Volume 10, Issue 3
June 2021

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X



# A Complete Implementation of Driver Monitoring System using Artificial Intelligence

**[1]T. Prabhavathy and [2]S. Krithiga**

[1]PG Student of Applied Electronics, Department of ECE, TPGIT, Vellore, Tamil Nadu, India.

[2]Assistant Professor, Department of ECE, TPGIT, Vellore, Tamil Nadu, India.

E-mail: [1]prabhatamilvanan@gmail.com, [2]krithiga.sri@gmail.com

## ABSTRACT

In recent days the technological trends in the development of AI and ML paved the way to a driver monitoring system. For the past few years, many algorithms and models are proposed for driver drowsiness detection. The behavior-based method is used to determine the behavioral pattern of eye movement based on AI and deep learning. The Haar Cascade classifier and Viola-Jones facial landmark detection are used to detect faces and to extract the Region Of Interest (ROI). The Eye Aspect Ratio (EAR) is used to detect fatigue based upon the relation between the width and height of the eye. Due to the increase in traffic accidents, a proper technique to detect drowsiness is required. Microsleeps are of short duration where the driver has his eyes closed and cannot perceive or react which is a major reason for road accidents. With the development of deep learning technologies, the possibility of detecting driver drowsiness without human intervention can be greatly improved. The high accuracy and feasible results can be obtained with proper construction and optimization of the model. The model has been built to analyze driver behavior especially the ratio of eye-opening for the detection of drowsiness. To achieve a simple model with less computation and more accurate results the CNN model has been built trained and optimized. The trained CNN model has been deployed in both Windows OS as well as Android OS.

**Keywords:** *Driver Monitoring System (DMS), Convolution Neural Network (CNN), Machine Learning (ML), Eye Aspect Ratio (EAR), Region Of Interest (ROI), Hyperparameter Tuning, Windows OS, Android OS*

## 1. INTRODUCTION

The global increase in road accidents reports that 1 in 4 accidents are caused by drowsy driving and 1 in 25 adult drivers report that they have fallen asleep at the wheel in the past 30 days [1]. Driver drowsiness detection system has more potential to prevent sleep-induced road accidents. Researchers have proposed a wide variety of drowsiness detection methods through various kinds of data sources, which can be categorized into physiological methods, vehicle-based measures, subjective measures, and behavior-based methods [2]. The physiological drowsiness detection method requires wearable devices and are bulky. The vehicle behavior based method depends on the capability of the driver to concentrate on the vehicle.

The main drawback of vehicle behavior based method is detection of post-drowsiness and variation based on road geometry. In a subjective measure system, the drivers alert is estimated by the level of sleepiness using a sleepiness scale. The main drawback is the infeasibility of implementation in real-world driving conditions. This paper overcomes the above drawback because it deals with behavior-based drowsiness detection. Here, the eye is considered an important parameter for pre-drowsiness detection. Since the convolution neural network successfully captures the spatial and temporal dependencies in an image with the help of relevant filters, a CNN-based model has been built, trained, optimized, and finally deployed in Windows OS and Android OS. This system provides a cost-effective solution, it doesn't require any additional hardware devices except the mobile phone. Nowadays, mobile has become one of the common thing among everyone, on which the app could be installed and ready for use.

The rest of the paper is structured as follows: Section2 presents related work carried out by researchers, Section3 presents the construction of the training model, Section4 presents hyperparameter optimization, Section5 presents training performance acceleration, Section6 presents testing the proposed model with .csv images, Section7 presents model testing and analysis with images, Section8 presents model deployment on Windows OS, Section9 presents model deployment on Android OS, Section10 presents final results, Section11 presents the future scope and finally concludes with references.

## 2. RELATED WORK

Many researchers, scholars, and crew members of the corporate world are still working in this sector using various image processing, machine learning, and artificial intelligence.

Jabbar et al. [3] has proposed a system that focuses on the detection of microsleep and drowsiness using neural network methodologies. The author obtains the video dataset from NHTU and performs data preprocessing, data augmentation, and feature extraction from those images. It is trained using the D2CNN-FLD model and then implemented using android architecture. The system detects facial landmarks from captured images on a mobile device and passes them to a

Volume 10, Issue 3
June 2021

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X



CNN-based trained Deep Learning model for drowsy driving detection. This model has 83.33% of accuracy.

Vasudevan et al. [4] describe that it is essential to monitor the EAR (Eye Aspect Ratio), steering angle, and acceleration pattern. The author uses OBD (On-Board Diagnostics), machine learning, and image processing for detection of deviation and alerts the driver. The EAR is calculated with the help of six coordinates. When the distance between the opposite points decreases, it detects drowsiness. When the steering angle remains constant for a particular threshold period, it detects the deviation. The author uses the Intel UP2 AI vision board for processing real-time algorithms, Intel UP2 machine vision USB camera for acquiring high-resolution images, and On-Board Diagnostics (OBD) for retrieving data like steering angle. This system requires Machine learning at higher scales and an android application to report feedback.

Han Wei et al. [2] discuss the importance of extracting eye features for drowsiness detection and also describes extracting eye features from low resolution or degraded images. Driver drowsiness detection is based on eyelid movement information. Pre-processing of the image is achieved by SSQ (single scale self quotient image) technique. For eye openness recognition, Orthogonal locality preserving projections (OLPP) and an Extreme learning machine (ELM) are used. Viola-jones framework is used for face and eye detection and proposed a novel method of unsupervised learning. However, the drowsiness detection performance of this method is affected by the limited amount of data.

Chirra et al. [5] use a viola-jones face detection algorithm to detect the face images and it is given as input to viola-jones eye detection algorithm. Once the face is detected, the viola-jones eye algorithm is used to extract the eye region from the facial images and given as input to the CNN. CNN with four convolution layers is used to extract the deep features and those features are passed a fully connected layer. The Soft-max layer in CNN classifies the images into sleepy or non-sleepy images. First, the experiment is performed on the collected dataset, and second, the experiment is performed on video. This proposed system achieves an accuracy of about 96%

Ignatov et al. [6] discusses about deep learning frameworks in the android ecosystem. The TensorFlow mobile, TensorFlow lite, Caffe, and caffe2 are the main deep learning frameworks. In TensorFlow mobile, the TensorFlow's programming model is described as a directed graph and Once the model has been trained, it can be transported as a .pb graph. In TensorFlow lite, the pre-trained model is converted into .tflite. The Caffe and Caffe2 are deep learning frameworks made with expression, speed, and modularity in mind. According to the Caffe2 Github repository, the speed of the mobile library is generally comparable to TensorFlow lite and claims up to higher speed when using the OpenGL backend for GPU computations.

