GRADUATE CERTIFICATE INTELLIGENT SENSING SYSTEMS PRACTICE MODULE REPORT

Distracted-Driver-Detection

Institute of Systems Science, National University of Singapore, Singapore 119615

ABSTRACT

Driving a car is a complex task, and it requires complete attention. Distracted driving is any activity that takes away the driver's attention from the road. Many States now have laws against texting, talking on a cell phone, and other distractions while driving. We believe that computer vision can augment the efforts by the governments to prevent accidents caused by distracted driving. Our algorithm automatically detects the distracted activity of the drivers and alerts them. We envision this type of product being embedded in cars to help drivers avoid distraction.

1. INTRODUCTION

According to the CDC motor vehicle safety division, one in five car accidents is caused by a distracted driver. Sadly, this translates to 425,000 people injured and 3,000 people killed by distracted driving every year In this project, I have created and refined machine learning and Deep Learning models to detect what the driver is doing in a car given driver images. This is done by predicting the likelihood of what the driver is doing in each picture.

2. OVERVIEW

In this project, I have created and refined machine learning and Deep Learning models to detect what the driver is doing in a car given driver images. This is done by predicting the likelihood of what the driver is doing in each picture.

Acquired data from Kaggle Competition, data contain driver images with 10 Actions Safe driving, Texting - right, Talking on the phone - right, Texting - left, Talking on the phone - left,Operating the radio, Drinking, Reaching behind, Hair and makeup, Talking to passenger

3. PROBLEM STATEMENT

Given a dataset of 2D dashboard camera images, an algorithm needs to be developed to classify each driver's behaviour and determine if they are driving attentively, wearing their seatbelt, or taking a selfie with their friends in the backseat etc..? This can then be used to automatically detect drivers engaging in distracted behaviours from dashboard cameras.

4. DATASET

Given driver images, each taken in a car with a driver doing something in the car (texting, eating, talking on the phone, makeup, reaching behind, etc).

Our goal is to predict the likelihood of what the driver is doing in each picture

The 10 classes to predict are:

- c0: Safe Driving
- c1: Texting - Right
- c2: Talking on the phone - Right
- c3: Texting - left
- c4: Talking on the phone - Left
- c5: Operating the Radio
- c6: Drinking
- c7: Reaching Behind
- c8: Hair and Makeup
- c9: Talking to Passenger

To ensure that this is a computer vision problem, we have removed metadata such as creation dates. The train and test data are split on the drivers, such that one driver can only appear on either train or test set

The data set contains 22424 driver images in total, Stratified splitting will use to split the data set into 80:10 Training-Testing ratio and The training data set is further split into 90:10 Training-Validation set, and this data set contain 10 classes to predict, In data prepossessing step resize every image into 224*224 and extract feature using computer vision techniques HOG, SIFT, LBP, KAZE, Color histogram, Gray Scale image and normalize data. PCA , LDA for data dimensional reduction techniques

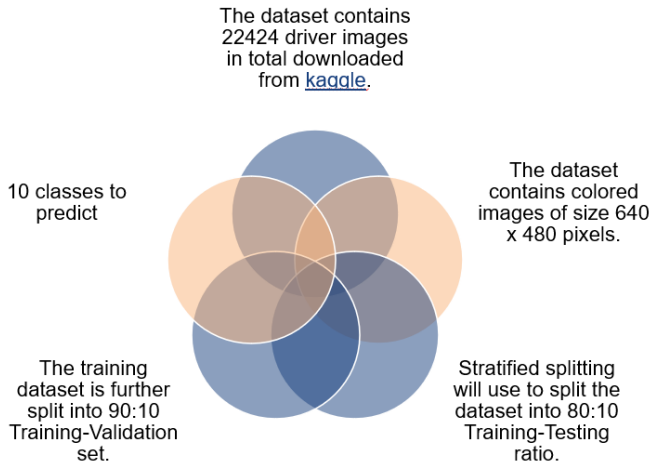


Fig. 1. Dataset.

5. DATA EXPLORATION

The provided data set has driver images, each taken in a car with a driver doing something in the car (texting, eating, talking on the phone, makeup, reaching behind, etc). This dataset is obtained from Kaggle(State Farm Distracted Driver Detection competition).

The 10 classes to predict are:

- c0: Safe Driving
- c1: Texting - Right
- c2: Talking on the phone - Right
- c3: Texting - left
- c4: Talking on the phone - Left
- c5: Operating the Radio
- c6: Drinking
- c7: Reaching Behind
- c8: Hair and Makeup
- c9: Talking to Passenger

There are 102150 total images. Of these 17939 are training images,4485 are validation images and 79726 are training images. All the training, validation images belong to the 10 categories shown above. The images are coloured and have 640 x 480 pixels each as shown below

6. DATA PREPROCESSING

Preprocessing of data is carried out before model is built and training process is executed. Following are the steps carried out during preprocessing.

- Initially the images are divided into training and validation sets.
- The images are resized to a square images i.e. 224 x 224 pixels.
- All three channels were used during training process as these are color images.
- The images are normalised by dividing every pixel in every image by 255.
- To ensure the mean is zero a value of 0.5 is subtracted.

7. IMPLEMENTATION

A standard CNN architecture was initially created and trained. We have created 4 convolutional layers with 4 max pooling layers in between. Filters were increased from 64 to 512 in each of the convolutional layers. Also dropout was used along with flattening layer before using the fully connected layer. Altogether the CNN has 2 fully connected layers. Number of nodes in the last fully connected layer were setup as 10 along with softmax activation function. Relu activation function was used for all other layers.Xavier initialization was used in each of the layers.

