Drowsy Driver Detection Using Deep Learning Technologies

1st Aishwarya Mohan
Department of Applied Data Science
San Jose State University
San Jose, USA
aishwarya.mohaniyengar@sjsu.edu

2nd Gangotri Biswal
Department of Applied Data Science
San Jose State University
San Jose, USA
gangotri.biswal@sjsu.edu

3rd Kaamya Ravikumar
Department of Applied Data Science
San Jose State University
San Jose, USA
kaamya.ravikumar@sjsu.edu

4th Palak Shah
Department of Applied Data Science
San Jose State University
San Jose, USA
palak.pankanjkumarshah@sjsu.edu

5th Pravallika Vallapuri
Department of Applied Data Science
San Jose State University
San Jose, USA
pravallika.vallapuri@sjsu.edu

***Abstract*—Humans have always invented machines and devised techniques to ease and protect their lives for daily activities like travelling to work or for more important purposes like aircraft travel. Neglecting our duties towards a safer travel leads to thousands of precious lives being lost and road accidents are a leading reason of these. Although accidents occur due to a number of reasons like breaking traffic rules, drunk driving, etc, accidents that happen due to drivers feeling tired or drowsy and thus loose control of the vehicle has seen an increase in recent times. According to statistics, driver drowsiness was present in around 8.5 to 9.3 percent of all crashes that happen. This usually happens during late night or early morning or late afternoon periods. In this project, we have proposed VGG and AlexNet models with an accuracy of 98.2% and 96.5% respectively for the classification of driver drowsiness and a demo model that will detect driver drowsiness in real time based on the best performing model and trigger an alarm that will be triggered in case the person's eyes are closed beyond a certain limit. Future Scope and other related works have also been discussed.**

***Index Terms*—driver drowsiness, road accidents, AlexNet, Deep learning models, VGG**

I. Introduction

Countless number of people drive on the highway during the day and night. Taxi drivers, bus drivers, truck drivers and people traveling long-distance suffer from lack of sleep due to which it becomes very dangerous to drive when feeling sleepy. The National Highway Traffic Safety Administration estimated that drowsy driving is estimated for 91,000 traffic accidents.Drowsiness detection is a safety technology that can prevent accidents that are caused by drivers who fell asleep while driving.

A. Background

According to the survey done by the National Safety Council (NSC) , each year drowsy driving has caused 100,000 crashes among which about 71,000 are injuries, 1,550 are fatalities. There are also many incidents which have happened and killed entire families in car crashes. This would have been avoided if there was an alarm in the car which detects if the driver is drowsy. If one of the detection and alarms at the right time would say many lives. Why not? Keeping this as a social cause and also motivating the number of deaths which have happened due to drowsiness of drivers, our team has done this proposal to detect the drowsiness of drivers and also give an alarm at the right time which would avoid serious accidents.

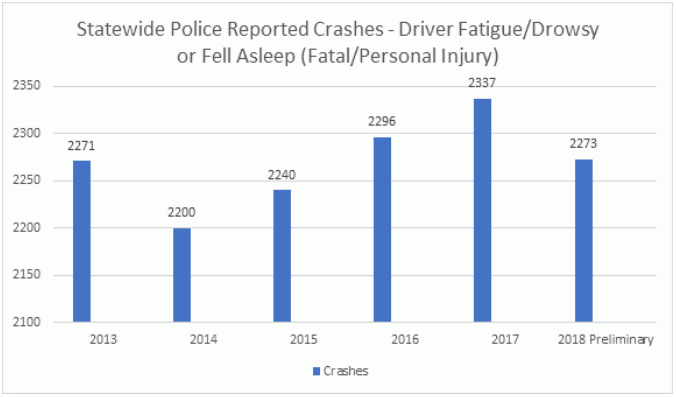

Fig. 1. Statistics of Crashes Due to Driver Drowsiness

Fig. 1 clearly shows the impact caused due to driver drowsiness over the years

II. Motivation

We all for a fact know that, we won't be aware of the moment we fall asleep.People usually have a habit of falling asleep watching a Netflix series or a movie during the night.

