Pattern Recognition

Car Detection

Group Details:

K. Niveda - S20190020224

S. Anuhya - S20190020251

R. Varshini - S20190020245

Introduction:

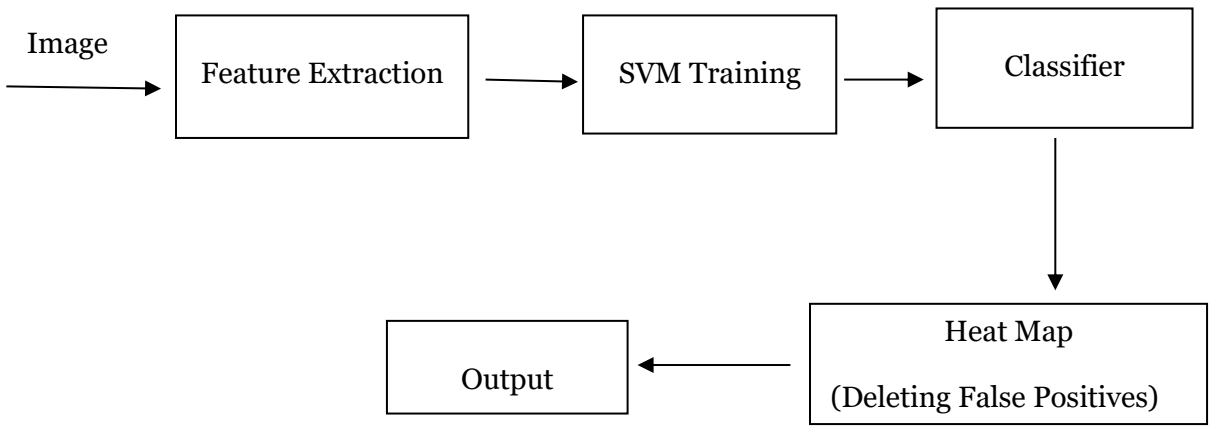
Vehicle detection is very important for automotive safety driver assistance system. This project is focused on improving the performance of vehicle detection system with single camera and proposed a HOG feature and SVM classifier for vehicle detection. The HOG feature basically uses the gradient information of objects. Because the typical appearance of vehicle has clear edges in vertical and horizontal directions, the HOG can be a good candidate as a visual feature of vehicle. And SVM is an effective scheme that is widely used for the classification of various objects. We can expect much better accuracy as well as robustness to wild environment by using this combination.

Dataset:

The labeled data come from a combination of the GTI vehicle image database and KTTTI vision benchmark suite. The non-vehicle data also contain some images extracted from a real dash cam video with hard negative mining to reduce the number of false positives. This is the summary of the data

No. of vehicle images=8792
No. of non-vehicle images=8968
Dimensions=64*64

Methodology:



Feature Extraction: We are using features of an image to train the data. Features of an image play a major role in the project. We tried various combinations of parameters using the grid search and found that HOG channels encode most of the required features. The feature descriptor that is used for this project is HOG.

Histogram of Oriented Gradients (HOG): HOG is a feature descriptor which is used to characterize objects on the basis of their shapes. HOG technique calculates histogram (occurrences) of each gradient orientation in the mentioned part of image. Below is the step by step process of feature extraction which includes HOG also:

- **Preprocess the Data:** The images are read and bring down the width to height ratio to 1:2 (aspect ratio). The image size should preferably be 64x128 pixels. This is because we will be dividing the image into 8*8 and 16*16 patches to extract the features. Having the specified size (64 x 128) will make all our calculations simple.
- **Calculating Horizontal and Vertical Gradients:** The next step is to calculate the gradient for every pixel in the image. Gradients are the small change in the x and y directions. Let's take an example matrix and calculate:

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

The resultant gradients in the X and Y direction for this pixel are:

- Change in X direction (G_x) = 89-78 = 11
- Change in Y direction (G_y) = 68-56 = 8

We have calculated the gradients in both x and y direction separately. The same process is repeated for all the pixels in the

image. The next step would be to find the magnitude and orientation using these values.

- **Calculate the Magnitude and Orientation:** Using the gradients, we calculated in the last step, we will now determine the magnitude and direction for each pixel value.