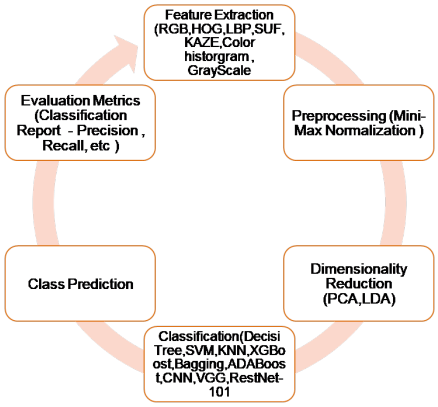
8. REFINEMENT

To get the initial result simple CNN architecture was built and evaluated. This resulted in a decent loss. The public score for the initial simple CNN architecture(initial unoptimized model) was 2.67118. After this to further improve the loss, transfer learning was applied to VGG16 along with investigating 2 types of architectures for fully connected layer. Model Architecture1 showed good results and was improved further by using the below techniques

- Drop out layer was added to account for overfitting.
- Xavier initialization was used instead of random initialization of weights
- Zero mean was ensured by subtracting 0.5 during Preprocessing.
- Training was carried out with 400 epochs and with a batch size of 16
- To further improve the loss metric ,VGG16 along with Model Architecture1 was

- selected and fine-tuning was applied. SGD optimiser was used with very slow
- learning rate of 1e-4. A momentum of 0.9 was applied with SGD

9. PROPOSED SYSTEM



10. EXPERIMENTAL RESULTS

Table 1. The performance comparison.

Approach	Val Score [?]	Test Score [?]
VGG	0.99	0.99
ResNet Strategy 1	0.65	0.63
ResNet Strategy 2	0.99	0.99

11. EVALUATION METRICS AND RESULTS

Machine Learning Algorithms are not fit for this Problem Statement Because Accuracy is very low , High accuracy is 0.77 only ,ResNet and VGG is performing well, Compare to ResNet Model VGG Accuracy is high, but in Real time Videos and images Rest Net Performing Well.

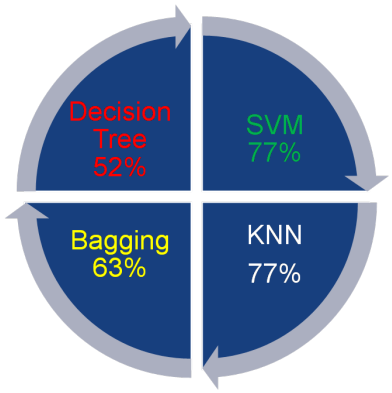


Fig. 2. Machine Learning Algorithms Performance.

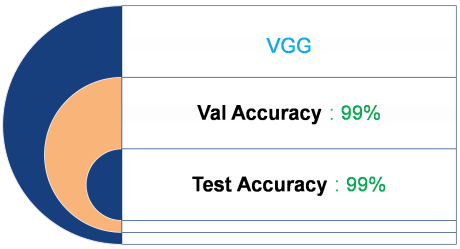


Fig. 3. VGG Performance.

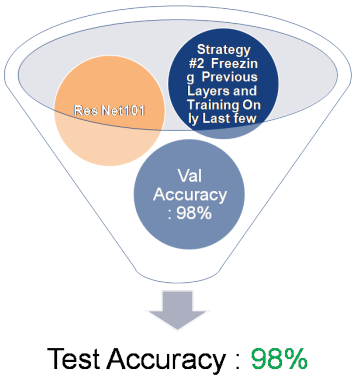


Fig. 4. ResNet101 Performance.