These are the things which are done while sleeping at night. But it causes severe damage if it is done behind the wheels. Usually people tend to feel drowsy as soon as they have some heavy food. During travel the drivers usually take light meals in order to avoid any drowsiness.After all that, there are instances where there has been an accident to driver drowsiness. The driver will not be putting just his life in risk but also the people who are traveling with him. This becomes a social problem. This created a motivation in us to attempt to detect the drowsiness of the driver. If there are too straight roads there are high chances of drivers falling asleep. As there won't be much of a curves, mountain or any distraction in a straight expressway, driver tends to fall asleep and by the time he wakes up there would be a high chance that it takes at least few seconds for him to release that he is in driving seat and that moment would have caused the accidents. As the cars will be very fast in the expressway, if there is an accident of one car the cars which are following tend to join the rash which lead to loss of many lives.

III. Literature Review

Arafat Y. et al [1] have proposed a methodology based on Convolutional Neural Networks that illustrates drowsiness detection as a task to detect an object. Based on real time video stream of drivers, it will detect whether the eyes are open or closed. The authors have used MobileNet CNN architecture with single shot multibox detector as the technology for object detection [1]. Their proposed approach is to ensure better accuracy and computational efficiency [1].

Manishi. et al [2] have proposed an advanced methodology to detect drowsiness based on multiple facial features like eye sight and yawn. The data source for their research is from the NTHU Drowsy Driver Detection Repository. They have used the Voila Jones Algorithm to detect faces and AdaBoost Classifiers have been used to select the features and cascade of classifiers have been used to extract the features [2]. They were able to achieve a 98.4% accuracy for the overall model.

Yi Z. et al [3] published a paper in which they have utilized behavior of the driver and running state of the vehicle through simulation experiments. Factor Analysis was used to reduce the dimensionality of these parameters and the corresponding scores were computed. Particle Swarm Optimization was used to divide the scores into information granules and the mean and standard deviation was computed and fused into one single data. Finally LIBSVM was developed as the final model which achieved an 86.47% accuracy.

Fig. 2 provides a comparison of the different techniques researched by other authors along with their advantages and disadvantages

A. Our Proposed Model

Several Papers have utilized different techniques to classify drivers as drowsy or not but what was lacking was a clarity on why certain models worked and why others did not. We hope to implement multiple models to present a comparative study and also provide a justification on why a particular model worked best for this project.

We will also not stop with just building a classification model but we will aim to implement a real time alarming system that will trigger when the driver seems drowsy.

Source of Paper	Methodology	Advantages	Disadvantages
Paper 1	This paper proposes a model that consists of two different components which are CNN and LSTM. Sequences of video frames were used as input to the framework. The NTHU dataset was used to train and test the model.	The model was implemented by using a number of CNN extractors in parallel. The model took temporal dependencies of input data into consideration.	The NTHU DROZY dataset which was used in this research paper lacked proper labeling of input images.
Paper 2	The camera captures the image and sends it to the raspberry pi. The data is encoded by the encoder and the encoded signal is decoded back in the decoder. The LPC2148 is the arm used in the microcontroller. If the decoded signal exceeds the 2-3 sec threshold, the alarm will automatically turn on and the parking light will be on in order to alert the driver.	The driver is alerted in real time. The drowsiness is detected irrespective of the lighting conditions or the driver wearing spectacles. Drowsiness is detected within a span of a few seconds.	The proposed methods involve the usage of sensors which might age with time.
Paper 3	This paper implemented pre-trained models like ResNet and VGG to detect the drowsiness of drivers and added additional layers to customize the model architecture.	The authors have used a combination of 3 different datasets to implement the model. This is done to make sure the model is trained on a variety of images that are of multiple races and ethnicities. This ensures that the model does overfit for a particular dataset and generalizes well.	The custom developed test dataset was not available to use publicly to develop more innovative model architectures based on that.
Paper 4	This paper focuses on using Haar algorithm for detecting eye and facial features. The SVM classifier is further trained on different states of the eyes in order to trigger the alarm sound accordingly.		These models have a higher training time and higher computational cost. There is a trade off between size of the dataset and efficiency. When we want to use more data to create a more generalized model, the efficiency of this classifier reduces.

Fig. 2. Literature Survey Comparison

IV. Project Methodology

This section will include the data collection, data preprocessing, data modeling, specifications of the model and its final implementation.

