International Journal For Technological Research In Engineering
Volume 7, Issue 12, August-2020 *ISSN (Online): 2347 - 4718*

REAL TIME DRIVER MONITORING SYSTEM

Meghana A Rajeev[1], Komal R D[2], Dr Manjula S[3]
[1,2]B.E. Student Dept. of C S & E, JSS S & T U, Mysuru, Mysuru
[3]Assistant Professor, Dept. of C S & E, JSS S & T U

Abstract: Driver distraction is a leading factor in car crashes. Monitoring driver's behavior can significantly help in reducing accidents caused by driver distractions. This study proposes a driver distraction detection system which identifies various types of distractions through a camera observing the driver through a camera and also providing conversational alerts to the driver if he or she is found to be distracted.

In this modern age there is rapid increase in number of vehicles and so is the number of car theft attempts, locally and internationally. Real time vehicle security system based on computer vision provides a solution to this problem. The proposed vehicle security system performs image processing based real time user authentication using face detection and recognition techniques and Raspberry Pi based control system fixed on board with the vehicle. The face of the person which is classified as unknown is sent to the mobile of the owner as an SMS.

Keywords: Raspberry Pi, Deep Learning, Image processing, face recognition

I. INTRODUCTION

With a goal to reduce traffic accidents and improve transportation safety, this study proposes a driver distraction detection system which identifies various types of distractions through a camera observing the driver. We collected a dataset which consists of images of the drivers in both normal and distracted driving postures. Different deep convolutional neural networks including VGG-16, AlexNet, GoogleNet, etc n are implemented and. In addition, we developed a conversational warning system that alerts the driver in real-time when he/she does not focus on the driving task.

In this modern age there is rapid increase in number of vehicles and so is the number of car theft attempts, locally and internationally. With the invention of strong stealing techniques, owners are in fear of having their vehicles being stolen from common parking lot or from outside their home. Thus the protection of vehicles from theft becomes important due to insecure environment. Real time vehicle security system based on computer vision provides a solution to this problem. The proposed vehicle security system performs image processing based real time user authentication using face detection and recognition techniques and microprocessor based control system fixed on board with the vehicle. As the person enters the parked car a camera attached to the driver's seat of the vehicle acquires images of the person and face of the person is detected using Viola Jones algorithm. The extracted face is recognized using the enhanced Linear Discriminant Analysis (LDA) algorithm which discriminates much of the features rather than looking for exact pattern based on Euclidean distance and also reliable to be used with large samples of data. Performing authorization involves setting the threshold value and comparing with that of Euclidean distance above which the person is not authenticated. The face of the person which is classified as unknown is sent to the mobile of the owner as an SMS.

II. PROPOSED SYSTEM

The proposed solution for driver distraction monitoring and conversational alert and driver face recognition and SMS notification system is as follows -

Driver distraction detection and alert:

- The video stream of the driver is taken in real time using a Raspberry Pi camera module.
- We divided the activity of the driver into 10 categories and one of the 10 is safe driving.
- Deep learning can be used to divide the driver activity into different categories like 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 and talking to passengers.
- Different deep learning models have proved to provide good results, so we will be using them for the classification. The different models include - CNN, MobileNet, ResNet, ImageNet, Xception model and AlexNet.
- Based on the result of the classification a buzzer does off to alert the driver. Python library "playsound" is used for it.

Driver face recognition and SMS notification:

- When image quality is taken into consideration, there are a plethora of factors that influence the system's accuracy. It is extremely important to apply various image pre-processing techniques to standardize the images that you supply to a face recognition system.
- OpenCV uses a type of face detector called a Haar Cascade classifier which we have chosen for our project. Given an image, which can come from a file or from live video, the face detector examines each image location and classifies it as "Face" or "Not Face." Classification assumes a fixed scale for the face, say 50x50 pixels.
- The model is to be built on a custom dataset consisting of images of the people authorized to use the given vehicle.
- A video stream of the driver is continuously captured and the face in the stream, if detected is

recognized as either one of the known faces in the dataset or as an "unknown" person.
- If the face is classified as "unknown" an SMS notification is sent to the owner of the vehicle indicating intrusion.

III. IMPLEMENTATION

Driver distraction detection and alert:

Dataset- The dataset chosen for the project was Kaggle's state farm dataset. StateFarm's Distracted Driver Detection competition on Kaggle was the first publicly available dataset for distraction classification. In the competition, StateFarm defined ten activities to be detected: safe driving, texting using right hand, talking on the phone using right hand, texting using left hand, talking on the phone using left hand, operating the radio, drinking, reaching behind, doing hair and makeup, and talking to passenger. Our work, in this paper, is mainly inspired by StateFarm's Distracted Driver's competition. The train and test data are split on the drivers, such that one driver can only appear on either train or test set.