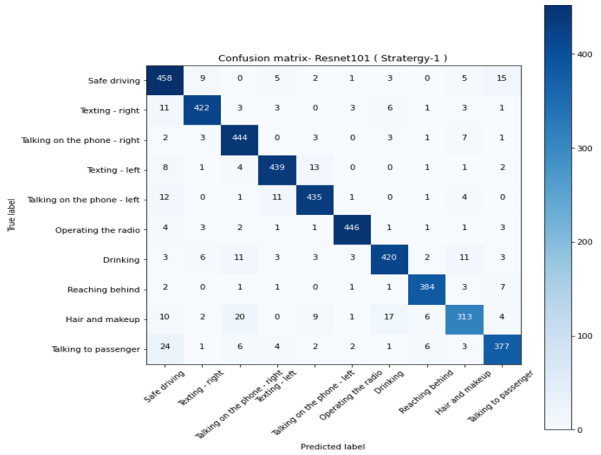


Fig. 5. Res Net 101 – Strategy 1

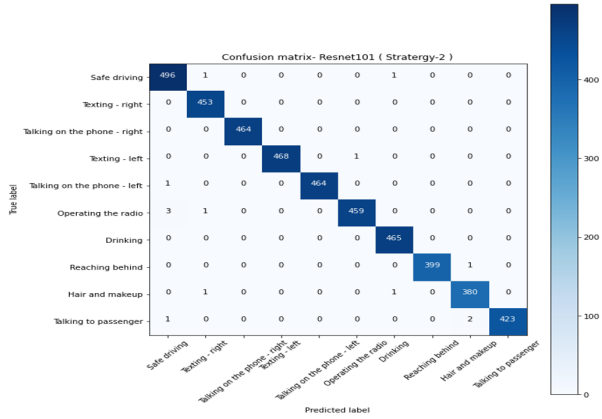


Fig. 6. Res Net 101 – Strategy 2

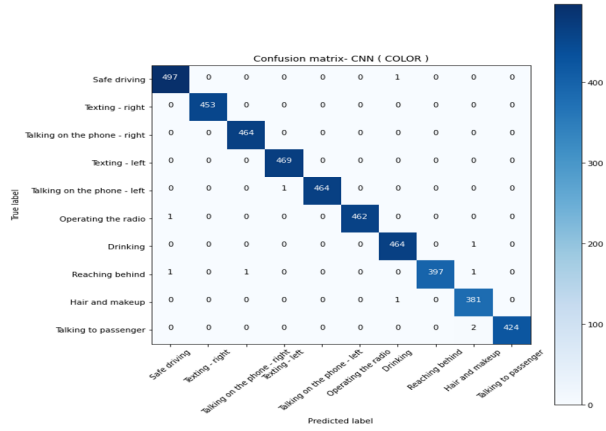


Fig. 7. VGG Result

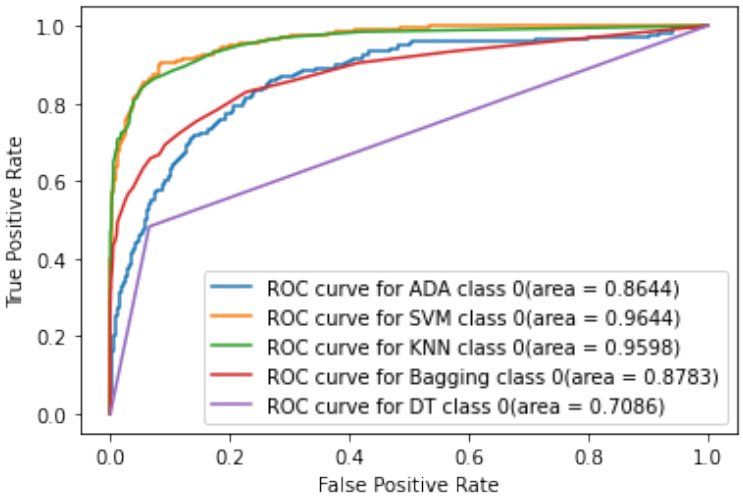


Fig. 8. ROC Curve (LDA - Class 0)

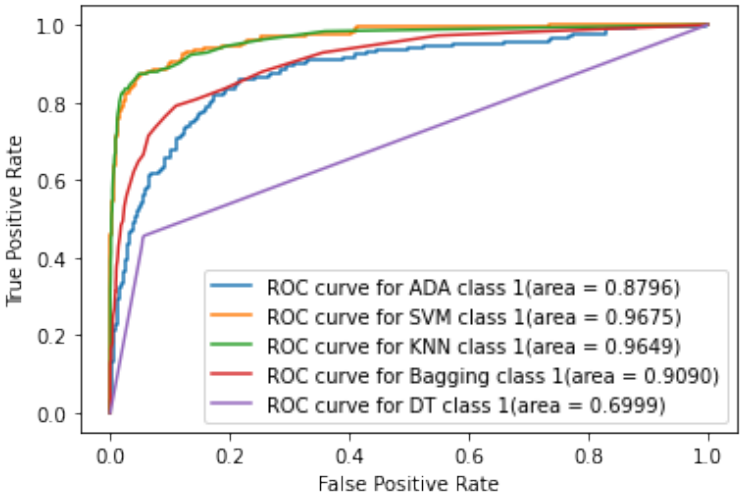


Fig. 9. ROC Curve (LDA - Class 1)

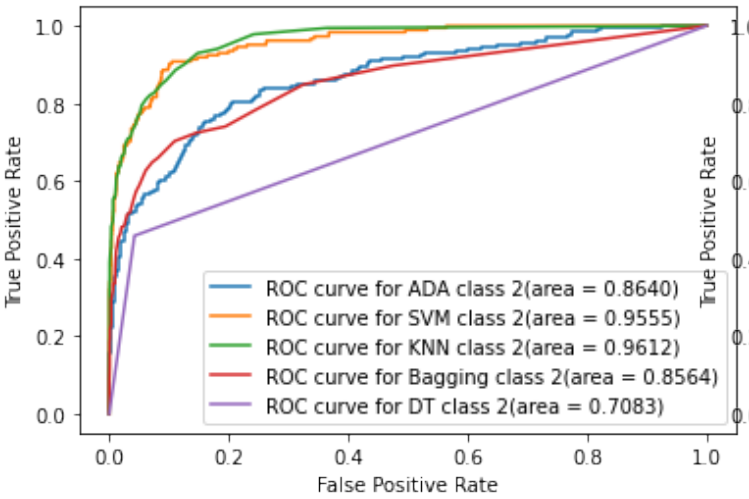


Fig. 10. ROC Curve (LDA - Class 2)

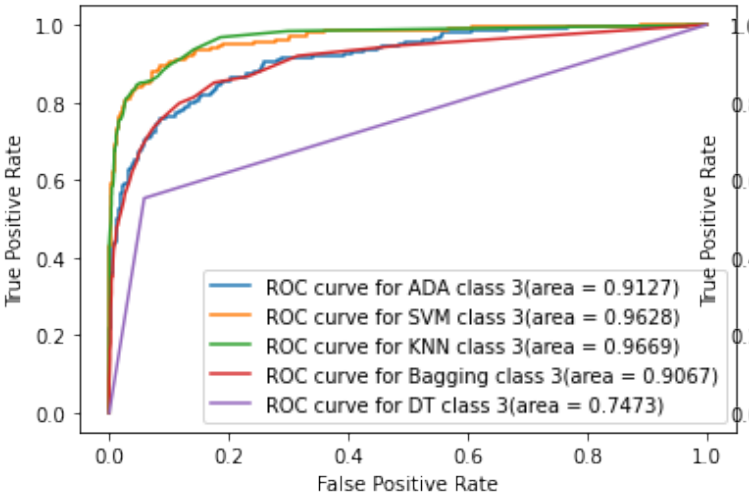


Fig. 11. ROC Curve (LDA - Class 3)

