2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS) | 978-1-6654-0521-8/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICACCS51430.2021.9441670

Synchronous System for Driver Drowsiness Detection

Using Convolutional Neural Network, Computer Vision and Android Technology

Purvika Bajaj, Renesa Ray, Shivani Shedge
Dept. of CSE
MIT ADT University
Pune, India
purvikabajaj599@gmail.com, rayrenesa@gmail.com,
shibss05@gmail.com

Sagar Jaikar, Pranav More
Dept. of CSE,
MIT ADT University
Pune, India
sagar.jaikar@mituniversity.edu.in,
pranav.more@mituniversity.edu.in

***Abstract*—One of the major reasons behind car accidents is the drowsy nature acquired by a driver while driving any vehicle. Owing to the ongoing scenario, in this project, we aim to develop a real time driver drowsiness detection system in order to detect the drivers' fatigue status, such as dozing, flickering of eye lids and time span of eye closure without having to equip their bodies with devices. The objective of this project is to build a drowsiness detection system that will detect that a person's eyes are closed for a few seconds. This system will alert the driver when drowsiness is detected.** *This approach is based on the Convolutional Neural Network that can beimplemented on Android applications with high accuracy. Apart from CNN, computer vision also plays a major role to detect the drowsiness pattern of the driver. Cloud architecture has also proved to be beneficial in case of capturing and analyzing real time video streams.*

Keywords—Convolutional Neural Network, Android application, Computer Vision, Fatigue Detection, VGG16, Resnet50, GoogLeNet

I. INTRODUCTION

In recent years, it has been predominantly observed that almost every family owns at least one car. Most of the heavy vehicles have to travel during the entire night, where the drivers need to be cautiously awake throughout their journey. Fatigue and microsleep in this situation are often the cause of serious road accidents. The National Highway Traffic Safety Administration's report concluded that a total of 72,77,000 traffic accidents occurred in the United States in 2016, resulting in 37,461 deaths and 3,144,000 injuries [1]. Approximately 20% − 30% traffic accidents were caused by fatigue driving amongst these. However, before a critical situation arises, the initial signs of fatigue can be determined.

Drowsy behaviour of the driver can be easily interpreted by focusing on their facial features, which appears to be quite different as compared to the one when they are not sleepy. In our system we would tend to embed the concepts of Convolutional Neural Network and Computer Vision in an Android application. Our app would be able to capture the state of the individual in real time and accordingly notify through an alarming tone to keep the driver awake. It uses cloud storage for storing the captured live stream which will be accessed by the model for making predictions. In this way, drowsiness could be detected efficiently and an immediate response can be generated according to the situation.

II. RELATED WORK

The [2] Real-time Driver Drowsiness Detection for Android Application Using Deep Neural Networks Techniques, 2018, Procedia Computer Science, Elsevier In this publication, an advanced approach is devised for real-time drowsiness detection. It is based on deploying a deep neural network to an Android application which achieves high accuracy. The most significant achievement of this work is the usage of lightweight model which is obtained from a heavy baseline model. The model achieves an accuracy of more than 80%.

In [3] Real-Time Driver-Drowsiness Detection System Using Facial Features, 21 August 2019, IEEE Access In this work, a system DriCare is introduced, which is responsible for detecting the drivers' weariness status, such as yawning, blinking, and duration of eye closure, using video images, without equipping their bodies with devices. Basically, in DriCare, features of both eyes and mouth are captured and accordingly the device could alert the driver by generating a fatigue alarm. It achieved an accuracy of 92%.