Nirmal Kumar et al. [7] proposes a real-time IR camera-based driver monitoring system. They have collected data from people under different lighting conditions and optimized the model to run on an embedded platform. The technique to reduce the complexity is replacing the convolution layer with a depthwise convolution layer and reducing the number of weights. They have used TensorFlow for training because of its good support for distributed computing, graph visualization tools, faster compilation time than other libraries, and provides both C++ and Python APIs. It also supports an optimized, lightweight inference library (Tflite) which is best suited for embedded platforms.

## 3. PROPOSED CONSTRUCTION OF TRAINING MODEL

Tensor Flow has made the implementation of machine-learning deep-learning models easier. It is an open-source library developed by Google for deep learning applications. The Google Colab is a free cloud service and it also supports free GPU, improves the Python programming language coding skills, and develops deep learning applications using some libraries. To investigate the performance of eye closeness detection in different conditions, a dataset for eye closeness detection in the Wild CEW [Closed Eyes in the wild] is used. In particular, this dataset contains 2874 subjects, consists of both eyes closed and open images which are collected from the Labeled Face in the Wild (LFW) database [8]. Figure 1 represents the CNN training with the processed dataset. Database fields have been converted into a format that contains a single line where a comma separates each database record. The .csv is similar to plain text files, can be shared easily and it reduces time consumption and storage capacity.

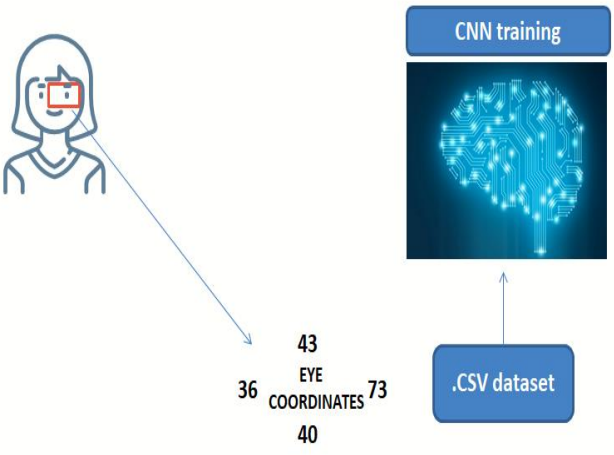



Figure 1: CNN Training with Processed Dataset

The construction of CNN [9] [10] includes exporting the dataset to CNN, preprocessing the dataset, and defining proper layers, neurons, and activation functions. A proper path must be established for exporting the .csv file. It can be read in dictionary format and the list has been created with every row of the file. Two NumPy arrays are created to store the image and tag of the image, to assert to the previous array values. Then the two arrays are shuffled and returned. A proper layer

Volume 10, Issue 3
June 2021

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X



has been defined that is the number of convolution layers, neurons, and activation functions. Initially, some default values are assigned.


Figure 2: Plot between epoch and training loss/accuracy for Epoch = 5

CNN model has been trained with appropriate optimizer and loss function. Training accuracy and training loss has been plotted. Binary cross-entropy loss function has been used in an application for binary class problem open/closed eye. Random CNN neural structure has been called and training has been started to observe the impact on accuracy and loss. Initially, the epoch has been set to 5, training loss and accuracy are observed to be 0.83 and 0.35 as shown in Figure 2. And then epoch value has been set to 10, training loss and accuracy at the first iteration are observed to be 0.88 and 0.38 as shown in Figure 3. Multiple iterations for train runs have been done to observe a gradual increase in training loss and accuracy.

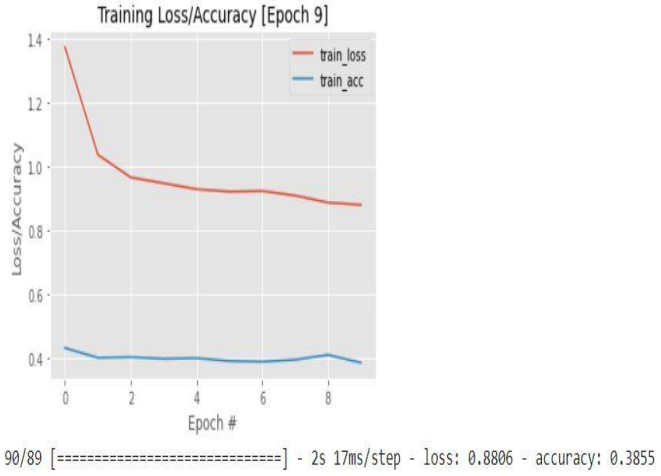



Figure 3: Plot between epoch and training loss/accuracy for Epoch = 10

## 4. HYPERPARAMETER TUNING

The hyperparameter is a parameter from a prior distribution as it is used to capture prior beliefs before data is observed [11]. For any machine learning, these parameters should be initialized before training a model. The model hyperparameters are used to govern the entire training process. Choosing the appropriate parameters plays a vital role in the success of our neural network architecture.

### 4.1 Optimization Of Learning Rate (LR)

Case 1 - On decreasing the learning rate below 0.001, the training loss increases, and accuracy decreases. It seems it slightly suffers from underfitting and thus the model may miss some important patterns in data. Figure 4 represents the impact of training loss on varying the learning rate. When the learning rate is set to 0.001, the training loss is 0.0615. On decreasing the learning rate the training loss increases, for a minimal learning rate of 0.00001, the training loss is very high of about 0.4364. Figure 5 represents the impact on accuracy on varying the learning rate. When the learning rate is set to 0.001, the accuracy is 0.9795. On decreasing the learning rate the accuracy decreases, for a minimal learning rate of 0.00001, the accuracy is very low at about 0.8372.

