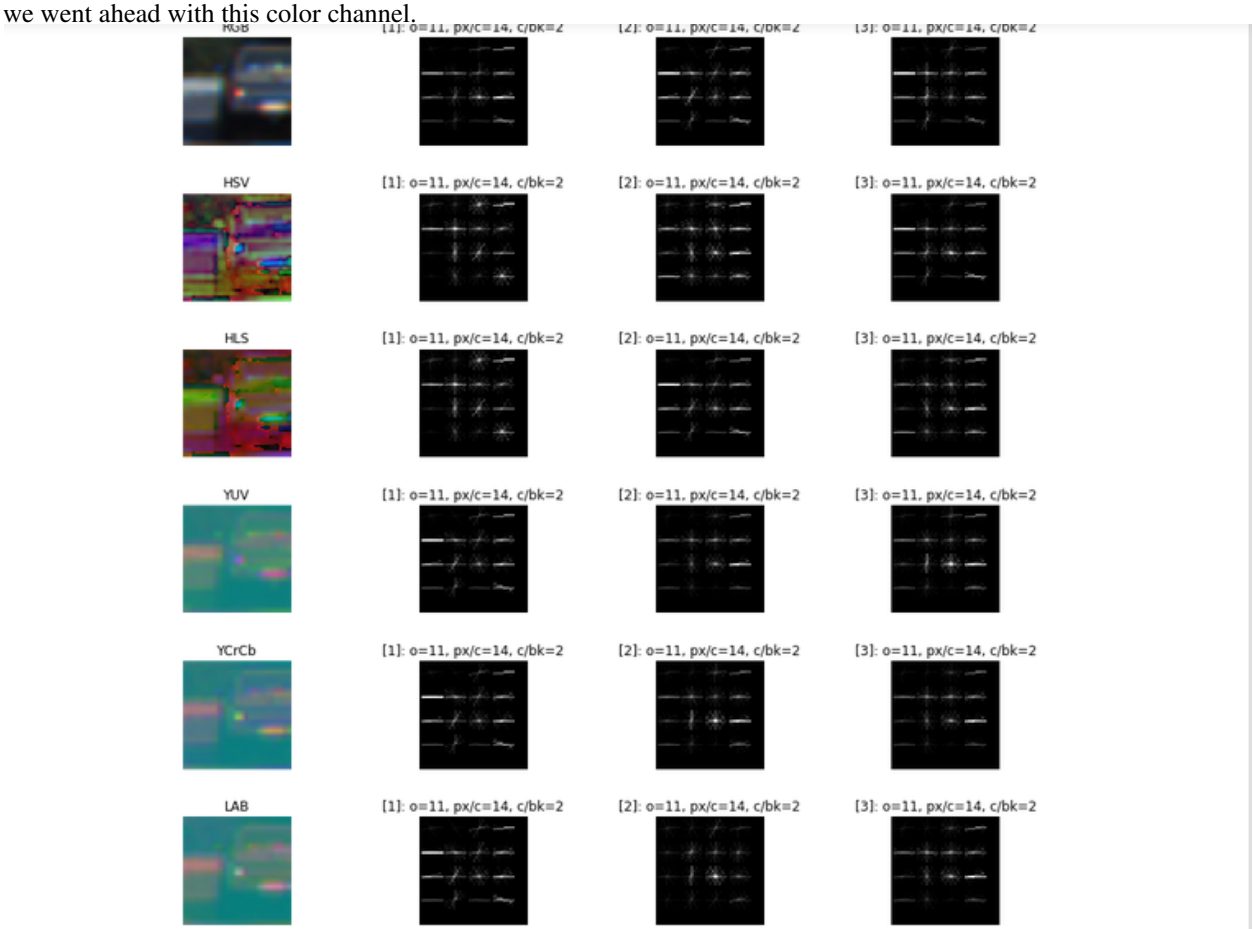
In order to figure out the best hyper-parameters for calculating the HOG, we try out a couple most commonly used values for cell-size/ pixels per cell, block size, scale. Upon experimentation, we concluded that pixel-per-size of 14

with block size of 2 worked best for drawing out the best features.



In our approach, we calculated HOG along all three channels of the image separately in order to check if we can draw a mixture of features that might help improve the training process. We observed that **RGB** gave similar features along all three channels. We tried other color channels like HSV, YUV, HLS, YCrCb, LAB. While other rest of the color

channels were firing for arbitrary features, **YCrCb** appeared to give varying, useful features across multiple images. So, we went ahead with this color channel.



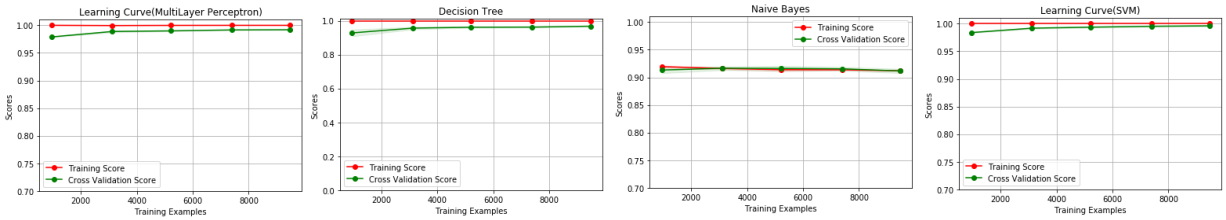
We train our models on these features drawn on these parameters. While testing it out on provided test videos[21], we used a strategy to take a 32×32 patch across each frame, drawing 10 frames per sample. In order to check if the model is detecting vehicles, we implemented a method that saves the top-left and bottom-right co-ordinates of the patches in a frame and draws a rectangle around it. Methods were implemented to smooth the bounding boxes based on the centroid of the boxes so that the output in videos is a clean box. To get a clearer picture of what the model is doing, a method to visualize heap-map in areas where cars are detected by the model was also implemented.