978-1-6654-0521-8/21/$31.00 ©2021 IEEE

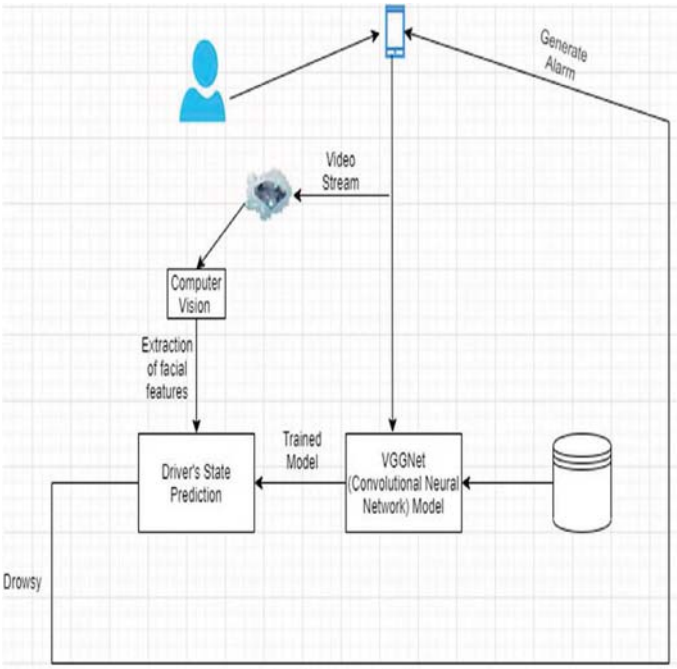

Fig. 1. Architecture of the system

[4] Real time driver drowsiness detection using a logistic-regression-based machine learning algorithm, *2016, IEEE Explore*

An approach that detects the driver's drowsiness by making use of heart rate variation is proposed. An advanced logistic regression-based Machine Learning algorithm is used. The model requires minimal time for predictions and achieves an accuracy above 90%.

[5] Driver drowsy detection using representation learning, *2014*, IEEE

A vision-based driver drowsiness detection system which uses Convolutional Neural Network in order to learn hidden and complex facial representations. The work is based on both qualitative and quantitative results.

[6] Real-Time Driver Drowsiness Detection for Embedded System Using Model Compression of Deep Neural Networks, *2017*, IEEE Explore

A deep learning based approach deployed on a low cost embedded board which attains a high accuracy. The proposed model achieved an accuracy of 89.5% on 3-class classification and speed of 14.9 frames per second (FPS) on Jetson TK1.

III. Proposed System

In the proposed system, an evaluation of the driver's fatigue level is done based on the key facial features. Facial key points define the facial regions of detection. VGG16 is used as the CNN architecture. Therefore, our Android app is a real-time system as it has a high operation speed. It would be able to detect drowsiness pattern of any individual who is using it and accordingly get a visualization graph. Acknowledgement through alarm whenever detection is done works well.

A. Training the CNN Model

The model has been trained and fine-tuned using the drowsiness detection dataset from Kaggle. A comparative study of three pre-trained models namely, VGG-16, Inception and ResNet-50, using transfer learning was performed and VGG - 16 was reported to attain maximum training as well as validation accuracies and least training as well as validation loss. Therefore, the same model has been using in the training.

B. Real-time Working of the System

The android phone mounted in the vehicle records the real-time footage of the driver using the front camera of the mobile. The video streams are stored in a cloud storage and are accessed by the system using OpenCV and the trained model. Computer Vision is used to extract the important facial features of the driver in order to determine whether the driver is sleepy or not. If found sleepy, an alarm is generated on the mobile phone to alert the driver or awake him.

We extract features from the VGG-16 pre-trained model and feed them to the dense layers of our model. Softmax activation function is used in the output layer and has two neurons since there are two classes namely, drowsy and alert. The existing VGG-16 network comes with a softmax layer having 1000 categories. The weights of this network have been used for training and classification.

The output of the VGG-16 network is passed as an input to the CNN model specific to our problem. The second layer used is the Global Average Pooling layer and the output of the previous layer is fed as an input to this layer. It calculates the average output of each of the feature maps in the preceding layer [7]. A dense layer with 128 neurons and relu activation function is used as well. The model is then compiled using sparse categorical crossentropy loss function and adam optimizer.

C. Convolutional Neural Network

A Convolutional Neural Network (CNN) is a type of deep neural network which has at least one convolution layer in its architecture. It performs feature extraction in image or video datasets [8]. It is most commonly used for computer vision and widely used in applications such as object detection, face recognition, classification, etc. The feature selection technique used by CNN makes it very efficient as compared to a feed forward neural network. A CNN consists of the following layers:

- Convolution Layer: it creates a feature map for each of the features in an image by multiplying the input values with the kernel value. The entire image is scanned by the filter.

Identify applicable sponsor/s here. If no sponsors, delete this text box (*sponsors*).