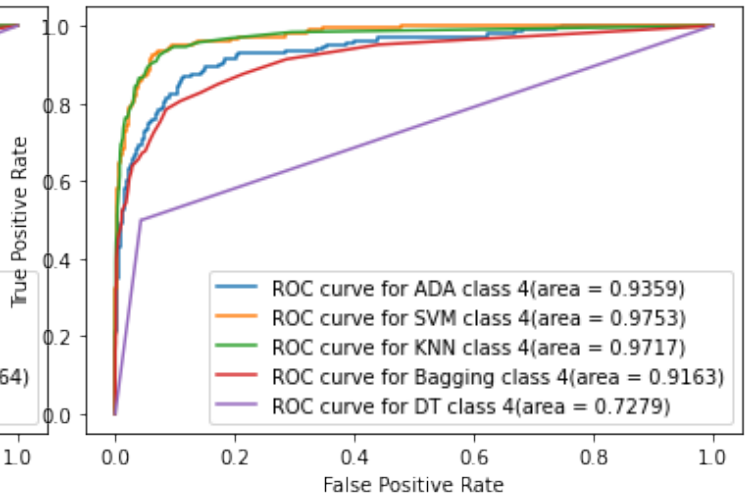


Fig. 12. ROC Curve (LDA - Class 4)

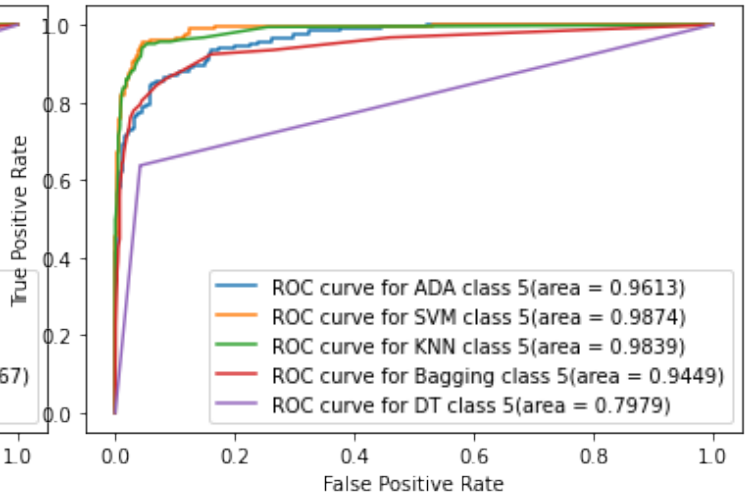


Fig. 13. ROC Curve (LDA - Class 5)

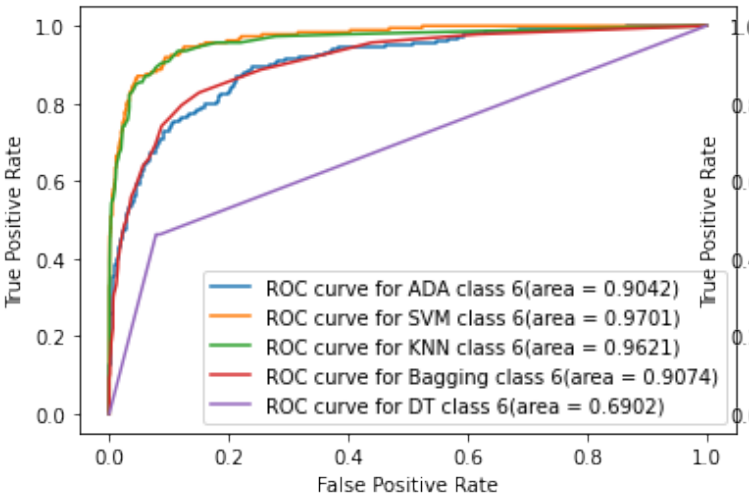


Fig. 14. ROC Curve (LDA - Class 6)

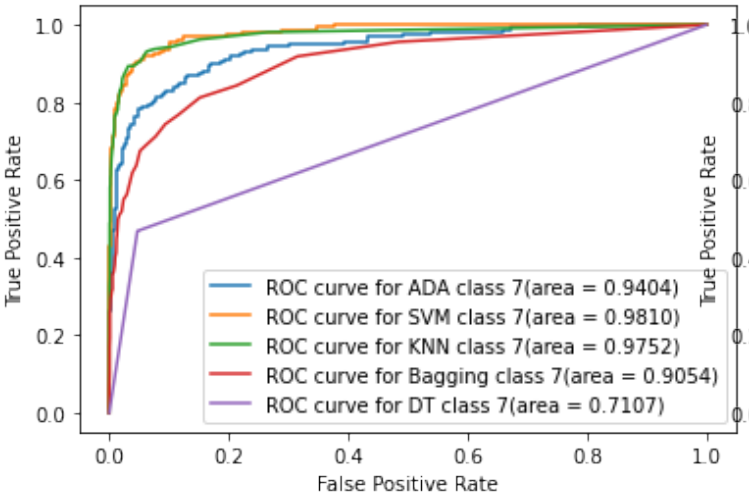


Fig. 15. ROC Curve (LDA - Class 7)