Case 2 - On increasing the learning rate above 0.001, the training loss increases, and accuracy decreases. It seems it slightly suffers from overfitting and thus there may be some collisions. Figure 6 represents the impact of training loss on varying the learning rate. When the learning rate is set to 0.001, the training loss is 0.0615. On increasing the learning rate the training loss increases, for maximal learning rate of 0.9, the training loss is very high of about 0.8686. Figure 7 represents the impact on accuracy on varying the learning rate. When the learning rate is set to 0.001, the accuracy is 0.9795. On increasing the learning rate the accuracy decreases, for maximal learning rate of 0.9, the accuracy is very low of about 0.5129. Thus, the optimized learning rate is 0.001, identified on the basis of learning rate impact on training loss and accuracy.

| LEARNING RATE | ACCURACY |
|---|---|
| 0.00001 | 0.8372 |
| 0.00002 | 0.8612 |
| 0.00003 | 0.8699 |
| 0.00004 | 0.8987 |
| 0.00005 | 0.8984 |
| 0.00006 | 0.913 |
| 0.00007 | 0.9266 |
| 0.00008 | 0.9287 |
| 0.00009 | 0.9415 |
| 0.0001 | 0.9346 |
| 0.0002 | 0.9631 |
| 0.0003 | 0.9656 |
| 0.0004 | 0.9708 |
| 0.0005 | 0.9732 |
| 0.0006 | 0.9701 |
| 0.0007 | 0.9736 |
| 0.0008 | 0.9722 |
| 0.0009 | 0.9753 |
| 0.001 | 0.9795 |

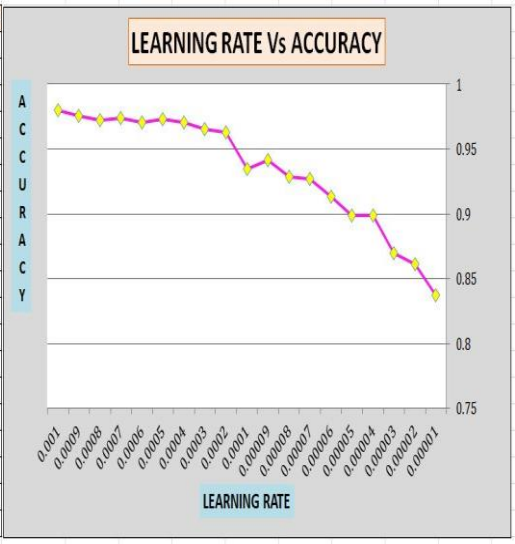



Figure 4: Graph between LR vs. ACCURACY for LR below 0.001

Volume 10, Issue 3
June 2021

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X



| LEARNING RATE | LOSS |
|---|---|
| 0.00001 | 0.4364 |
| 0.00002 | 0.3413 |
| 0.00003 | 0.3108 |
| 0.00004 | 0.2529 |
| 0.00005 | 0.2559 |
| 0.00006 | 0.224 |
| 0.00007 | 0.2043 |
| 0.00008 | 0.1965 |
| 0.00009 | 0.1652 |
| 0.0001 | 0.1733 |
| 0.0002 | 0.1089 |
| 0.0003 | 0.1051 |
| 0.0004 | 0.0937 |
| 0.0005 | 0.085 |
| 0.0006 | 0.0807 |
| 0.0007 | 0.0748 |
| 0.0008 | 0.0849 |
| 0.0009 | 0.071 |
| 0.001 | 0.0615 |

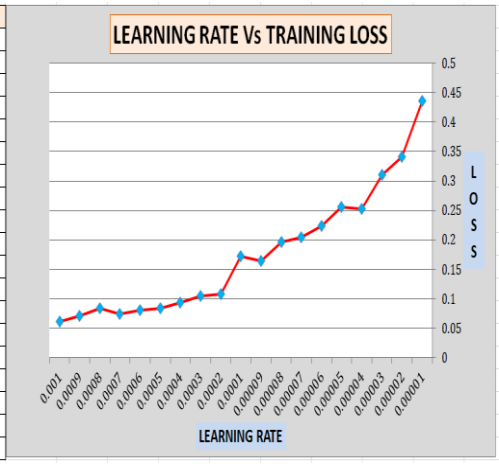



Figure 5: Graph between LR vs. TRAINING LOSS for LR below 0.001

| LEARNING RATE | LOSS |
|---|---|
| 0.001 | 0.0615 |
| 0.002 | 0.0742 |
| 0.003 | 0.0897 |
| 0.004 | 0.1412 |
| 0.005 | 0.1592 |
| 0.006 | 0.1582 |
| 0.007 | 0.1802 |
| 0.008 | 0.1905 |
| 0.009 | 0.2064 |
| 0.01 | 0.2334 |
| 0.02 | 0.3828 |
| 0.03 | 0.4127 |
| 0.04 | 0.4906 |
| 0.05 | 0.6937 |
| 0.06 | 0.693 |
| 0.07 | 0.6933 |
| 0.08 | 0.6969 |
| 0.09 | 0.6995 |
| 0.1 | 0.6934 |
| 0.2 | 0.7258 |
| 0.3 | 0.745 |
| 0.4 | 0.8023 |
| 0.5 | 0.7061 |
| 0.6 | 0.8026 |
| 0.7 | 0.6949 |
| 0.8 | 5.7818 |
| 0.9 | 0.8686 |

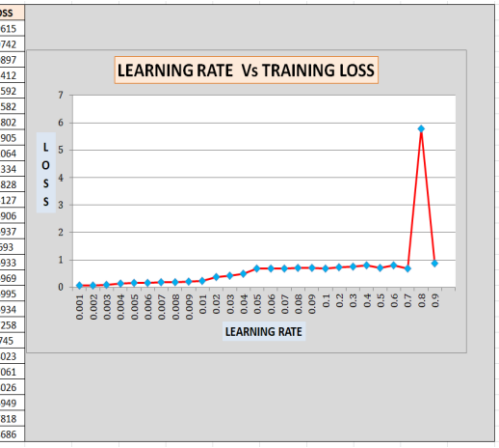



Figure 6: Graph between LR vs. ACCURACY for LR above 0.001

| LEARNING RATE | ACCURACY |
|---|---|
| 0.001 | 0.9795 |
| 0.002 | 0.977 |
| 0.003 | 0.9708 |
| 0.004 | 0.9516 |
| 0.005 | 0.9447 |
| 0.006 | 0.9388 |
| 0.007 | 0.9388 |
| 0.008 | 0.9266 |
| 0.009 | 0.9241 |
| 0.01 | 0.9148 |
| 0.02 | 0.8271 |
| 0.03 | 0.8232 |
| 0.04 | 0.7738 |
| 0.05 | 0.5129 |
| 0.06 | 0.5108 |
| 0.07 | 0.5122 |
| 0.08 | 0.5031 |
| 0.09 | 0.5038 |
| 0.1 | 0.5205 |
| 0.2 | 0.4955 |
| 0.3 | 0.5115 |
| 0.4 | 0.5216 |
| 0.5 | 0.4983 |
| 0.6 | 0.5045 |
| 0.7 | 0.5115 |
| 0.8 | 0.4889 |
| 0.9 | 0.5129 |