- **Pooling Layer:** it reduces the dimensions of the information extracted by the convolutional layer and keeps the most relevant and important information.
- **Fully connected input layer (Flatten):** outputs of the previous layers is fed as an input to this layer. The flatten layer turns the input into a single dimension vector which in turn, can be used as an input to the next layers.
- **Fully connected layers (Dense):** weights are applied over the input which is the result of feature analysis, for accurate predictions.
- **Fully connected output layer:** this layer generated the final probabilities of each of the classes using activation functions. Usually when there are two classes, sigmoid activation function is used. If and when there are more than two classes, softmax activation is used.
- Relu activation function is most commonly used in convolution layers and dense layers.

D. VGG16

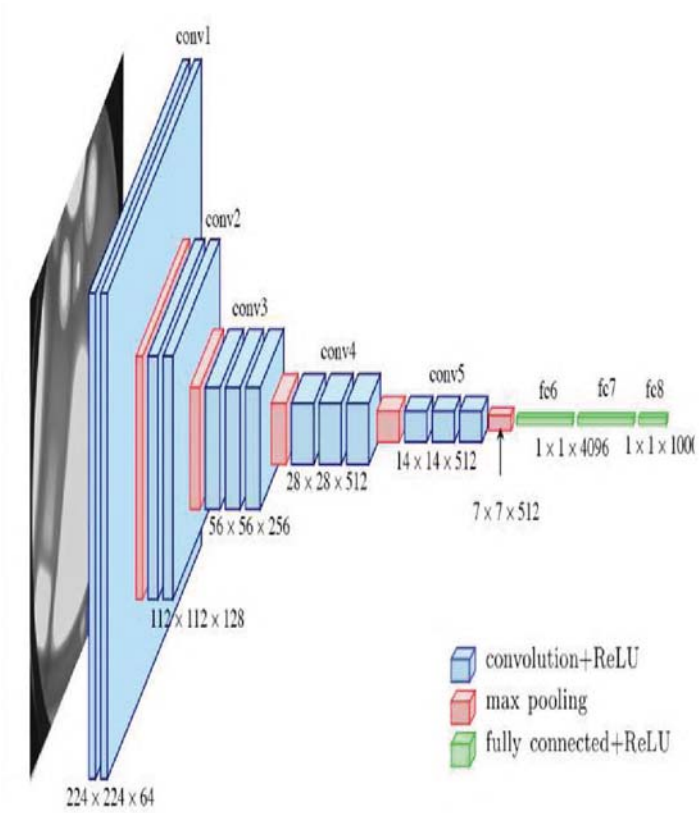

Fig. 2. VGG16 Architecture

Source: https://medium.com/towards-artificial-intelligence/the-architecture-and-implementation-of-vgg-16-b050e5a5920b

(224,224,3) is the size of the input image, that is taken into consideration. It is having one convolution layer, followed by another convolution layer with 64 number of filters. Then it is followed by maxpooling layer of stride *(2, 2)*, following that there are two convolutional layers with 256 filters. Again, a max pooling layer is stacked after this which is similar to the previous layer. Sequentially, convolutional layers are used, where the dimension of the kernel is 3x3, with a count of *256*. The image is then transferred to two consecutive convolution layers. Kernel size is 3*3 throughout, for all the convolution operations. We flatten the output (7,7,512) to make it a *(1, 25088)* feature vector. The first fully connected layer takes input from the last feature vector and outputs a *(1, 4096)* vector, second layer also outputs a vector of size *(1, 4096)* but the third layer on the other hand, lead to thousand channels for *1000* classes of ILSVRC challenge [9]. Then following the result of the last fully connected layer, it is shifted to a layer, having softmax as its activation function to get the standardized values of the probabilistic quantities for each class. Rectified Linear Unit is extensively used in every hidden layers of the model, is computationally more accurate as it leads to speedy learning. As a consequence, the probability of vanishing gradient problem is reduced.