Evaluation Metric- The metric that has been chosen for this project is Log loss which as the confidence with which we classify a driver's action as distracted is very important in evaluating the performance of the model.

Data Leakage- With the understanding of what needs to be achieved, we proceeded to build the CNN models from scratch. We added the usual suspects — convolution batch normalization, max pooling, and dense layers. The results — loss of 0.014 and accuracy of 99.6% on the validation set in 3 epochs.

The accuracy achieved was very high, so we tested the model on the test data and noticed that there was a huge difference in the accuracies. So we further analyzed our data to see what could have gone wrong. We found that our training data had multiple images of the same person within a class with slight changes of angle and/or shifts in height or width. This was causing a data leakage problem as the similar images were in validation as well, i.e. the model was trained much of the same information that it was trying to predict.

To counter the issue of data leakage, we split the images based on the person IDs instead of using a random 80–20 split. Now, we saw more realistic results when we fit our model with the modified training and validation sets. We achieved a loss of 1.76 and an accuracy of 38.5%.

Transfer Learning- We used the following models for the classification: VGG16, Resnet50, Xception, and Mobilenet.

Extra Layers - To maximize the value from transfer learning, we added a few extra layers to help the model adapt to our use case. Purpose of each layer:
- Global average pooling layer retains only the average of the values in each patch
- Dropout layers help in controlling for overfitting as it drops a faction of parameters(bonus tip: it's a good idea to experiment with different dropout values)
- Batch normalization layer normalizes the inputs to the next layer which allows faster and more resilient training
- Dense layer is the regular fully- connected layer with a specific activation function

Training- We started by using the ImageNet weights and trained only the new layers since the number of parameters to train would be lesser and the model would train faster.

Optimizer- The most popular algorithm in the deep learning world is Adam which combines SGD and RMS Prop. In our case Adam showed an erratic pattern of descent while SGD was learning gradually. By doing some literature survey, we found that in few cases SGD is superior to Adam because SGD generalizes better. As SGD was giving stable results, we used it for all our models.

Ensemble Models –We also tried multiple ensembling techniques to improve the log loss further according to some research papers:
- Mean Ensembling: This is the easiest and most widely used ensembling method where the posterior probability is calculated as the mean of the predicted probabilities from the component models.
- Trimmed Mean Ensembling: This is Mean Ensembling by excluding the maximum and minimum probabilities from the component models for each image. It helps in further smoothing our predictions leading to a lower log loss value.
- KNN for Ensembling: Since the images are all snapped from video snippets while drivers were engaged in a distracting activity or were driving, there are a substantial number of images from the same class that are similar. Based on this premise, finding similar images and averaging the probabilities over these images helped us smoothen predicted probabilities for each class.
- To find the 10 nearest neighbors, we used outputs from the penultimate layer of the VGG16 transfer learning model as features on the validation set.

Based on the classification result, a buzzer goes off if the driver is distracted. A python library has been used for the alert.

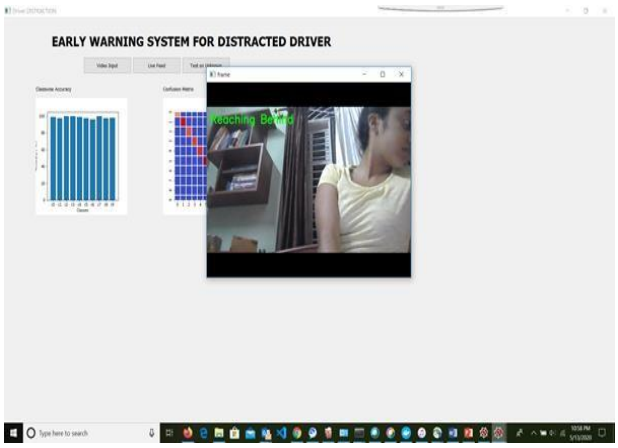

Figure 1: Real time driver distraction detection

Driver face recognition and SMS notification:

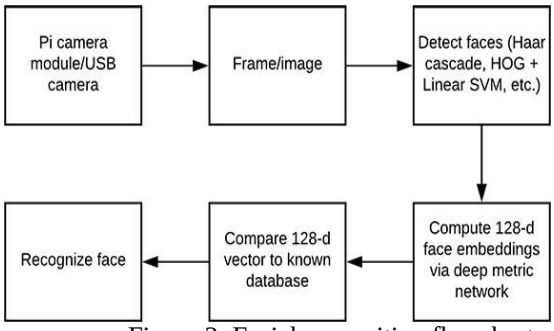

Figure 2: Facial recognition flowchart

We first constructed our own dataset for the system using a python script that uses OpenCV. 95 images were collected to form the training set.