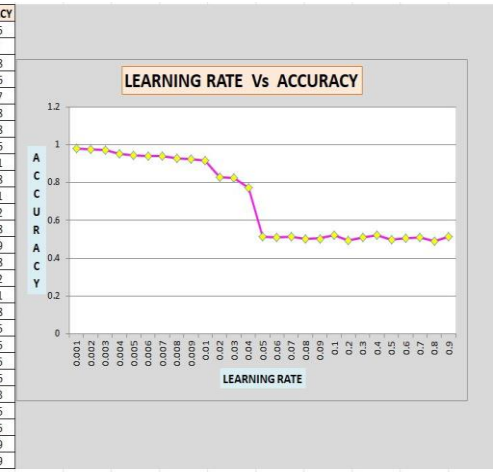



Figure 7: Graph between LR vs. TRAINING LOSS for LR above 0.001

### 4.2 Epoch Optimization

An epoch represents the number of cycles, the dataset is passed forward and backward through the neural network. As the number of epochs increases or decreases, more or less the weight changes in the neural network and the curve goes from underfitting to optimal to overfitting curve. The dataset of 2874 is divided into a batch size of 32 then it will take 89 to 90 iterations for one epoch. If the epoch is set to 5, then it starts from epoch1 to epoch5. Below are the graph plotted epochs versus loss and epochs versus accuracy. On changing the number of epochs, optimized epoch number is found based upon training loss and accuracy.

When epoch is set to 5, it slightly suffers from under fitting as shown in Figure 8. When Epoch is set to 8, loss slightly decreases as shown in Figure 9. When Epoch is set to 10, loss decreases, and accuracy increases. The plot seems to be the optimal one as shown in Figure 10. When Epoch is set to 12, loss increases and accuracy decreases when compared to the previous one as shown in Figure 11. When Epoch is set to 15, loss decreases but slightly suffers from overfitting. Thus, the optimum epoch number is 10, identified on the basis of epoch impact on training loss and accuracy.

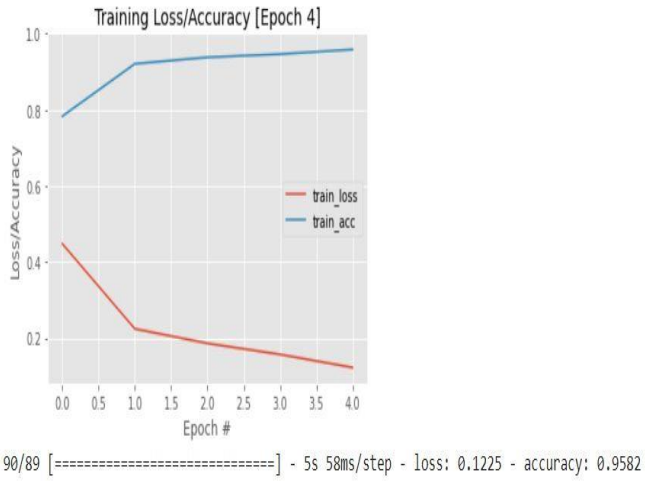



Figure 8: Effect of setting Epoch to the value 5

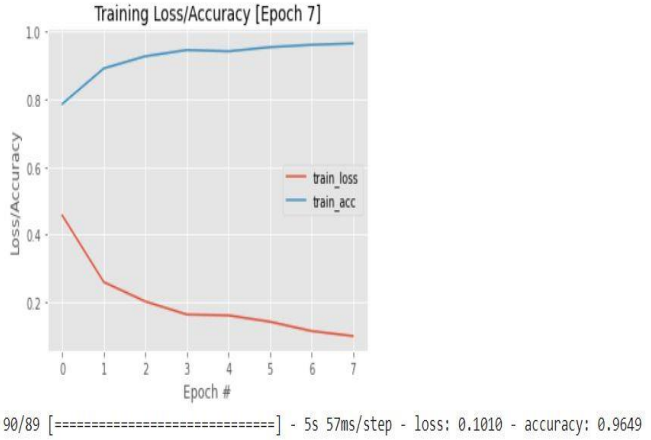


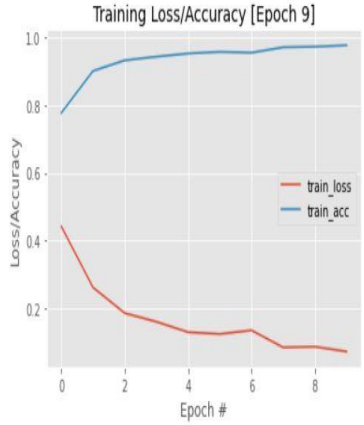

Figure 9: Effect of setting Epoch to the value 8

Volume 10, Issue 3
June 2021

ITEE Journal
Information Technology & Electrical Engineering

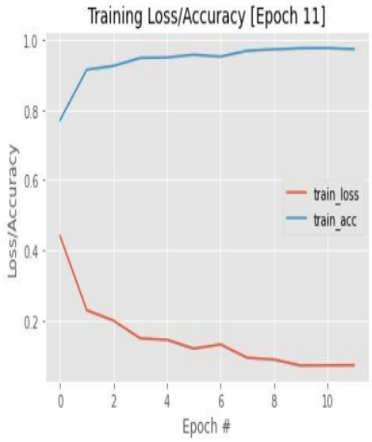
ISSN: - 2306-708X





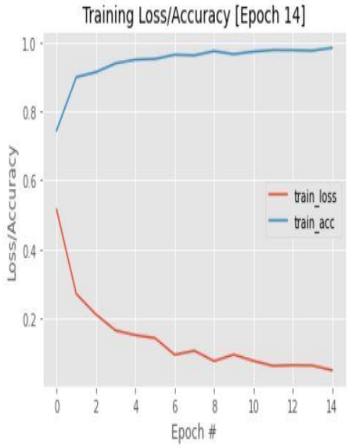

```
90/89 [==============================] - 5s 58ms/step - loss: 0.0718 - accuracy: 0.9774
```

Figure 10: Effect of setting Epoch to the value 10




```
90/89 [==============================] - 5s 57ms/step - loss: 0.0725 - accuracy: 0.9739
```

Figure 11: Effect of setting Epoch to the value 12




```
90/89 [==============================] - 5s 57ms/step - loss: 0.0490 - accuracy: 0.9843
```