E. Inception

Most of the activations in a complex and deep network are of no use and occurs repeatedly, due to certain dependencies between them. As a result of which GoogleNet was introduced. Sparse connectivity between the activations would lead to a potent proposed system, where not all nodes will see interaction with every other node in the network. The concept of Inception thus came into picture in GoogleNet, that exactly matches a sparse/scattered CNN with a standard dense layout. In order to capture different types of features, this module incorporates all possible filter sizes, like (5x5,2x2,3x3) for convolution at a single layer itself. A major aspect of this module is that, it utilizes 1x1 convolutions. There were 192 networks which consisted of 128 (3x3) kernels and 32 in case of (5x5), as far as GoogleNet's first module is concerned. If we apply all the possible convolutions together, probably that would increase the number of parameters. Therefore, to resolve it, we can consider executing 1x1 convolutions before applying other filters, which would reduce the depth of the obtained output feature map. So, 1x1 convolution is first applied on the input with only 16 filters during the first inception module, until it is forced into 5X5 convolutions. All these improvements make it possible for the network to be wide in width and depth. This neural network architecture got rid of the fully connected dense layers and instead, ended up with global average pooling layer after the last convolutional layer that averages the channel values across the 2D function map. The GoogleNet gives us Auxillary classifiers, which are added in the middle. It has top-5 error rate of 6.67% in classification task. An ensemble of GoogLeNets gives 43.9 % mAP on ImageNet test set [10].

F. ResNet50

Raising the depth can improve the network's precision. but, with the increase in depth, the gradient becomes smaller and ultimately there is no updating of weights at the earlier layers.

This is called vanishing gradient. In addition, training the deeper networks leads to maximizing the enormous parameter space and thus naively introduce the layers that lead to greater error in training. A 20-layer CNN produces less error rate then a 56-layer CNN architecture, on both training and testing dataset. This network uses a VGG-19 inspired 34-layer plain network architecture in which the link to the shortcut is then applied which is then transformed into residual networks through these shortcut links. To form a residual network, a shortcut link is inserted from the VGGNet. On transforming into the residual network, the 34-layer network has much less training error than the residual 18-layer network.

Residual Block:

This design implemented the idea called the Residual Network to address the issue of the disappearing/exploding gradient. A method called skip connections is used in this network [11]. The underlying method behind such a network is that we allow the network to match the residual mapping instead of layers learning the underlying mapping. Rather than, the initial mapping, say H(x), let the network fit,

$$F(x) := H(x) - x \text{ which gives } H(x) := F(x) + x \qquad (1)$$

The main advantage of this type of skip linkage is that whichever layer impairs the structural performance, is omitted by regularization. As a result, the deep neural network can be trained without facing the problems based on vanishing or exploding gradient.

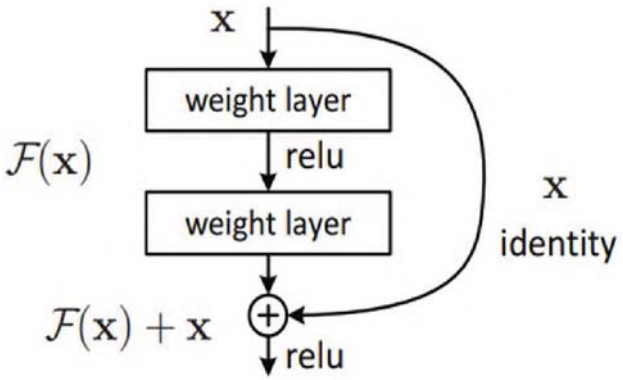

Fig. 3. Residual Block

Source: https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

IV. Data Visualization

A. Total images count

The graph represents the total count of images belonging to the two categories: drowsy and alert. The training dataset consists of 716 alert images and 437 drowsy images. On the other hand, the validation dataset consists of 250 alert and drowsy images each.

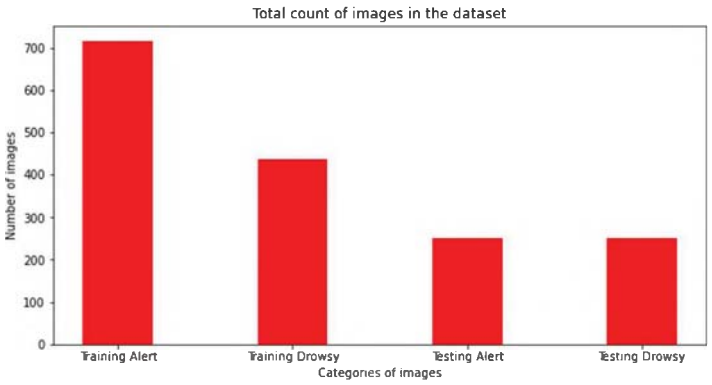

Fig. 4. Total number of images in the dataset

B. Accuracy Curves

The accuracies attained by using different models are plotted against the number of epochs. The x-axis represents the number of epochs and the y-axis represents the number of epochs. The number of epochs is 30 in all the cases.