A. Data Collection

The data set collected for this project is taken from Kaggle dataset and the data available from s3 bucket which consists of images of eyes that are 'open' and 'closed'. The Images taken from the two sources are merged manually and the final data set contains of 1452 records and the shape of the image is 32*32*3 since the images are in RGB format. Below shows the final dataset extracted from kaggle and S3 bucket.

Fig. 3. Sample Image of Closed Eye from the Dataset

Sample images from the source data has been provided in Fig. 3, and Fig. 4

Fig. 4. Sample Image of Closed Eye from the Dataset

B. Data Pre-Processing

Once the dataset is collected, the next step includes pre-processing. In preprocessing, the images are resized as the architecture requires all the images to be in fixed size when we are feeding the data to the model. Below figure shows how we resize the images.

```
for i in range(len(X)):
    img = X[i]
    img = cv2.resize(img, (32, 32))
    X[i] = img

print(len(X))
print(X[0].shape)
```

```
1452
(32, 32, 3)
```

Fig. 5. Code Snippet of Preprocessing the Images

We have used label encoders to transform our images. Here we are transforming the images from non numerical data to numerical data. We are doing this to normalize labels. We are normalizing labels to accelerate learning and convergence.

Once the data is preprocessed, the next step is Data Preparation. The dataset is split into train and test ratio of 80 to 20, where the train data contains 1161 images of 32*32*3 shape and test data contains 291 images. While fitting the model the validation data is considered from the test data in the same 80 to 20 ratio.

C. Data Modelling

Convolutional Neural Networks or CNN is a type of a neural network that is most often applied to image processing problems. It can also be implemented in natural language processing problems too. All CNN models consist of an input layer, output layers and several hidden layers. Based on how these layers are built, they are classified into several architectures.

```
label_encoder = LabelEncoder()
Y = label_encoder.fit_transform(Y)
print(Y.shape)
print(Y[0])
print(set(Y))
```

```
(1452,)
0
{0, 1}
```

Fig. 6. Code Snippet of Preprocessing the Images

We have implemented two architectures of CNN in addition to the baseline customized CNN model which was referred to as part of our literature review. The two models which we have implemented are VGG and AlexNet architectures. The baseline CNN model consists of 7 convolution layers and 3 fully connected layers with hyper parameters like batch normalization, max pooling and dropouts. We further wanted to implement these two architectures to obtain a comparative study and understand the impact it would have to model our chosen dataset.

D. VGG Architecture

The VGG stands for Visual Geometry Groups and the architecture has several variants of it like VGG-19, miniVGG, etc. For our project we have implemented a VGG-16 architecture. The actual architecture consists of 21 layers including 13 convolutional layers, 3 fully connected layers and 5 max pooling layers. The number 16 refers to the 16 layers that have learnable parameters. This network was developed by Simonyan and Zisserman for the ILSVRC 2014 competition. All the layers of the VGG network consist of only 3x3 kernels. The network has a total of 138 million parameters. The figure below shows the overall VGG network architecture.

The architecture of VGG is based on the idea that large number of hyperparameters can be replaced by consistently having just convolution layers with 3x3 kernels with a stride value of 1 and same padding and maxpool layers are used throughout the architecture with 2x2 kernels and a stride value of 1.

E. AlexNet Architecture

AlexNet is one of the earliest deep learning architectures of the 21st century. LeNet and AlexNet have a very similar architecture design. There are also several variants of AlexNet. AlexNet consists of 5 convolutional layers and 2 fully connected layers and a softmax layer. Each convolutional layer consists of kernels of a specific size and MaxPooling layer. It has around 60 million parameters. Layer 1, Layer 2 and Layer 5 are followed by a max pooling layer which reduces the dimensionality of the input by a significant amount. The number of layers, kernel sizes, number of neurons in