Figure 12: Effect of setting Epoch to the value 15

### 4.3 Selection Of Optimizer

The optimizers are used to update the weight parameters in order to reduce the loss function. Stochastic Gradient Descent (SGD) divides N samples into equal-sized batches, update weights once per batch and one epoch completes when all samples are used once. SGD is much faster than the other optimizers but results are a little far from optimum. The effect of SGD on loss and accuracy is shown in Figure 13. Adagrad really works well for sparse datasets. It produces better results compared to SGD as shown in Figure 14 but they are computationally extensive. Adam was slightly faster than Adagrad and produces less training loss compared to other methods as shown in Figure 15. Thus, the Adam optimization function provides a better trade-off between more computation power and more optimum results.

```
2874
Epoch 1/10
90/89 [==============================] - 5s 54ms/step - loss: 0.6932 - accuracy: 0.4756
Epoch 2/10
90/89 [==============================] - 5s 55ms/step - loss: 0.6925 - accuracy: 0.4847
Epoch 3/10
90/89 [==============================] - 5s 56ms/step - loss: 0.6919 - accuracy: 0.5174
Epoch 4/10
90/89 [==============================] - 5s 56ms/step - loss: 0.6910 - accuracy: 0.5511
Epoch 5/10
90/89 [==============================] - 5s 56ms/step - loss: 0.6907 - accuracy: 0.5623
Epoch 6/10
90/89 [==============================] - 5s 56ms/step - loss: 0.6900 - accuracy: 0.5839
Epoch 7/10
90/89 [==============================] - 5s 56ms/step - loss: 0.6890 - accuracy: 0.5981
Epoch 8/10
90/89 [==============================] - 5s 56ms/step - loss: 0.6883 - accuracy: 0.6124
Epoch 9/10
90/89 [==============================] - 5s 56ms/step - loss: 0.6880 - accuracy: 0.6423
Epoch 10/10
90/89 [==============================] - 5s 56ms/step - loss: 0.6875 - accuracy: 0.6374
```

Figure 13: Effect of SGD Optimizer

```
2874
Epoch 1/10
90/89 [==============================] - 5s 58ms/step - loss: 0.6930 - accuracy: 0.5355
Epoch 2/10
90/89 [==============================] - 5s 58ms/step - loss: 0.6923 - accuracy: 0.5790
Epoch 3/10
90/89 [==============================] - 5s 58ms/step - loss: 0.6910 - accuracy: 0.6395
Epoch 4/10
90/89 [==============================] - 5s 58ms/step - loss: 0.6902 - accuracy: 0.6778
Epoch 5/10
90/89 [==============================] - 5s 58ms/step - loss: 0.6890 - accuracy: 0.6893
Epoch 6/10
90/89 [==============================] - 5s 58ms/step - loss: 0.6877 - accuracy: 0.7143
Epoch 7/10
90/89 [==============================] - 5s 58ms/step - loss: 0.6866 - accuracy: 0.7237
Epoch 8/10
90/89 [==============================] - 5s 58ms/step - loss: 0.6851 - accuracy: 0.7213
Epoch 9/10
90/89 [==============================] - 6s 64ms/step - loss: 0.6832 - accuracy: 0.7258
Epoch 10/10
90/89 [==============================] - 5s 57ms/step - loss: 0.6811 - accuracy: 0.7196
```

Figure 14: Effect of Adagrad Optimizer

Volume 10, Issue 3
June 2021

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X



```
2874
Epoch 1/10
90/89 [==============================] - 5s 57ms/step - loss: 0.4507 - accuracy: 0.7871
Epoch 2/10
90/89 [==============================] - 5s 58ms/step - loss: 0.2346 - accuracy: 0.9099
Epoch 3/10
90/89 [==============================] - 5s 58ms/step - loss: 0.1750 - accuracy: 0.9384
Epoch 4/10
90/89 [==============================] - 5s 58ms/step - loss: 0.1640 - accuracy: 0.9419
Epoch 5/10
90/89 [==============================] - 5s 58ms/step - loss: 0.1367 - accuracy: 0.9513
Epoch 6/10
90/89 [==============================] - 5s 58ms/step - loss: 0.1141 - accuracy: 0.9624
Epoch 7/10
90/89 [==============================] - 5s 58ms/step - loss: 0.1028 - accuracy: 0.9656
Epoch 8/10
90/89 [==============================] - 5s 58ms/step - loss: 0.1093 - accuracy: 0.9635
Epoch 9/10
90/89 [==============================] - 5s 58ms/step - loss: 0.0820 - accuracy: 0.9711
Epoch 10/10
90/89 [==============================] - 5s 58ms/step - loss: 0.0634 - accuracy: 0.9774
```

Figure 15: Effect of Adam Optimizer

### 4.4 Optimization Of Neurons

The impact of neuron/node count in convolution layers on accuracy and loss has been analyzed to determine the optimum neuron count for the CNN model as shown in Table 1. Based upon the observed accuracy and losses, the number of neurons in the respective convolution layer as follows,

Convolution layer 1 → (20 min – 40 max)
Convolution layer 2 → (40 min – 80 max)
Convolution layer 3 → (80 min – 160 max)

Table 1: Optimizing the Neurons

| NUMBER OF NEURONS | | | LOSS | ACC |
|---|---|---|---|---|
| CONV L1 | CONV L2 | CONV L3 | | |
| 5 | 5 | 5 | 0.3244 | 0.8715 |
| 10 | 5 | 5 | 0.2486 | 0.9019 |
| 20 | 5 | 5 | 0.2538 | 0.9015 |
| 20 | 10 | 5 | 0.2618 | 0.8968 |
| 20 | 20 | 5 | 0.2308 | 0.9148 |
| 20 | 30 | 5 | 0.1812 | 0.9290 |
| 20 | 40 | 5 | 0.3129 | 0.8702 |
| 20 | 40 | 10 | 0.1983 | 0.9290 |
| 20 | 40 | 30 | 0.1313 | 0.9516 |
| 20 | 40 | 60 | 0.1328 | 0.9576 |
| 20 | 40 | 80 | 0.1103 | 0.9607 |
| 20 | 40 | 100 | 0.0943 | 0.9704 |
| 30 | 40 | 100 | 0.0912 | 0.9666 |
| 30 | 60 | 100 | 0.0876 | 0.9649 |
| 30 | 60 | 110 | 0.0972 | 0.9759 |
| 30 | 60 | 120 | 0.0757 | 0.9732 |
| 40 | 60 | 120 | 0.0900 | 0.9659 |
| 40 | 80 | 120 | 0.0961 | 0.9687 |
| 40 | 80 | 140 | 0.1043 | 0.9628 |
| 40 | 80 | 160 | 0.0837 | 0.9711 |

### 4.5 Dropout Rate Optimization

Dropout is dropping out units in a neural network. Dropout is a technique used to prevent overfitting and co-adaptations of neurons by setting the output of any neuron to zero with some probability. Optimizing the dropout rate is shown in Table 2. Thus, increasing the dropout beyond 0.10 results in a decrease in accuracy, and decreasing the dropout below 0.10 results in saturation of accuracy. Hence, the optimum dropout rate is 0.10 for hidden and visible units.