  Total Gradient Magnitude = $\sqrt{[(G_x)^2+(G_y)^2]}$

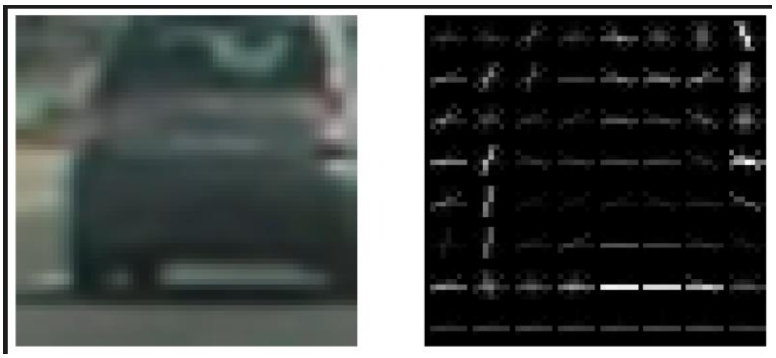
  Gradient Direction (Φ) = arctan (G_y / G_x)

- **Calculate HOG in 8*8 cells (9×1):** Using If we divide the image into 8×8 cells and generate the histograms, we will get a 9 x 1 matrix for each cell. Using the frequency, we can use the gradient magnitude to fill the values in the matrix.
- **Normalize gradients in 16×16 cell (36×1):** Here, we will be combining four 8×8 cells to create a 16×16 block. And we already know that each 8×8 cell has a 9×1 matrix for a histogram. So, we would have four 9×1 matrices or a single 36×1 matrix. To normalize this matrix, we will divide each of these values by the square root of the sum of squares of the values.
- **HOG features:** The parameters that we chosen are

Parameter	Value
Color space	YCrCb
Orientations	9
Pixel per cell	8
Cells per block	(2,2)
No. of channels	3

Total no. of features for our image are [7x7 block positions] x [2x2 cells per block] x [9 orientations] x [3 channels] =5292

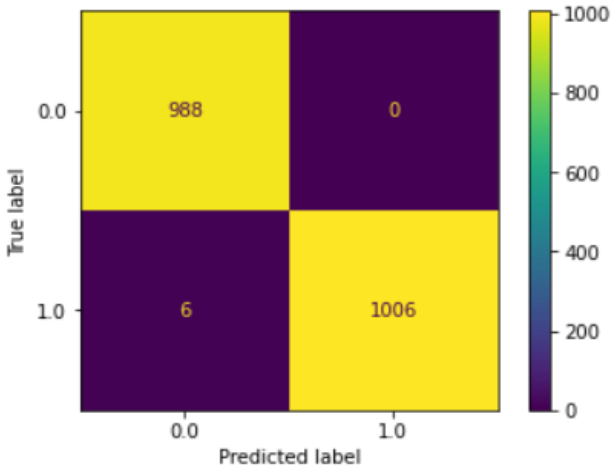
204.95 Seconds to extract features... Below image is an visualization of hog features



Linear SVM: Before training as a safety measure, we use a scaler to transform the raw features before feeding them to our classifier for training or predicting, reducing the chance wrong prediction. We divide the dataset into training (80%) and testing (20%) with a random state of a number between 0 to 100.

```
8.99 Seconds to train SVC...
Test Accuracy of SVC = 0.9985
```