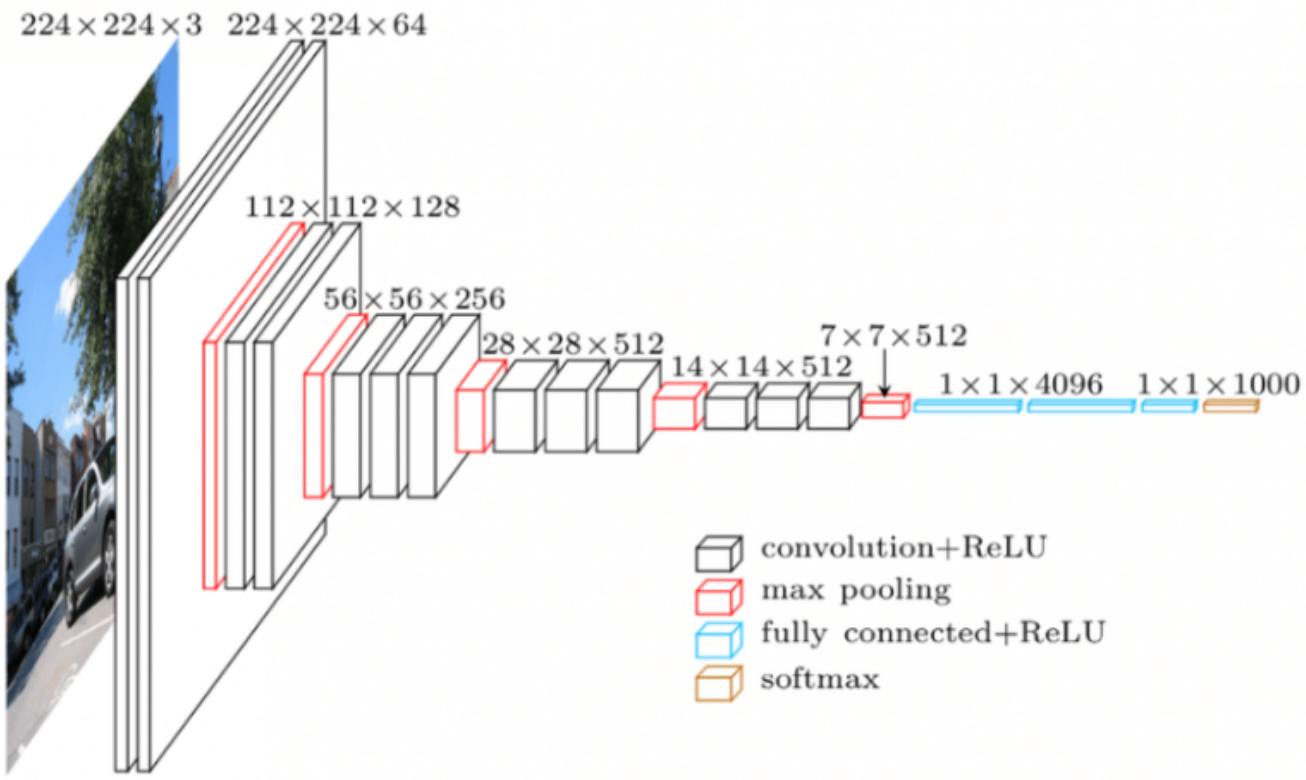

Fig. 7. Overall VGG Architecture from Google Database

fully connected layers and addition of regularizer are some of the hyperparameters which can be added to the existing architecture to make it more efficient.

F. Model Specifications and Training

All the deep learning model architectures were implemented using Keras library. The overall implementation of our project requires the installation of the following software libraries specified in Table 1.

TABLE I
PACKAGED REQUIRED FOR EXECUTION

Tools	Justifications
OpenCV	Detection of facial features
Tensorflow	Prerequisite for Keras
Keras	Used to develop the model architecture
Pygame	Trigger the alarm sound

The following specifications and hyper parameters specified in Table 2 were used to train the CNN models. We have trained all the models using the same hyperparameters in order to achieve accurate comparative study.

TABLE II
MODEL SPECIFICATIONS

Specifications	Values
Activation Function	ReLu
Optimizer	Adam
Epochs	25
Batch Size	64
Input Size	32x32x3

G. Demo Implementation

Step 1: As a first step we will use images as input using a webcam. We have implemented the code separately that will trigger the webcam and capture each frame in an infinite loop. OpenCV provides the method cv2.VideoCapture to access the webcam and set the capture object (cap). Cap.read() method will read each frame and this image is stored in a frame variable.