### 4.6 Batch Size Optimization

Batch size is the number of training examples going to be used in one iteration. The size of the update is dependent on the particular samples that are drawn from the dataset. The larger batch sizes can improve per image processing speed but it may lead to lower asymptotic accuracy. Smaller batch sizes are easier to fit one batch worth of training data in memory but it may be slightly noisy. Thus, the optimum batch size is required. Thus, the optimum batch size is 32 from Table 3.

Table 2: Optimizing the Dropout Rate

| DROPOUT IN VISIBLE UNITS | DROPOUT IN HIDDEN UNITS | LOSS | ACC |
|---|---|---|---|
| 0.10 | 0.10 | 0.1306 | 0.9506 |
| 0.10 | 0.20 | 0.1471 | 0.9443 |
| 0.10 | 0.30 | 0.1261 | 0.9523 |
| 0.10 | 0.40 | 0.1573 | 0.9388 |
| 0.10 | 0.50 | 0.1839 | 0.9308 |
| 0.10 | 0.60 | 0.2084 | 0.9189 |
| 0.10 | 0.70 | 0.2411 | 0.8960 |
| 0.20 | 0.10 | 0.1186 | 0.9576 |
| 0.20 | 0.20 | 0.1605 | 0.9408 |
| 0.20 | 0.30 | 0.1627 | 0.9408 |
| 0.20 | 0.40 | 0.1968 | 0.9214 |
| 0.20 | 0.50 | 0.1983 | 0.9269 |
| 0.20 | 0.60 | 0.2371 | 0.9033 |
| 0.30 | 0.10 | 0.1642 | 0.9360 |
| 0.30 | 0.20 | 0.1674 | 0.9367 |
| 0.30 | 0.30 | 0.1639 | 0.9395 |
| 0.30 | 0.40 | 0.1843 | 0.9217 |
| 0.05 | 0.1 | 0.1078 | 0.9631 |
| 0.05 | 0.05 | 0.1032 | 0.9631 |

Volume 10, Issue 3
June 2021

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X



Table 3: Optimizing the Batch Size

| BATCH SIZE | LOSS | ACCURACY |
|---|---|---|
| 10 | 0.2411 | 0.9222 |
| 20 | 0.1461 | 0.9465 |
| 32 | 0.1170 | 0.9607 |
| 40 | 0.1317 | 0.9572 |
| 50 | 0.1531 | 0.9391 |
| 64 | 0.1763 | 0.9346 |

### 4.7 Selection Of Activation Function

Activation functions are non-linear function. It is used to model the input-output relationship using very complex functions. These functions have a major effect on the neural network's ability to converge and the convergence speed.

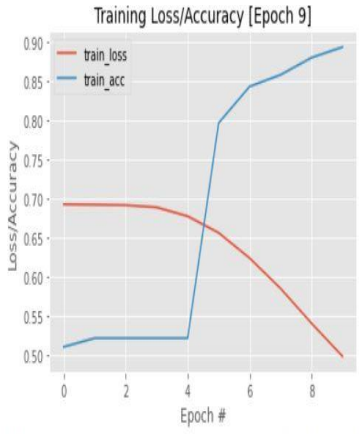



Figure 16: Effect of Softmax Function

Softmax function normalizes the outputs for each class between 0 and 1. The effect of the Softmax Function is shown in Figure 16. Tanh function makes it easier to model inputs that have strongly negative, neutral, and strongly positive values. The effect of the Tanh Function is shown in Figure 17. The effect of the Sigmoid function provides a smooth gradient, the output values bound between 0 and 1. The effect of the Sigmoid Function is shown in Figure 18. ReLU has a derivative function and allows for backpropagation. The effect of the ReLU Function is shown in Figure 19. Thus, the ReLU action function is used for the input and hidden layers. The output layer consists of a sigmoid function to predict the probability that exists between the range 0 and 1.