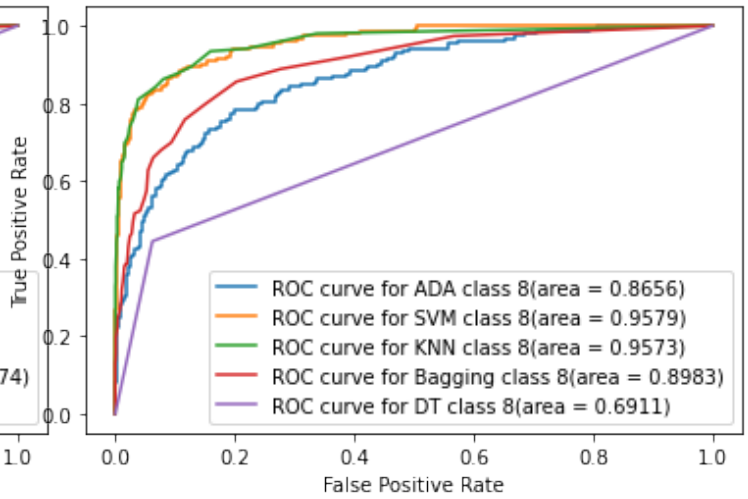


Fig. 16. ROC Curve (LDA - Class 8)

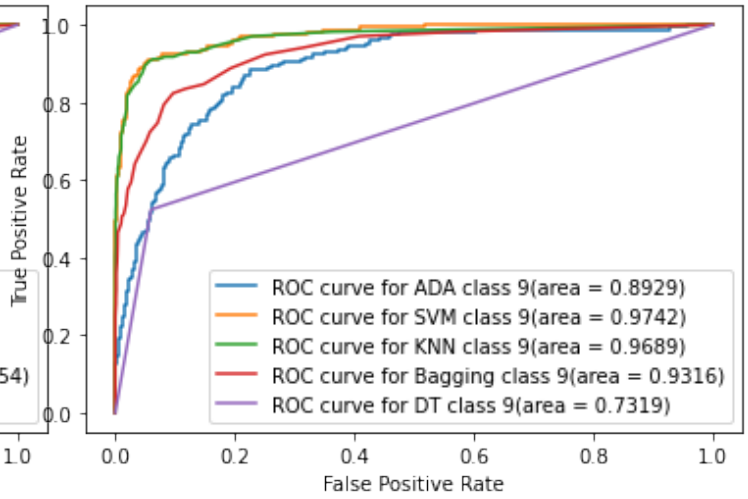


Fig. 17. ROC Curve (LDA - Class 9)

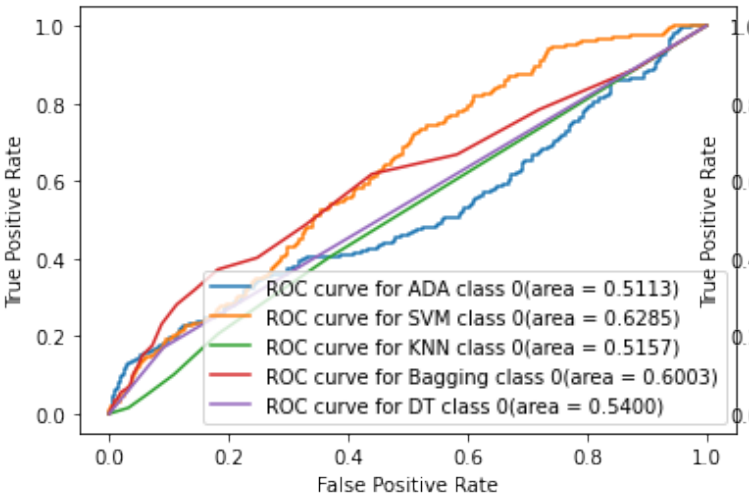


Fig. 18. ROC Curve (PCA - Class 0)

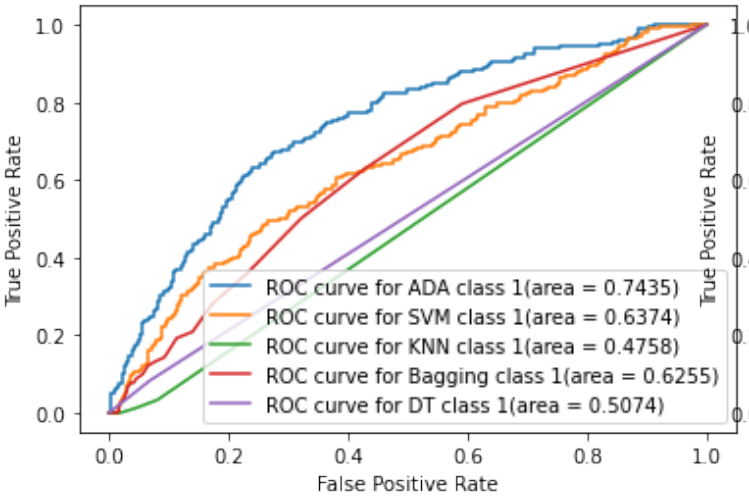


Fig. 19. ROC Curve (PCA - Class 1)

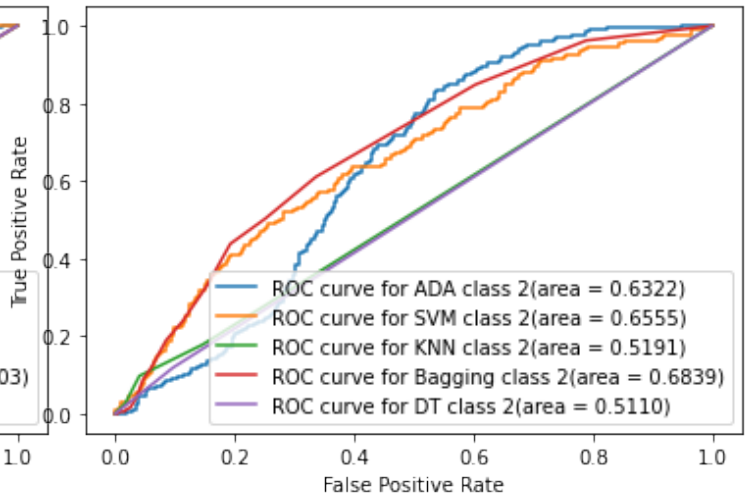


Fig. 20. ROC Curve (PCA - Class 2)