Step 2: The next step would be to detect the face in the image and create a region of interest (ROI). In order to detect the image, we will first convert the image from RGB to grayscale as the OpenCV algorithm takes gray images as the input. Haar cascade classifiers will be used to detect faces and we do not require coloured images for this.

Step 3: The next step is to detect the eyes from the face. In order to do this, the cascade classifier is set for the eyes in

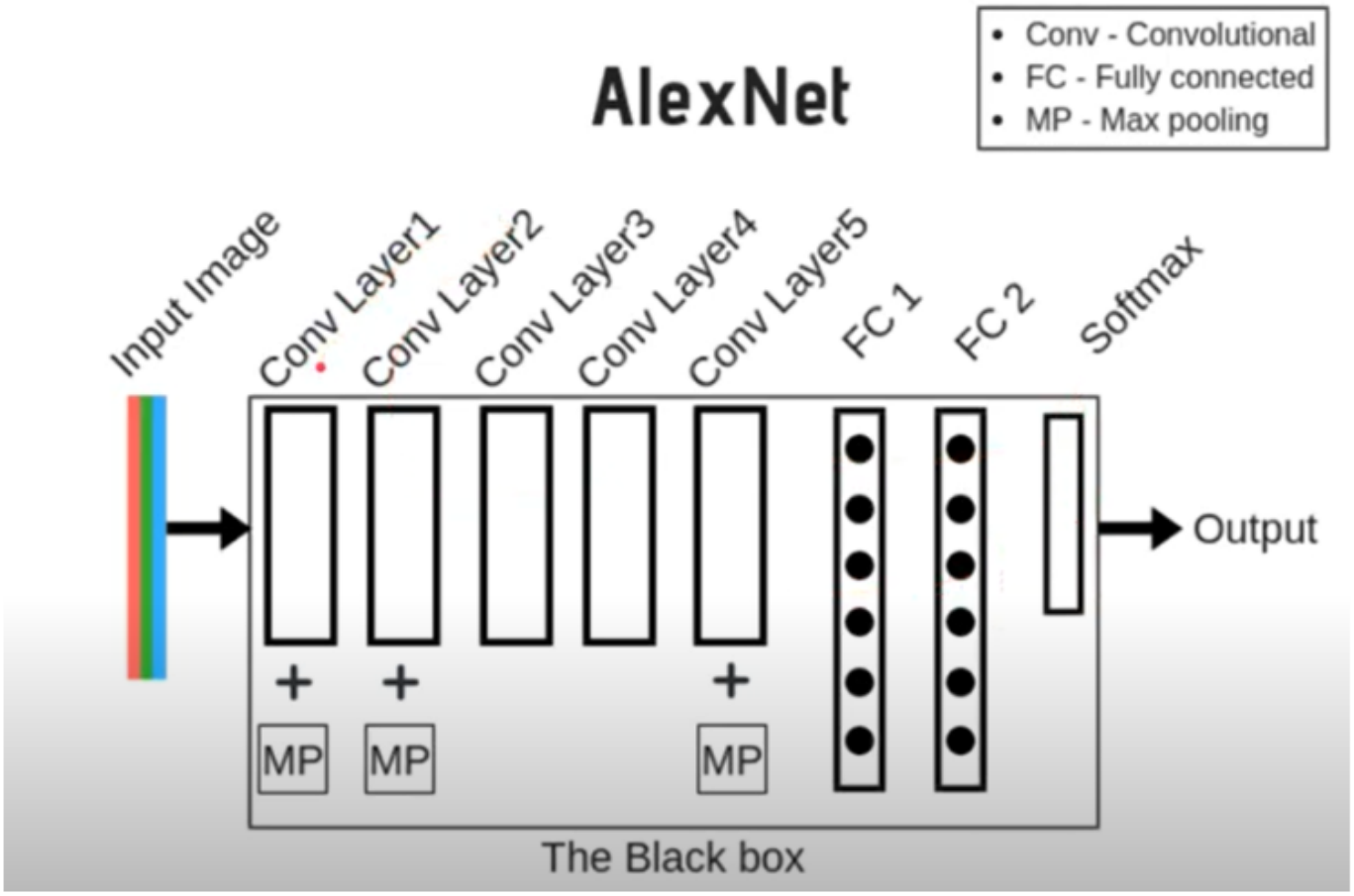

Fig. 8. Overall AlexNet Architecture from Google Database

leye and reye respectively. Only the eyes are required to be extracted from the whole image. This is done by extracting the boundary box of the eye and then the eyes are pulled out from the image using this code. We will extract the left and right eye into separate variables and these variables will only contain the image data of the eye. This will further be fed into the classifier to predict if the eyes are open or closed.