1) *VGG16:* Accuracy attained on training the model is 90% and the validation accuracy is 92%. It is observed that the training accuracy increases steadily whereas, fluctuations are noted in the validation accuracy.

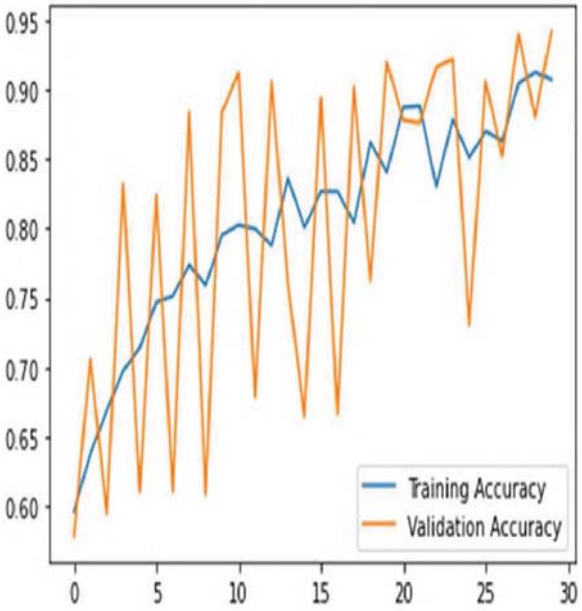

Fig. 5. Accuracy Curve of VGG16 Model

2) *Inception:* The training accuracy achieved is 89% whereas the validation accuracy is 83%. The fluctuations observed in validation curve are much more and sharper than those in the training curve.

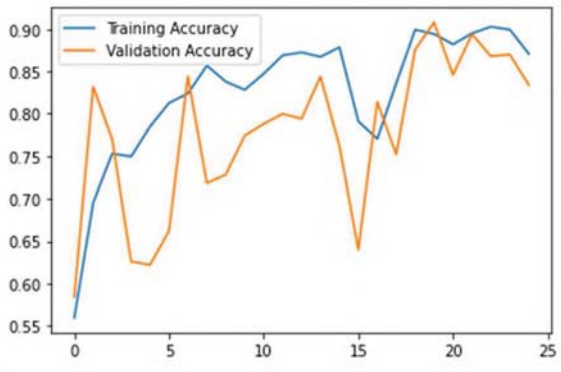

Fig. 6. Accuracy Curve of Inception Model

3) *ResNet50:* The model achieves training accuracy of 56% and validation accuracy of 50%. Steeper variations are seen in validation part as compared to the training part.

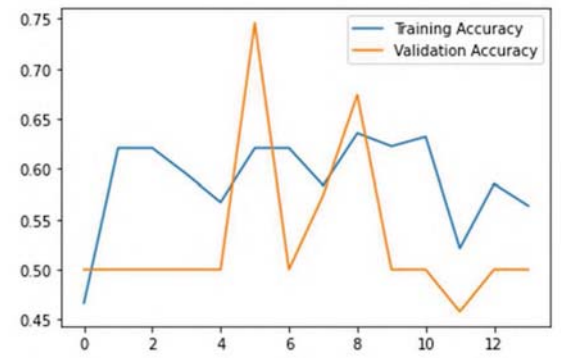

Fig. 7. Accuracy Curve of ResNet50 Model

C. Loss Curves

The loss attained by using different models are plotted against the number of epochs. The x-axis represents the number of epochs and the y-axis represents the number of epochs. All the models are run for 30 epochs.

1) *VGG16:* This line graph represents training and validation loss. Accuracy attained on training the model is 26% and the validation accuracy is 27%. Both the loss curves see steady variations.