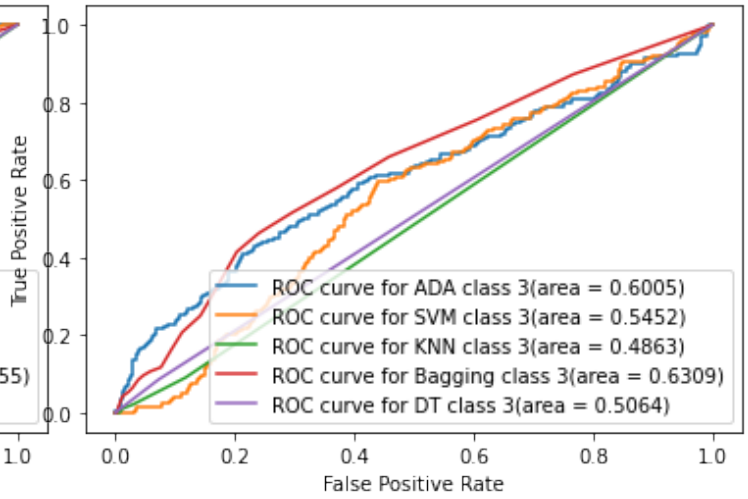


Fig. 21. ROC Curve (PCA - Class 3)

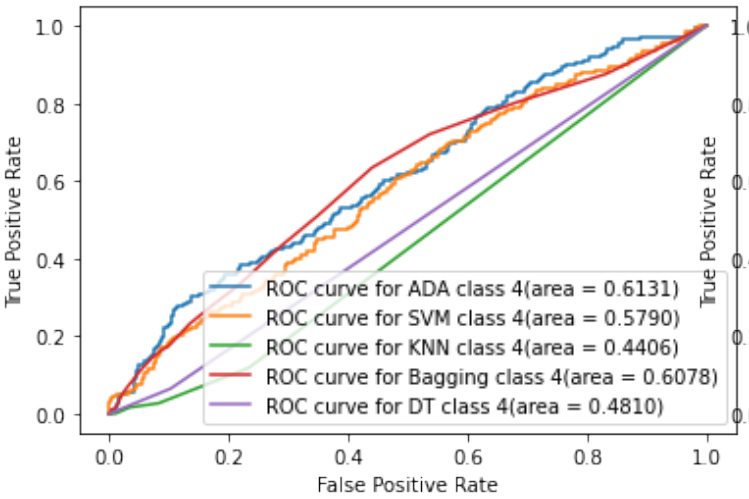


Fig. 22. ROC Curve (PCA - Class 4)

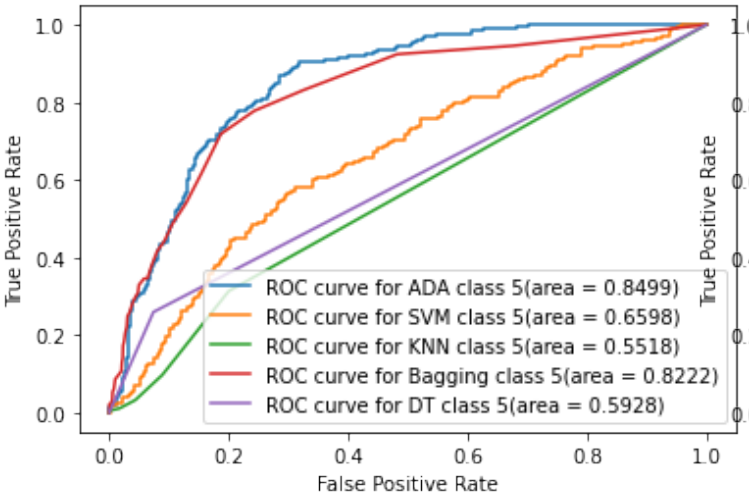


Fig. 23. ROC Curve (PCA - Class 5)

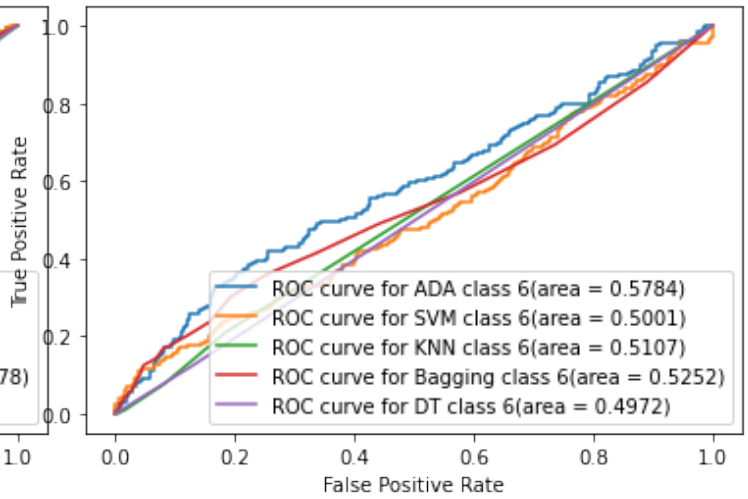


Fig. 24. ROC Curve (PCA - Class 6)