Step 4: The classifier will then categorize the eyes to be open or closed. This is achieved by the CNN architecture that we have built already. First the image is converted from color to grayscale. The images are then resized and normalized for better convergence. Each eye is then predicted to be open or close and the values are set as 1 or 0 resp.

Step 5: The last step is to calculate a score for drowsiness which will act as a threshold to trigger the alarm sound. The score increases if the person is closing the eyes for an extended period of time and automatically decrements when the eyes are open. The result is displayed on the screen using cv2.putText() function which will display the real time status of the person. The threshold value to trigger the alarm is set as 15 and the alarm is triggered using sound.play().

V. Model Evaluation

We have evaluated the models based on the accuracy score of each model. The model with the highest accuracy was VGG and this was taken as the best model and saved a h5 file. This h5 file is then used to trigger the demo file for triggering and detecting using the webcam. The graphs for training and validation loss and training and validation accuracy of AlexNet are as shown in the figures below.

The graphs for training and validation loss and training and validation accuracy of VGG are as shown in the figures below.

From the above figures we can see that the gap between the curves for AlexNet and VGG shows that even though both the models are performing fine, AlexNet is slightly underfitting. The accuracy and loss curves for VGG shows good results. The accuracy comparison of all the three models also show VGG to be the best performing model.

VI. Discussions Future Work

From our research, we can conclude that VGG architecture performed better than AlexNet or the baseline model. The