3 Results

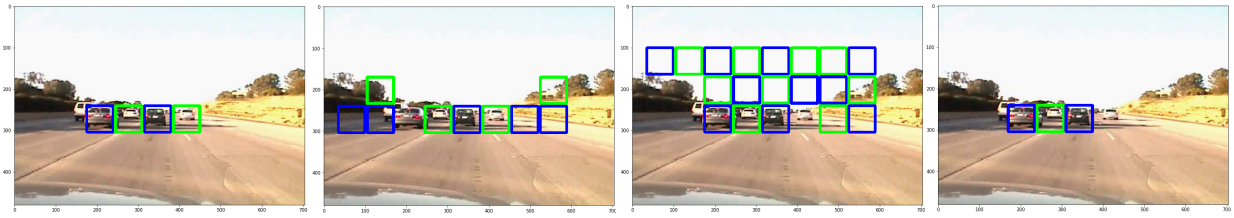
We trained out data set of 8792 vehicles and 8968 non-vehicles where a shuffled dataset of 14208 data points made the training set and the rest were used for testing the model. Below is the accuracy of the model on both training and testing dataset:

Classifier/Accuracy	Training	Test
MLP	1.0	0.9944
Decision-Trees	1.0	0.9718
Naive Bayes	0.92	0.9082
SVM	1.0	0.9957

The training process looked like:



When tested on frames from the test videos, below are the results for different models:



All the above images follow the order: **MLP, Decision Trees, Naive Bayes, SVM** The github repository with all code, files and results is: `https://github.com/Cloaked04/IML_Final_Project`

4 Conclusion & Future Work

Detecting vehicles using HOG feature extraction technique gives descent results on the test videos. But the method involves hard-coding multiple steps that is a bottleneck in generalizing this technique. For instance, in order to obtain the right classification, we had to take the frame in the range of $100 - 200$ along the height so as to not include the sign board. Not doing so results in the mis-classification of the sign board as it resembles the number plates at the back of a car. The quality of video also plays a role in prediction, better pixel density allows for more data to be captured in the frames allowing for better features. The test video was simple-definition video with a lower pixel density.
Modern techniques in Deep Learning like **YOLO**, **Faster-RCNN** and others have given much better results on object detection tasks. These have been applied to Vehicle detection tasks and have given good results. In the future, we plan on implementing some state of the art models and combining them with segmentation[2] techniques and experiment with the features in hope of improving our results.

References

[1] Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *CoRR*, abs/1808.03314, 2018.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs, 2014.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015.

[4] M. Kachouane, S. Sahki, M. Lakrouf, and N. Ouadah. Hog based fast human detection. 2015.

[5] Cyrine Arouri, Engelbert Mephu Nguifo, Sabeur Aridhi, Cécile Roucelle, Gaelle Bonnet-Loosli, and Norbert Tsopzé. Towards a constructive multilayer perceptron for regression task using non-parametric clustering. a case study of photo-z redshift reconstruction, 2014.

[6] Nenad Markuš, Miroslav Frljak, Igor S. Pandžić, Jörgen Ahlberg, and Robert Forchheimer. Object detection with pixel intensity comparisons organized in decision trees, 2013.

[7] Dilip K. Prasad. Object detection in real images, 2013.

[8] Lei Zhang and David Zhang. Svm and elm: Who wins? object recognition with deep convolutional features from imagenet, 2015.

[9] Wenhao Ding, Shuaijun Li, Guilin Zhang, Xiangyu Lei, and Huihuan Qian. Vehicle pose and shape estimation through multiple monocular vision, 2018.

[10] Burak Satar and Ahmet Emir Dirik. Deep learning based vehicle make-model classification. 2018.

[11] Saghir Ahmed Saghir Alfasly, Yongjian Hu, Tiancai Liang, Xiaofeng Jin, Qingli Zhao, and Beibei Liu. Variational representation learning for vehicle re-identification. 2019.

[12] Icaro O. de Oliveira, Rayson Laroca, David Menotti, Keiko V. O. Fonseca, and Rodrigo Minetto. Vehicle re-identification: exploring feature fusion using multi-stream convolutional networks, 2019.

[13] Marcel Sheeny, Andrew Wallace, Mehryar Emambakhsh, Sen Wang, and Barry Connor. Pol-lwir vehicle detection: Convolutional neural networks meet polarised infrared sensors, 2018.

[14] Xiaowei Hu, Xuemiao Xu, Yongjie Xiao, Hao Chen, Shengfeng He, Jing Qin, and Pheng-Ann Heng. Sinet: A scale-insensitive convolutional neural network for fast vehicle detection. 2018.

[15] Mohammad Mahdi Moghimi, Maryam Nayeri, Majid Pourahmadi, and Mohammad Kazem Moghimi. Moving vehicle detection using adaboost and haar-like feature in surveillance videos. 2018.

[16] Chen Fu, Chiyu Dong, Xiao Zhang, and John M. Dolan. Low-cost lidar based vehicle pose estimation and tracking, 2019.

[17] Jon Arróspide, Luis Salgado, and Marcos Nieto. Video analysis-based vehicle detection and tracking using an MCMC sampling framework. *EURASIP Journal on Advances in Signal Processing*, 2012(1), January 2012.

[18] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. 2013.

[19] Satya Mallick. Home, Dec 2016.

[20] Victor Bogdan, Cosmin Bonchiş, and Ciprian Orhei. Custom extended sobel filters, 2019.

[21] Sayanan Sivaraman and Mohan Manubhai Trivedi. A general active-learning framework for on-road vehicle recognition and tracking. *Trans. Intell. Transport. Sys.*, 11(2):267–276, June 2010.