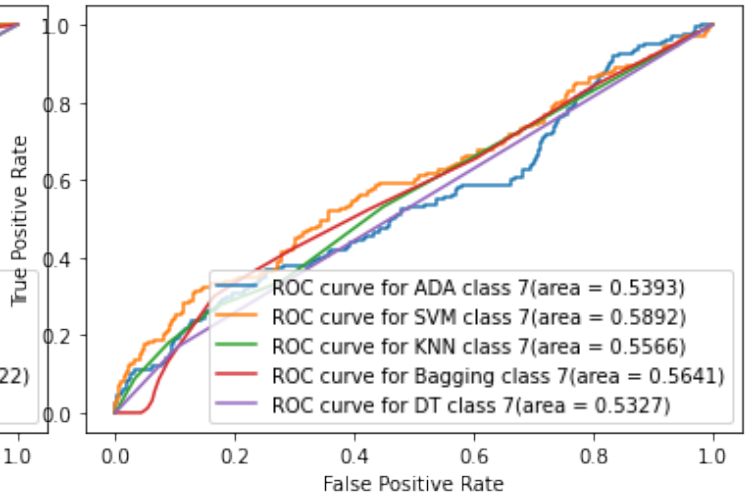


Fig. 25. ROC Curve (PCA - Class 7)

12. ACTIVATION MAP (TRAIN IMAGES)

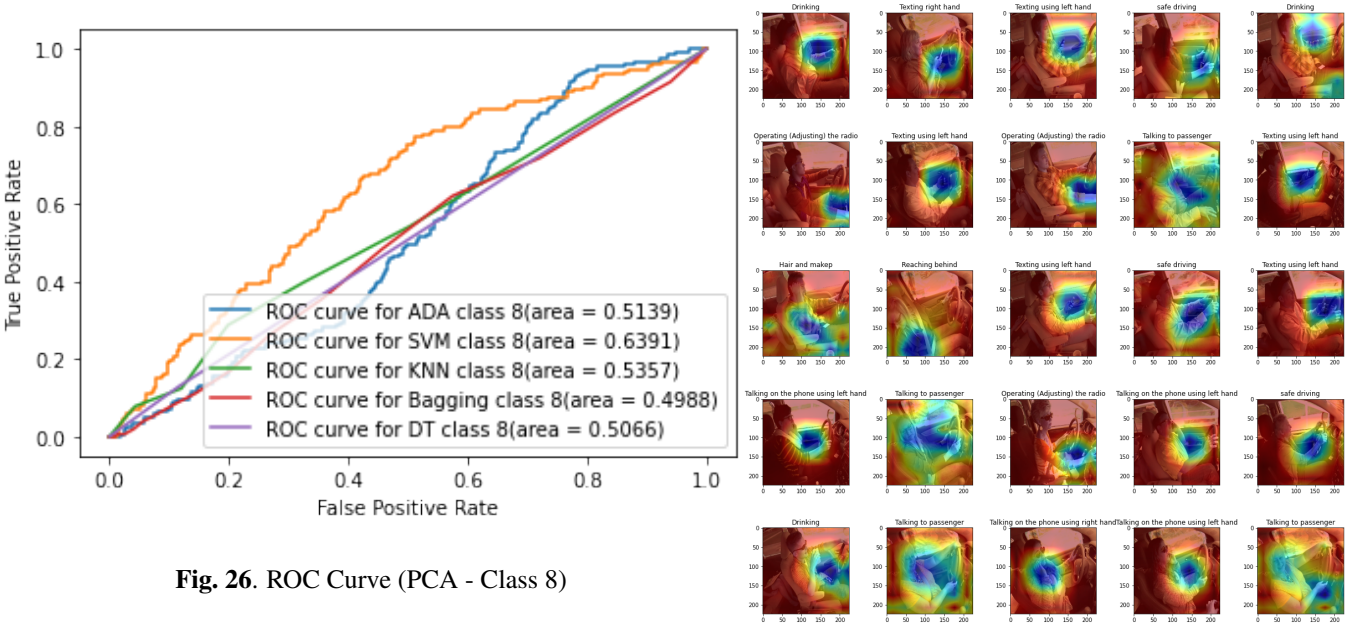


Fig. 26. ROC Curve (PCA - Class 8)