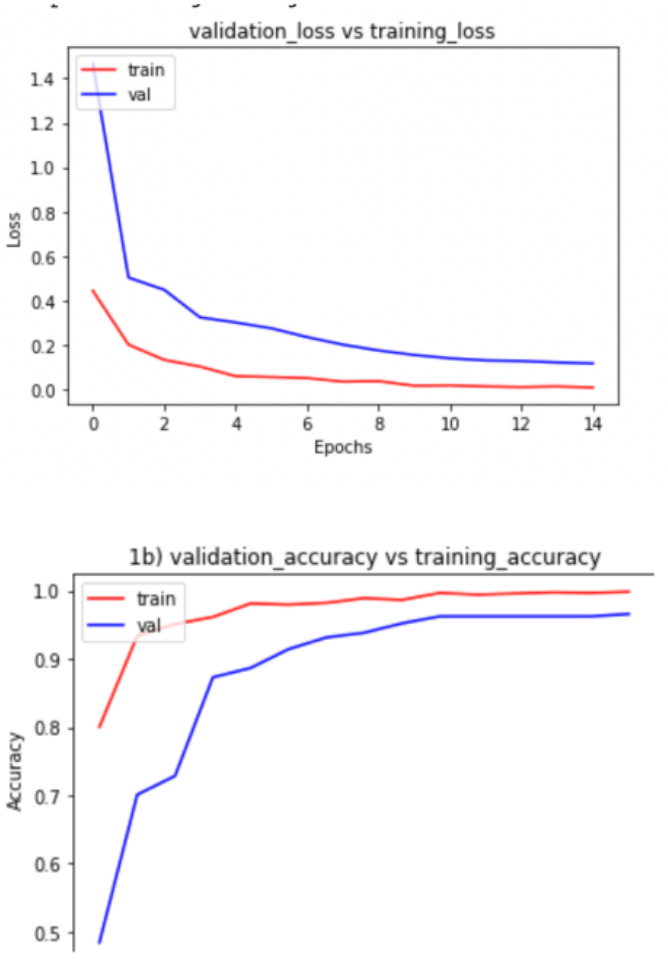

Fig. 9. Loss Accuracy Curves of AlexNet

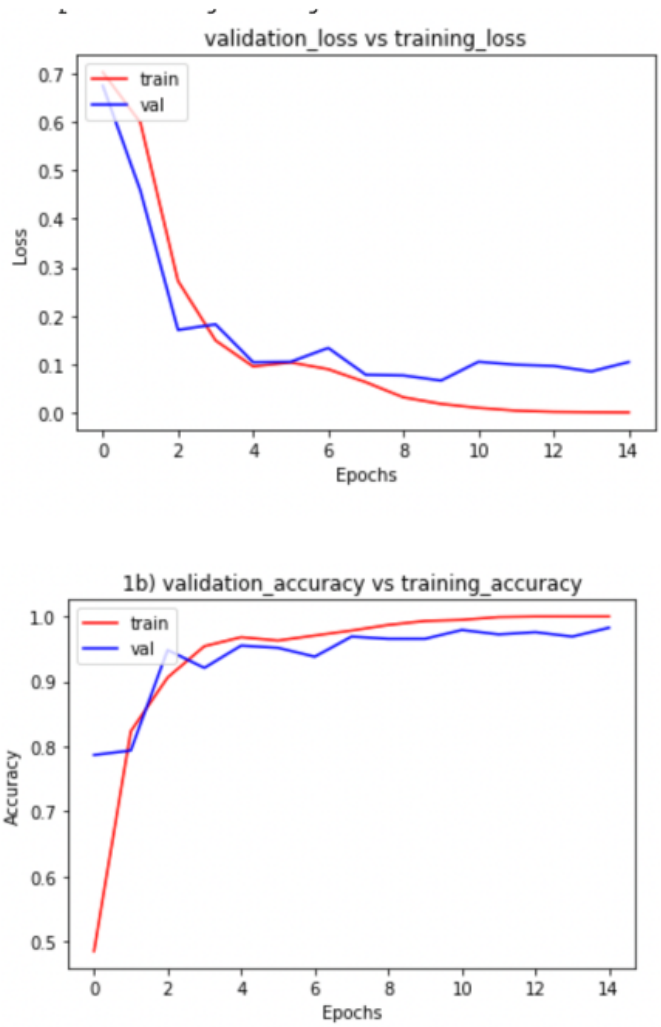

Fig. 10. Loss Accuracy Curves of VGG

TABLE III
MODEL EVALUATIONS

Model	Accuracy
Baseline CNN	87.9
AlexNet	96.5
VGG	98.2

justification for this result would be the fact that VGG has a very deep architecture and such deep architectures would allow the model to identify and learn lower and high level features more efficiently. Also addition of other hyperparameters like dropout, regularization and batch normalization allows the model to have a more robust performance. Some of the scope for future work on this project are as follows:

We have used 2 classes (Eyes open/close) for our model. We will plan to consider 2 more classes (Yawn/No yawn) in the future model.

Conduct additional simulator experiments to validate the algorithm, testing more road conditions.

Build the model based on additional and more diverse groups of data.

ACKNOWLEDGMENT

Our team would like to thank our Professor, Dr. Shayan Shams helping us understand important concepts in the field of Deep Learning Technologies through meaningful assignments and an opportunity to apply the knowledge gained throughout the semester on the term project. Our sincere thanks to our TA Yan Tang, who was always available to help us gain clarity on everything throughout the semester.

REFERENCES

[1] Y. Arafat, F.B. Rafiq, and S.S Baran, "Real-time Driver Drowsiness Detection using Deep Learning," Applied Sciences, vol. 12, no. 7, Jan. 2021, doi: 10.14569/IJACSA.2021.0120794.

[2] Manishi and N. Kumar, "Development of an Enhanced Drowsiness Detection Technique for Car Driver," Oct 2020, doi: 10.37896/jxu14.6/369.

[3] Y. Wang, L. Jin, K. Li, B. Guo, Y. Zheng and J. Shi, "Drowsy Driving Detection Based on Fused Data and Information Granulation," Dec 2019, doi: 10.1109/ACCESS.2019.2960157.

[4] J. Wei, "AlexNet: The Architecture that Challenged CNNs," Medium, Jul. 03, 2019. https://towardsdatascience.com/alexnet-the-architecture-that-challenged-cnns-e406d5297951