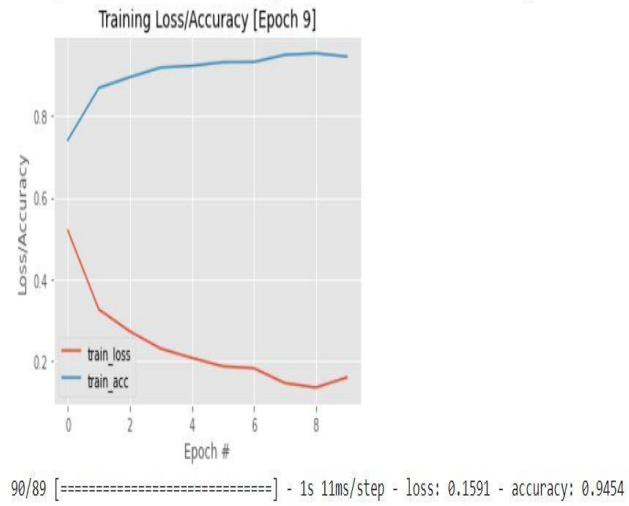



Figure 17: Effect of Tanh Function

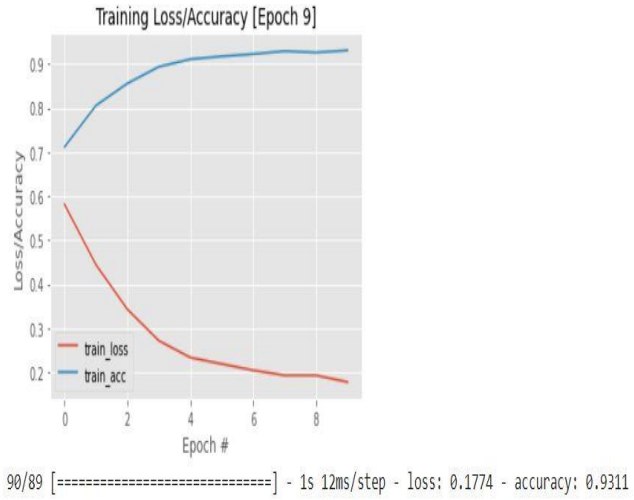



Figure 18: Effect of Sigmoid Function

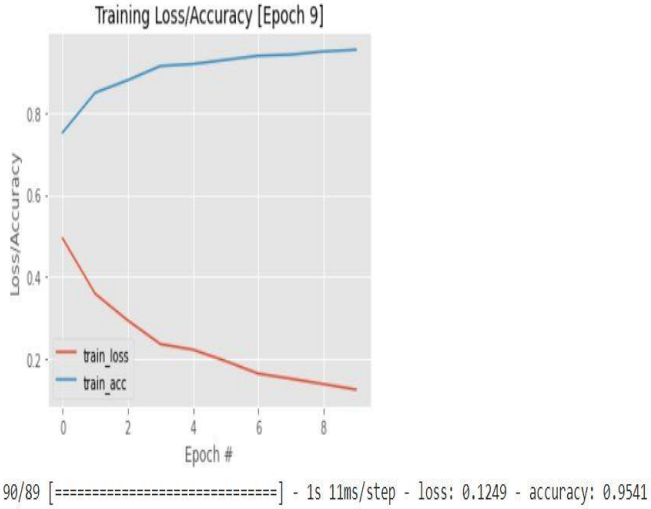



Figure 19: Effect of ReLU Function

Volume 10, Issue 3
June 2021

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X



## 5. TRAINING PERFORMANCE ACCELERATION

There are three types of runtime provided by Colab. They are CPU, GPU, and TPU. The CPU provides fewer cores, low latency but good for serial processing. Figure 20 represents the effect on performance when connecting the CPU. On connecting to GPU, tensor core accelerates large matrix operation. Perform mixed-precision matrix multiplies and accumulates calculations in a single operation. Accelerates traditional AI tasks due to parallel processing. Figure 21 represents the effect on performance when connecting to GPU. TPU consumes more training time when compared to GPU. But TPU is good for bulky datasets and greater batch size. Figure 22 represents the effect on performance when connecting to TPU.

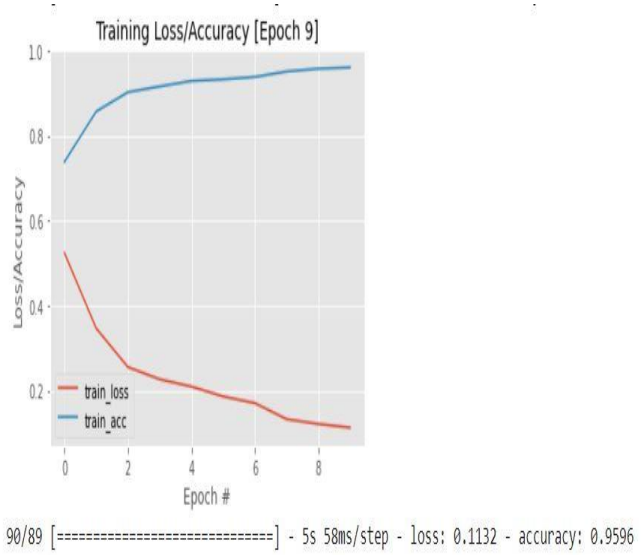



Figure 20: Connecting to CPU

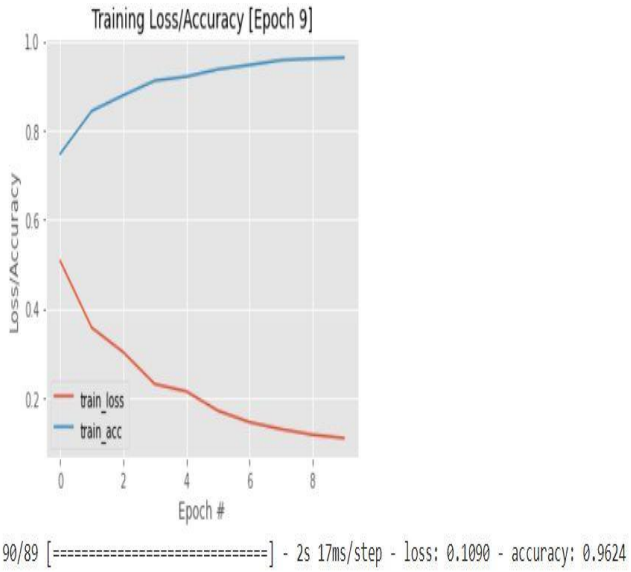



Figure 21: Connecting to GPU

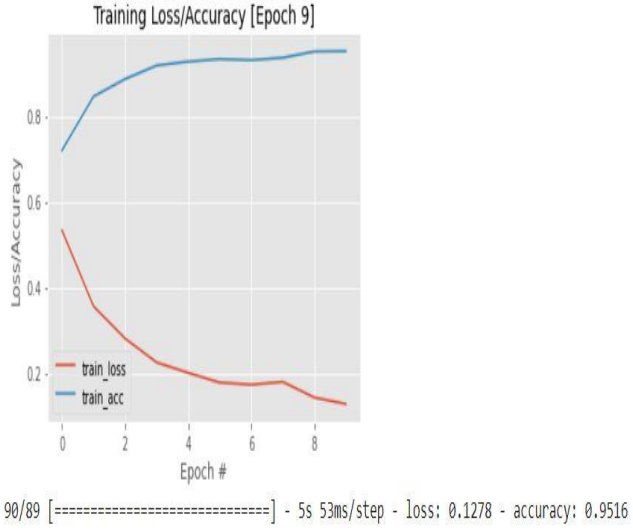



Figure 22: Connecting to TPU

## 6. EVALUATION WITH .CSV IMAGES

The model is evaluated with a test dataset, which consists of 100 eye images in a .csv file. The first step in designing the evaluation model is to load the model, which is needed to be evaluated. Second, resize the image if needed. On giving a .csv eye image to the newly optimized model as shown in Figure 23, the model predicts whether the eyes are closed or open accurately. The predicted results are shown in Figure 24 and Figure 25.

| OPTIMIZED CNN MODEL |
|---|
| CONVOLUTION LAYER |
| Number of Layers -3<br>Padding – same, Filter size 3x3<br>Nodes – 30(L1), 60(L2), 120(L3)<br>Activation - ReLU |
| POOLING LAYER |
| Number of Layers -3<br>Type: Max pooling<br>Pooling size (2,2), Dropout – 0.10 |
| FULLY CONNECTED LAYER |
| Number of Layers – Flatten(1), dense(3)<br>Activation – ReLU, sigmoid<br>Nodes – 512(dense1), 512(dense2), 1(dense3) |
| Optimizer – Adam<br>Loss – binary cross entropy |