The training algorithm runs through the images. Faces are detected using CNN and HOG and bounding box rectangles are drawn which correspond to the location of the face.Then 128-d encodings are calculated for each face, thereby quantifying the faces.
The algorithm loops over the faces and the following are done in each loop:

- Extract the person's name from the path
- Load and convert the image to rgb
- Localize faces in the image
- Compute face embeddings

In the real time face recognition system, video streams are captured from the webcam and Haar cascade is used to detect and localize the face.
Frames are grabbed and preprocessed. We convert the images into grayscale and RGB. Haar Cascades has been used to detect faces in the frames. Encodings are created. Then we look for possible matches.
In order to send text message notifications containing images of an intruder to our smartphone, we used the Twilio API. The Twilio API is free (with some minor restrictions) and is very simple to use.
Twilio and boto python packages have been used to send the message.
The public server that we used is Amazon S3.
A separate thread was created which was used to upload the image to S3 and then send it over the wire via the Twilio API. We use threading in this case, so we don't slow down our main video processing pipeline due to I/O latency. The Thread makes a call to a _send function. We connect to Amazon S3 using our supplied credentials followed by grabbing our bucket.
Once we created our bucket, we created a new file using the Key class by uploading the image to S3, making it public, and finally generating a URL for it.
After the message is sent, the temporary image is deleted from the bucket.
We start looping over frames from our video stream obtained from the camera, poll them one by one, pre-process the frame, and then detect faces in the image. We also draw the current timestamp on the frame.

The detected faces are looped over, and the face ROI are extracted and passed over to the face identifier.
If the consec variable is None, we initialize it as a list, containing the name of the face and the number of consecutive frames the face has appeared in.

Otherwise, if the predicted face matches the name in consec, then we update the consecutive frame count.
We then check if the captured face is an intruder or not. If the predicted face is Unknown, and has been Unknown for a sufficient number of frames, then an intruder has been detected.
Then a bounding box + name of the face in the frame, followed by checking to see if:
An intruder has been detected and
Enough time has passed in between Twilio message sends
Provided that we have labelled a face as an intruder, we handle that by sending the entire frame to our smartphone via Amazon S3 + the Twilio API.

If an intruder has been found in more than 40 frames, the face of the intruder is sent as a link to the owner along with a message.

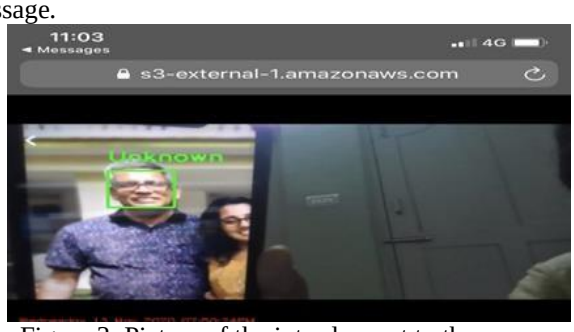

Figure 3: Picture of the intruder sent to the owner, which will be encountered in more than 40 frames

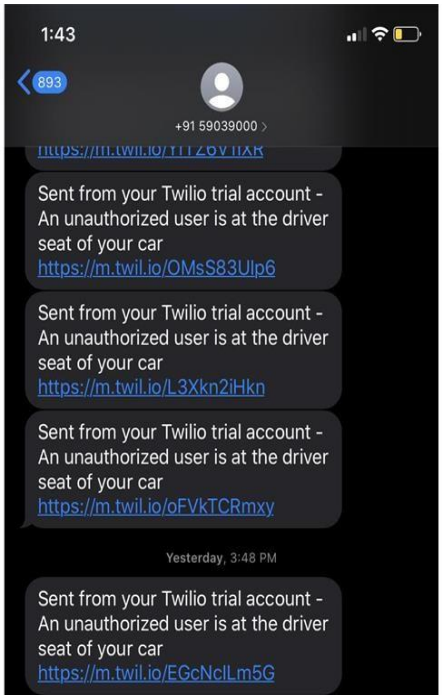

Figure 4: Message sent to the owner when an intruder is detected

IV. SYSTEM TESTING AND ANALYSIS

Driver distraction detection and alert system:

Model	Validation Loss	Validation Accuracy
VGG 16	1.6224	76%
Resnet	0.61943	80.96%
Mobilenet	0.4	85.7%
KNN Ensemble	0.24	89.97%
Trimmed Mean Ensemble	0.3	89.7%

Table 1: Comparison of accuracies achieved

SGD gave us stable results for all ourmodels when compared to Adam optimizer.