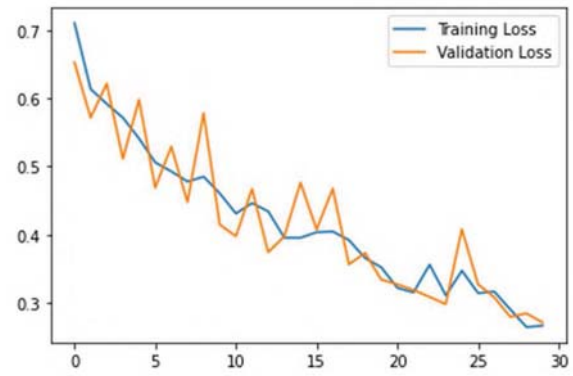

Fig. 8. Loss Curve of VGG16 Model

2) *Inception:* Loss values are examined here. Loss acquired on running the model is 30% and on validating is 34%. Both the loss curves see shaky variations.

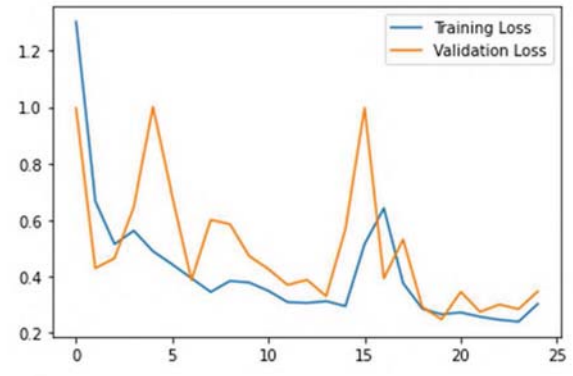

Fig. 9. Loss Curve of Inception Model

3) *ResNet50:* A very large value of loss is observed in both the cases of training and validation which is nearly around 73% and 67.8% independently.

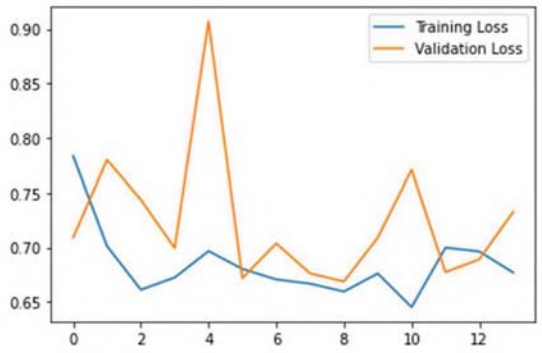

Fig. 10. Loss Curve of ResNet50 Model

V. Data Preprocessing

Data in its raw form cannot be fed to the Deep Learning model since it contains noise or information that is of less importance to the model. Thus, removal of these anomalies is necessary for obtaining proper results. The dataset has been

taken from 'Kaggle' and consists of two classes: alert and drowsy which are determined by the facial expressions. The training set consists of 716 alert images and 437 drowsy images. Data Augmentation has been performed on the dataset for increasing the number of training images using the ImageDataGenerator class. The images are augmented on the fly without making any changes to the actual dataset. For the training images, normalization is performed by dividing the pixels by 255 for scaling the different values of pixels in the same range. Various techniques such as rotation_range, zoom_range, shear_range, width_shift_range, height_shift_range and horizontal_flip is used for augmentation. As a result, a total of 1153 images belonging to 2 classes are obtained. All the images are sized to a dimension of (150, 150, 3). The resultant images are then displayed.

VI. Results and Discussions

We have trained our system using 3 different kinds of deep neural nets in order to know which of these would be a suitable model to be implemented in our system. We have used the pre-trained models namely VGG16, Inception and ResNet50 and had compared their performances. In order to better understand their performance, we have plotted the following graphs:

A. Accuracy Collation

VGG16 has acquired highest accuracy with reasonable number of epochs, as observed. The training accuracy acquired by VGG16, Inception and ResNet50 is about 91%, 89% and 56% respectively. After validation part, VGG16 proves to be best although with fluctuations. The validation accuracy acquired by VGG16, Inception and ResNet50 is about 94%, 83% and 50% respectively.