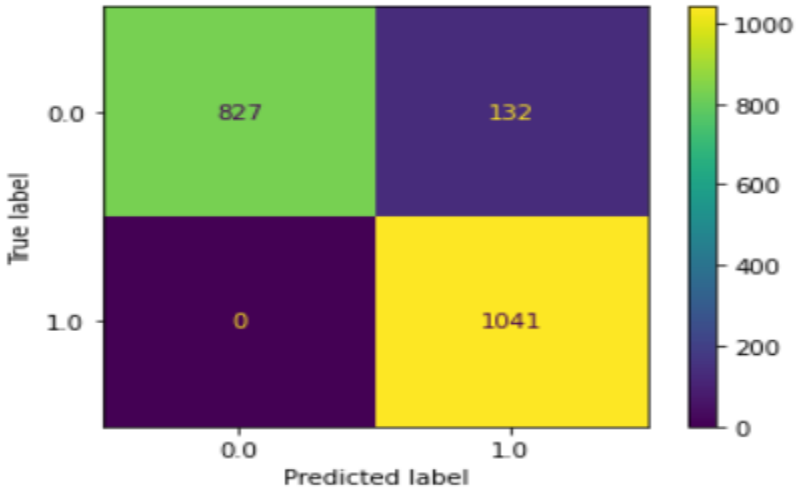
Confusion Matrix



KNN:

```
0.13 Seconds to train Knn...
Test Accuracy of Knn = 0.934
```

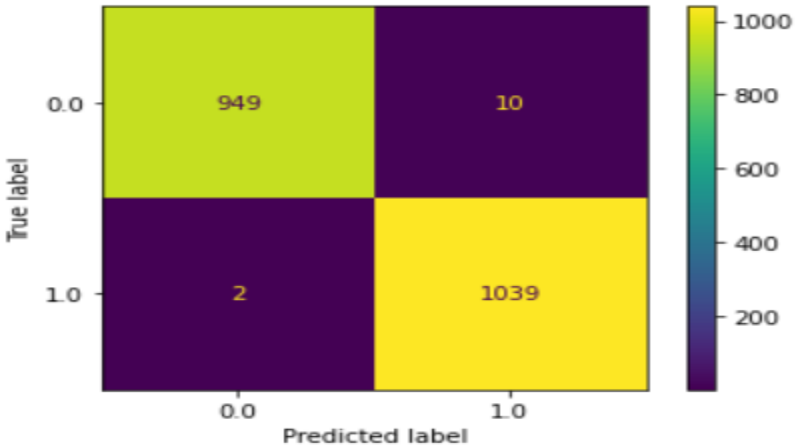
Confusion Matrix



Quadratic SVM:

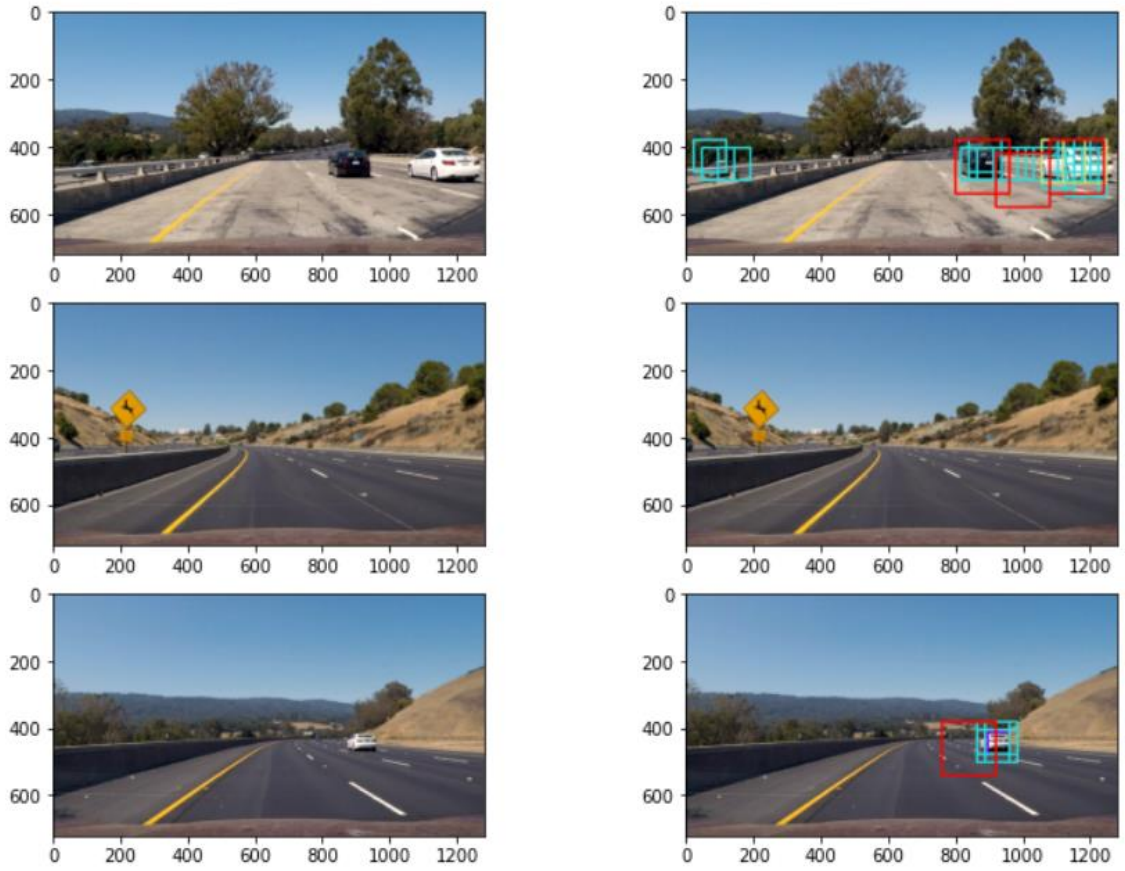
```
233.54 Seconds to train Poly svc...

Test Accuracy of Quadratic SVC = 0.994
```