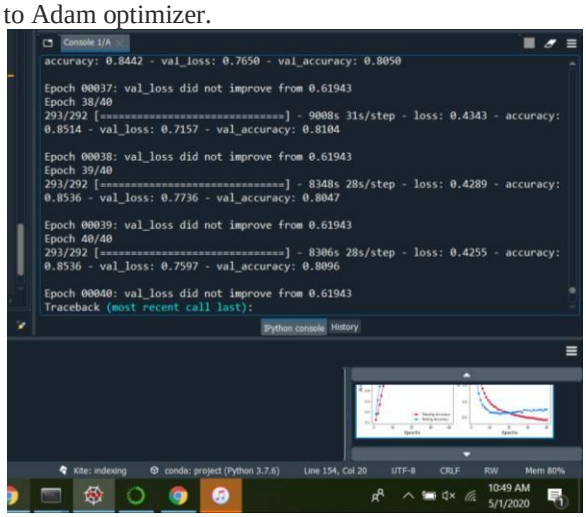

Figure 5: Screenshot of the python kernel after 40 epochs of the Resnet model was completed

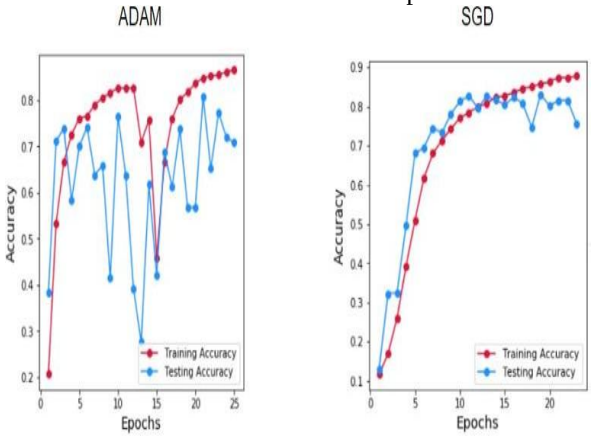

Figure 6: Accuracies observed for Resnet for the two optimizers Adam and SGD

Our real time system works as 5.7 FPS. The system sets off an alarm when the driver is distracted.

Driver Face Recognition and SMS Notification:

Our face recognition pipeline is running at approximately 6-7 FPS. The vast majority of the computation is happening when a face is being recognized, not when it is being detected. Furthermore, the more faces in the dataset, the more comparisons are made for the voting process, resulting in slower facial recognition.

Computing the full face recognition (i.e., extracting the 128-d facial embedding) once every N frames (where N is user-defined variable) and then applying simple tracking algorithms (such as centroid tracking) to track the detected faces will reduce the time taken. Such a process will enable us to reach 14-15 FPS on the Raspberry Pi for face recognition. We have although computed the embedding after each frame as 6-7 FPS is a considerable good rate.

We have created a separate thread and used it to upload the image to S3 and then send it over the wire via the Twilio API. We use threading in this case, so we don't slow down our main video processing pipeline due to I/O latency.

V. CONCLUSION AND FUTURE WORK

Distracted driving is a major problem leading to a striking number of accidents worldwide. Its detection is an important system component in semi-autonomous cars. We presented a robust vision-based system that recognizes distracted driving postures. We collected available distracted driver dataset that we used to develop and test our system. Our best model utilizes a genetically weighted ensemble of convolutional neural networks to achieve a 90% classification accuracy. We also showed that a simpler model could operate in real-time and still maintain a satisfactory classification accuracy.

The futuristic scopes of this project are the following features:

The following features can be included and was not included in the current phase as it was infeasible for us

- Encryption between the communication mobile devices to Raspberry Pi
- Motion sensing to start capturing video streams only when a motion is detected inside the vehicle
- Currently we are only alerting the driver with a buzzer when the driver is distracted. For an organisation, it would be more helpful for concerned authorities to receive notifications when their drivers are distracted. There is a lot they can do with this information. This feature can be added to the project.
- Security in terms of accessing the Raspberry Pi from a third-party device
- Security in terms of communication between Raspberry Pi and mobile application
- Lock mechanism to attach system to allow access only to authorised drivers.

REFERENCES

[1] C. Craye and F. Karray, "Driver distraction detection and recognition using RGB-D sensor," 2015, https://arxiv.org/abs/1502.00250.

[2] F. Coenen, B. Zhang, and C. Yan, "Driving posture recognition by convolutional neural networks," IET Computer Vision, vol. 10, no. 2, pp. 103–114, 2016.

[3] R. A. Berri, A. G. Silva, R. S. Parpinelli, E. Girardi, and R. Arthur, "A pattern recognition system for detecting use of mobile phones while driving," in Proceedings of the 9th International Conference on Computer Vision Theory and Applications, VISAPP 2014, vol. 2, pp. 411– 418, IEEE, Portugal, January 2014.

[4] T. H. Le, Y. Zheng, C. Zhu, K. Luu, and M. Savvides, "Multiple scale faster-rcnn approach to driver's cell-phone usage and hands on steering wheel detection," in Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 46–53, IEEE, Las Vegas, Nev, USA, June 2016.

[5] https://towardsdatascience.com/distracted -driver-detection-using-deep- learninge893715e02a4