Fig. 11. Training Accuracy Comparison

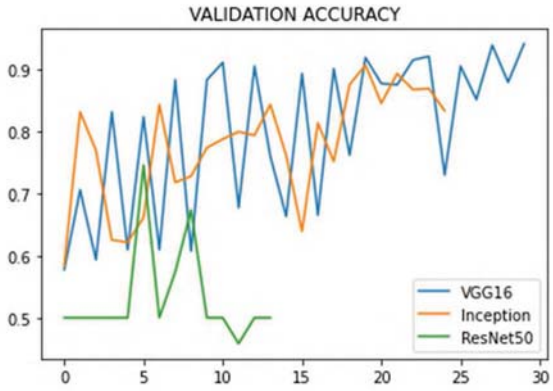

Fig. 12. Validation Accuracy Comparison

B. Loss Collation

VGG16 has achieved a steady loss value without any disturbances. 0.259, 0.303 and 0.678 are the values achieved by VGG16, Inception and ResNet50 individually. VGG16 appears to have lowest loss value, with lots of alterations in the graph. However, validation losses are quite closer as that of the training loss. Hence, after all these considerations and observations we ultimately came to a conclusion to utilize VGG16 as our deep net for proper identification of the condition of the driver.

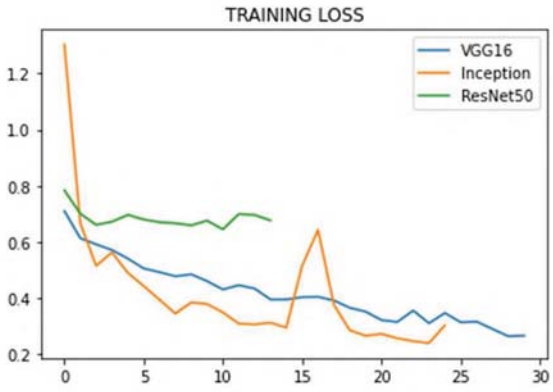

Fig. 13. Training Loss Comparison

Fig. 14. Validation Loss Comparison

C. Expected Results

A full-fledged application built using technologies such as Android and Convolutional Neural Network (CNN), a Deep Learning Algorithm is expected to attain an accuracy of more than 90%. The system would predict the drowsy state of the driver in minimal time and generate an alarm for alerting the driver.

VII. Conclusion

The existing drowsiness detection systems are not widespread and not very common among drivers because of their availability in luxurious vehicles. On one hand, there are systems that use real-footage to detect drowsiness but require expensive set-up and are not very convenient to use. Whereas, there are also systems that are cheaper and easy to use but not very accurate. In this project, our aim is to overcome the limitations of existing systems and inculcate both the convenience of use as well as cost effectiveness in developing a system for Driver Drowsiness Detection.

References

[1] J. Baraniak, J. Hauer, N. Schuhmann, and G. Leugering, "Implementation of Adaptive Filters for Biomedical Applications," presented at the IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering; SIBIRCON 2008. , Novosibirsk, 2008.

[2] Jabbar, R., Al-Khalifa, K., Kharbeche, M., Alhajyaseen, W., Jafari, M. and Jiang, S., 2018. Real-time driver drowsiness detection for android application using deep neural networks techniques. *Procedia computer science*, *130*, pp.400-407.

[3] Deng, W. and Wu, R., 2019. Real-time driver-drowsiness detection system using facial features. *IEEE Access*, *7*, pp.118727-118738.

[4] Babaeian, M., Bhardwaj, N., Esquivel, B. and Mozumdar, M., 2016, November. Real time driver drowsiness detection using a logistic-regression-based machine learning algorithm. In *2016 IEEE Green Energy and Systems Conference (IGSEC)* (pp. 1-6). IEEE.

[5] Dwivedi, K., Biswaranjan, K. and Sethi, A., 2014, February. Drowsy driver detection using representation learning. In *2014 IEEE international advance computing conference (IACC)* (pp. 995-999). IEEE.

[6] Reddy, B., Kim, Y.H., Yun, S., Seo, C. and Jang, J., 2017. Real-time driver drowsiness detection for embedded system using model compression of deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 121-128).

[7] An introduction to Global Average Pooling in convolutional neural networks. Retreived from https://adventuresinmachinelearning.com/global-average-pooling-convolutional-neural-networks/

[8] Convolutional neural network. Retrieved from https://en.wikipedia.org/wiki/Convolutional_neural_network

[9] VGG-16 | CNN Model. Retrived from https://www.geeksforgeeks.org/vgg-16-cnn-model/

[10] Understanding GoogLeNet Model – CNN Architecture. Retrived from https://www.geeksforgeeks.org/understanding-googlenet-model-cnn-architecture/

[11] Residual Networks (ResNet) – Deep Learning. Retrieved from https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/