Confusion Matrix



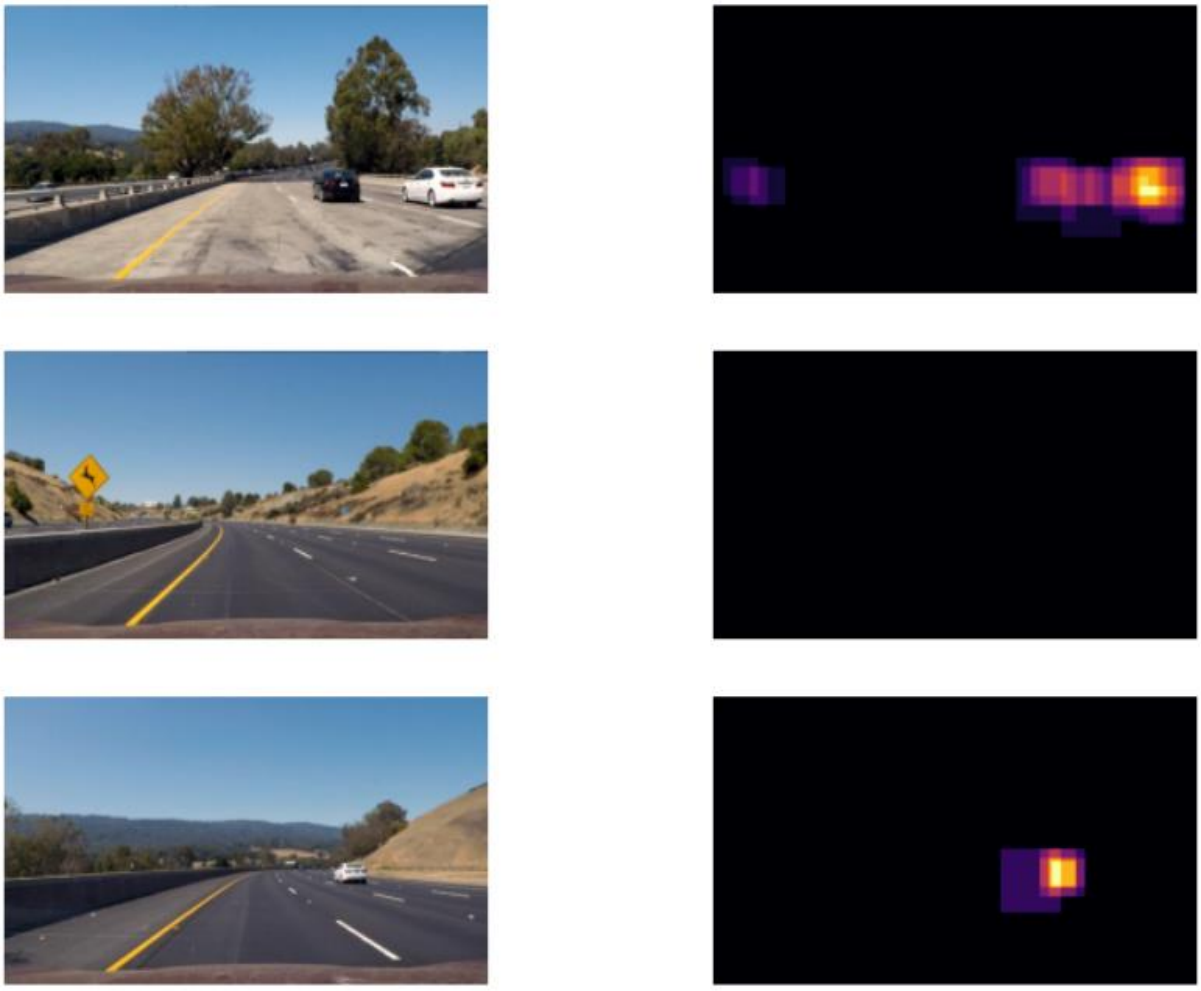
Sliding Window Search: To decrease the computation time, we implemented a sliding window approach in order to get windows at different scales for purpose of vehicle detection. After detecting we drew those windows onto an image to confirm that they were being created correctly. We used 4 different scales for 4 different zones

```
y_range = {1.0 : (380,508),
           1.5 : (380,572),
           2.0 : (380,636),
           2.5 : (380,700)}
```