13. FEATURE VISUALIZATION

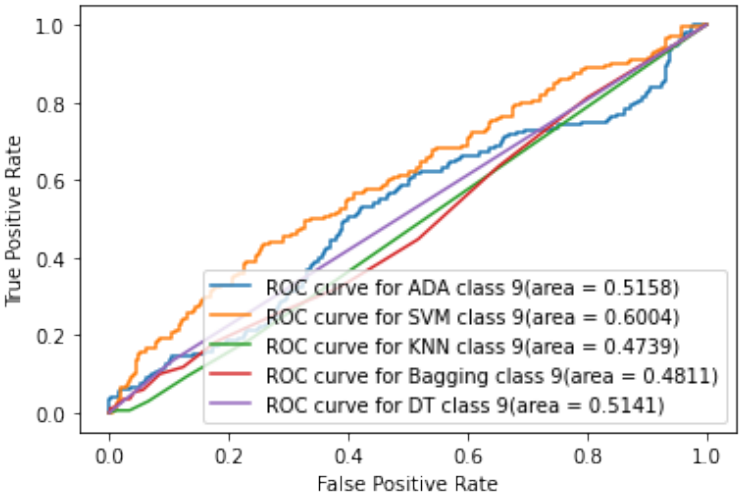
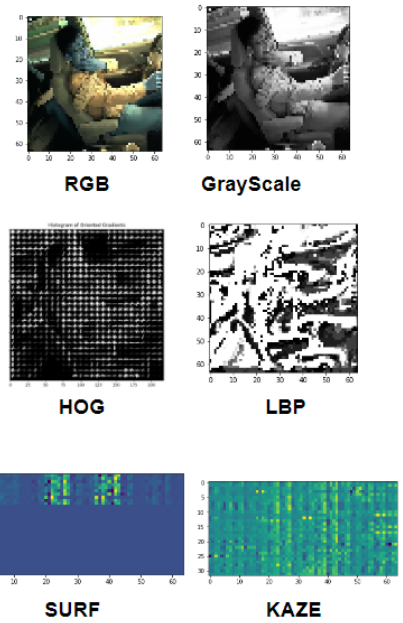
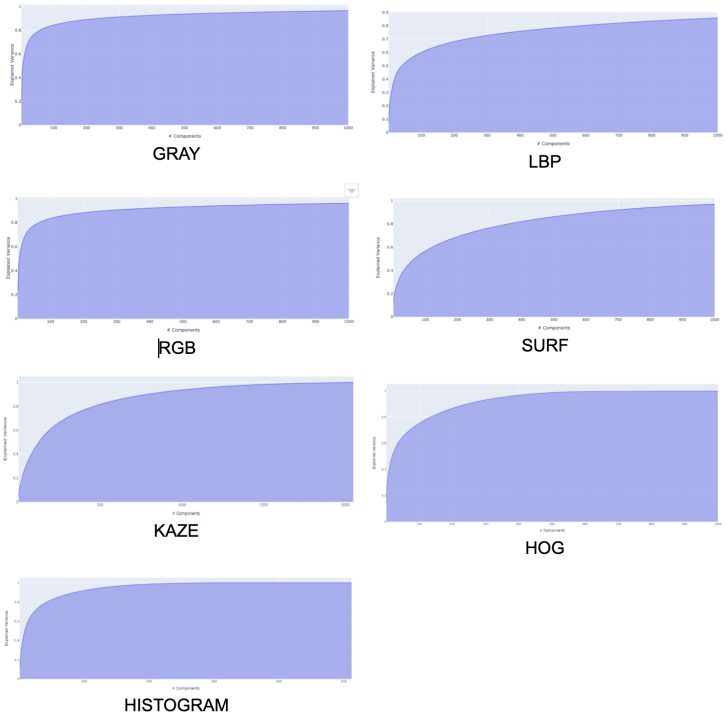


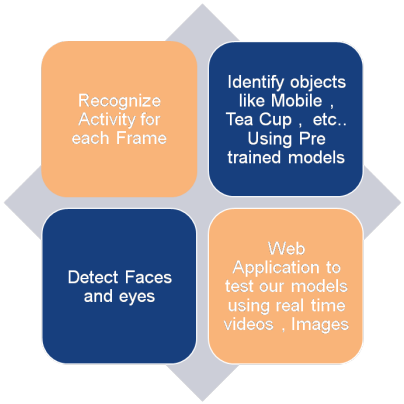
Fig. 27. ROC Curve (PCA - Class 9)



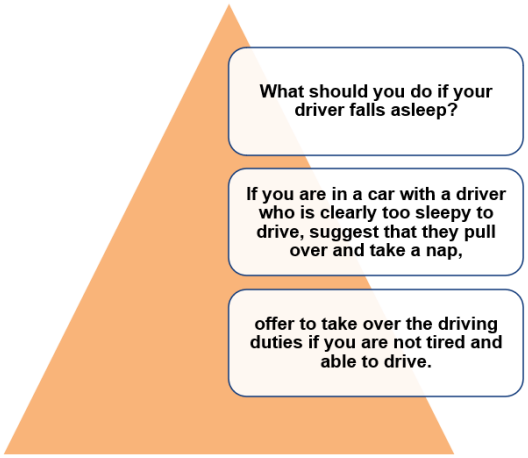
14. VARIANCE OF PCA PROJECTED OVER MIN-MAX NORMALIZED DATA



15. CURRENT FEATURES



16. FUTURE WORK



17. INTERPRETATION OF RESULTS

- LDA gives better results compared to LDA and LDA over PCA
- Bagging and Decesion Tree performed poorly on all sets of data and proved to be a bad choice as data does not suffer from high variance.
- SVM and KNN projected feature set (HOG, LBP, Color Hist, SURF, GRAY, KAZE RGB) gave the best metric scores because of the rich features and kernel tricks to classify data.
- Among Deep Learning methods, ResNet-101 with stratery-2 i.e retrain last few layers gave best accuracy.
- Feature Extration and Image Augumentation techniques helped in improving overall accuracy.

18. TEST RESULT

Predection

C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	Final Class
0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	Reaching behind

Predection

C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	Final Class
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	Hair and makeup

Predection

C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	Final Class
0.00	0.49	0.00	0.00	0.00	0.00	0.39	0.00	0.11	0.00	Texting - right

Predection

C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	Final Class
0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	Operating the radio

Predection

C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	Final Class
0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	Talking on the phone - right

19. CONTRIBUTIONS

Siva krishna THOTA

20. REFERENCES

[1] Jason Brownlee PhD, *https://machinelearningmastery.com/how-to-get-started-with-deep-learning-for-computer-vision-7-day-mini-course/*, Deep Residual Learning for Image Recognition.

[2] *https://snappishproductions.com/blog/2018/01/03/class-activation-mapping-in-pytorch.html.*

[3] *https://github.com/Abhinav1004/Distracted-Driver-Detection/blob/master/Distracted*

[4] https://github.com/Abhinav1004/Distracted-Driver-Detection/blob/master/Distracted

[5] *https://github.com/Garima13a/YOLO-Object-Detection/blob/master/YOLO.ipynb*.

[6] *https://colab.research.google.com/github/d2l-ai/d2l-en-colab/blob/master/chapter*$_{d}eep - learning - computation/use - gpu.ipynbscrollTo = mwrEJrSCo - OR$.

[7] *https://github.com/opencv/opencv*.

[8] *https://medium.com/@sam.bell*$_{4}3711/distracted - driver - detection - using - deep - learning - ecc7216ae8d0$.

[9] *https://github.com/AkankshaShrimal/Distracted-Driver-Detection*.