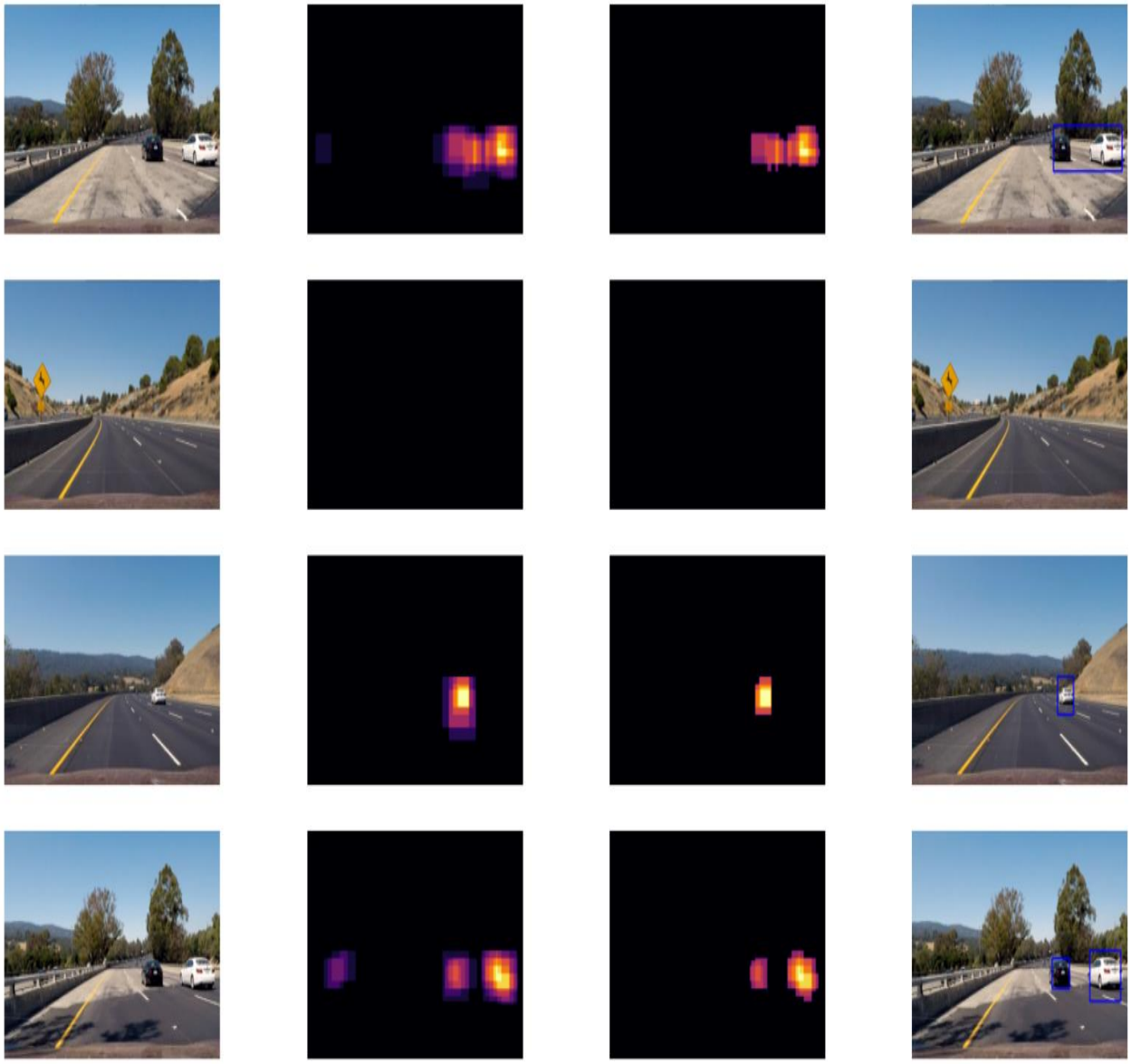


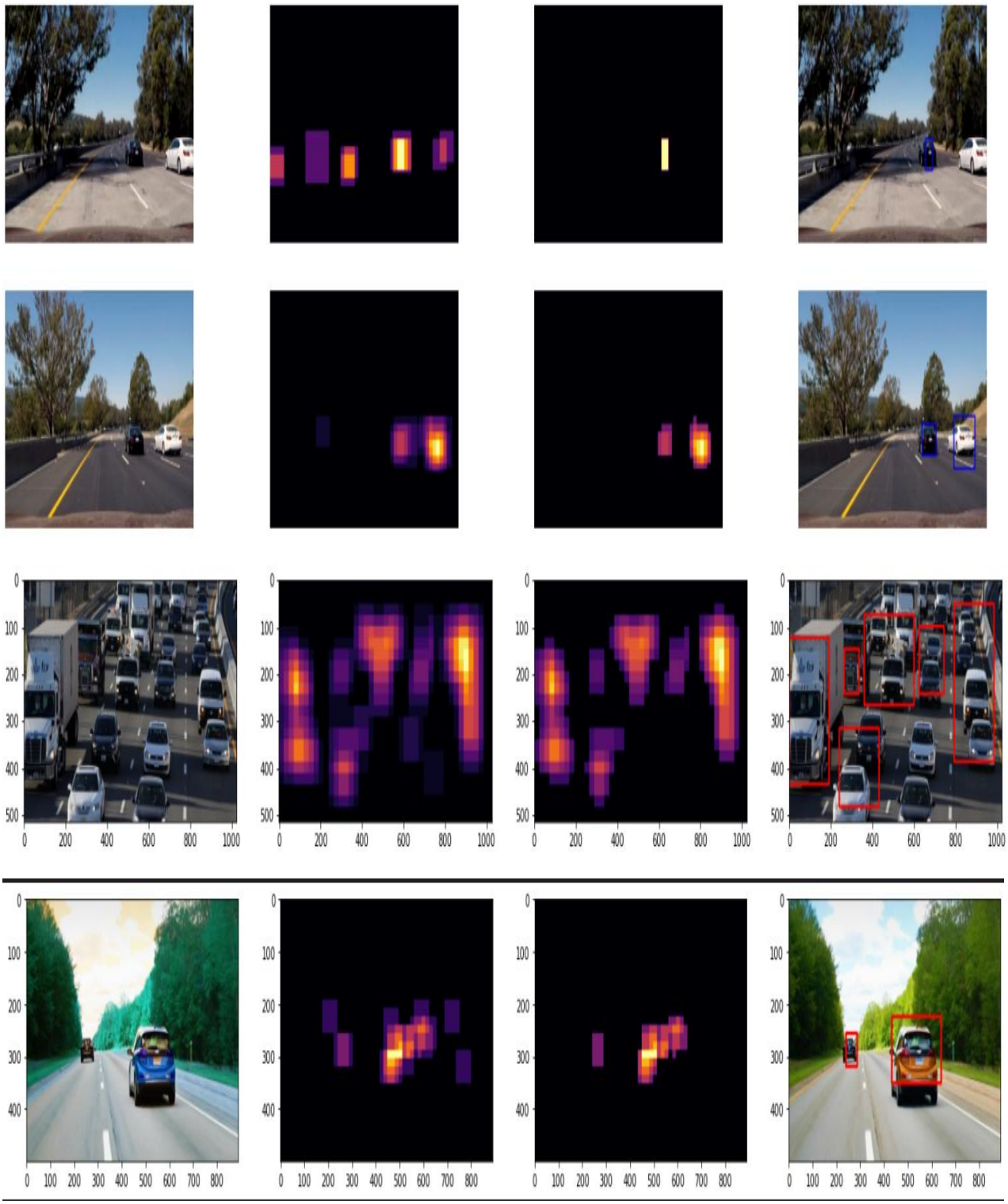
Heat Map: Consecutive box plot notice that there are overlapping detections and false positive detections are spaced out.Here, we start with a blank grid and add heat for all pixels within windows where the

+ve detections are reported by the classifier. The hotter the parts, the more likely it is a true positive.



Results:





Contributions:

Varshini R – Pre-processing the data, deciding the parameters for feature extraction, training using KNN.

Niveda K – Feature Extraction and Visualizing output using sliding window and heatmap.

Anuhya S – Training and testing using Linear SVM, Quadratic SVM and Visualizing output using sliding window and heatmap.

GitHub link:

https://github.com/Anuhya27/Car-detection-PR

References:

https://ieeexplore.ieee.org/document/6643881

https://www.researchgate.net/publication/328253593_HOG-SVM_Car_Detection_on_an_Embedded_GPU

http://www.ijettjournal.org/2017/volume-48/number-6/IJETT-V48P257.pdf

https://medium.com/@mithi/vehicles-tracking-with-hog-and-linear-svm-c9f27eaf521a

https://www.scribd.com/document/424830072/VEHICLE-DETECTION-USING-HOG-AND-SVM