Figure 23: Optimized Model

Volume 10, Issue 3
June 2021

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X



```
Total params: 1,037,201
Trainable params: 1,037,201
Non-trainable params: 0
_________________________________________________________________
Enter the eye parameters:[77, 79, 80, 79, 8
WARNING:tensorflow:11 out of the last 11 ca
[1.]
The given image is Open eye
```

Figure 24: Predicting the Open Eye Image

```
Total params: 1,037,201
Trainable params: 1,037,201
Non-trainable params: 0
_________________________________________________________________
Enter the eye parameters:[99, 122, 15
WARNING:tensorflow:11 out of the last
[0.]
The given image is Closed eye
```

Figure 25: Predicting the Closed Eye Image

## 7. TESTING MODEL WITH EYE IMAGES

The trained TensorFlow model for open eye and closed eye detection has been tested with different images. For this testing, the images from various lighting conditions and different views are given. This analysis is required to check the detection ability of the trained model under different conditions. For testing the model various open eye and closed eye images with different views and illumination have been downloaded. The new Colab notebook has been opened and mounted with the drive for accessing the images from the drive. Once the image has been accessed, its corresponding parameters link height, width and mode have been obtained. The image is made with indivisible segments called pixels and every pixel has its strength called pixel intensity. The RGB image is a combination of three and produces all possible color pallets. With color images, a huge volume of data needs to be worked out and becomes computationally expensive. But the greyscale image consists of only one channel and thus the image is converted into a greyscale image. The method flatten() is used to convert the array into a one-dimensional array and the values are taken row-wise.

The image is read as NumPy array and the image processing operations can be performed without the use of OpenCV libraries. The NumPy is an alternative for lists in python because it holds less memory, faster processing, and more convenient to use. Arrays in Numpy are more compact when compared to lists and the data specification which leads to code optimization. The image is converted into greyscale with the convert('L') and then passes it to np.array(), it returns the 2-dimensional (ndarray) whose shape is (row(height), col(width)). The ndarray has been obtained from the PIL image with np. asarray. The np. asarray returns a non – rewritable ndarray. The element in the ndarray is the object of a datatype object (called datatype). The datatype of the read ndarray is an 8-bit unsigned integer and processed as a floating-point number with the help of astype(). The .reshape(x,y) converts an array into a multi-dimensional array. Figure 26 represents the flow of the process involved in this testing and analysis. For this analysis, 8 open eye images and 8 closed eye images have been given to the model for testing. These 16 images are the input images. The preprocessing represents the conversion of images into greyscale and flattening it. The CNN model represents the model that has been built, trained, and optimized. This CNN model consists of three convolution layers and three dense layers. It also includes adam optimizer and binary cross-entropy loss function. The output represents the capability of the CNN model to predict whether the image is a closed eye or an open eye image.

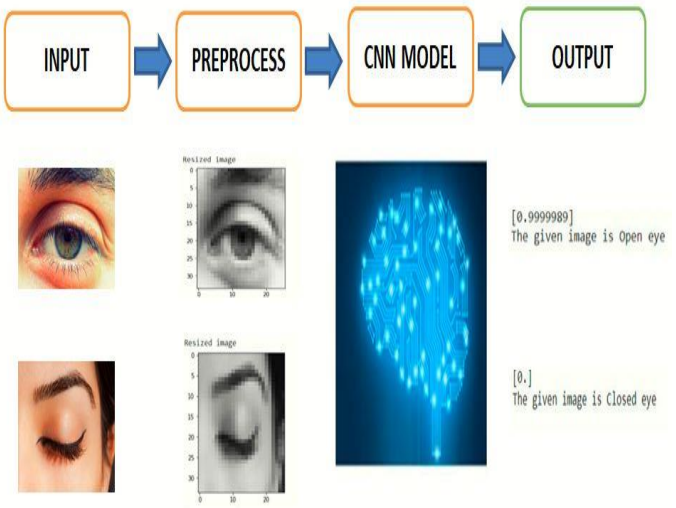



Figure 26: Process involved in testing the model with eye

The confusion matrix [12] is the table used to describe the performance of the classifier or classification model on a set of test data for which the true values are known. It represents the way in which the classification model gets confused in making predictions. Thus, a confusion matrix is a summary of prediction results on a classification problem. The confusion matrix includes true positive (TP), true negative (TN), false positive (FP), and false-negative (FN).

The predictions are the numbers that are organized into a table or matrix. The "expected down the side" represents each row of the matrix corresponds to a predicted class. The "predicted across the top" represents each column of the matrix corresponds to an actual class. The total number of correct predictions of a class moves into the expected row for

Volume 10, Issue 3
June 2021

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X



that class value and the prediction column for that class value. Similarly, the total number of incorrect predictions for the class moves into the expected row for that class value and the predicted column for that class value. The two-class problem is a special problem used to discriminate between observations with a specific outcome from normal observations. The "true positive" represents correctly predicted event values, the "true negative" represents correctly predicted no-event values, the "false positive" represents incorrectly predicted event values, and "false negative" represents incorrectly predicted no-event values.

Table 4: Prediction Truth Table

| PREDICTION TRUTH TABLE | | | | |
|---|---|---|---|---|
| No. images | True Positive (TP) | True Negative (TN) | False Positive (FP) | False Negative (FN) |
| 15 | 7 | 8 | NIL | 1 |

The input image includes eight open-eye images and 8 closed-eye images. The open eye images are considered as true positive and closed eye images are considered as true negative, if it is predicted correctly. If an open eye image is wrongly predicted as closed-eye by the model, it is a type I error and called a false negative. If a closed eye image is wrongly predicted as the open eye by the model, it is a type II error and called a false positive.

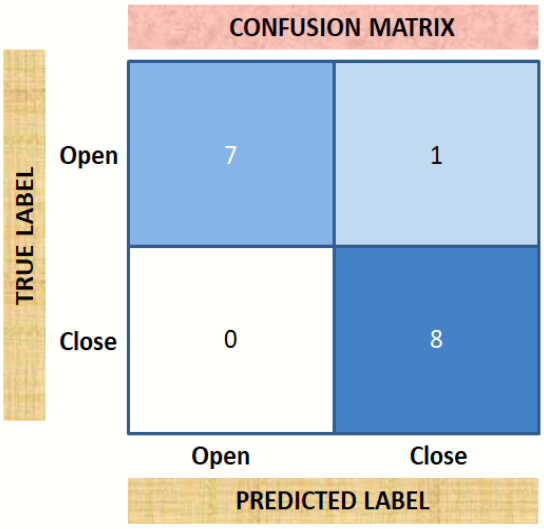



Figure 27: Confusion Matrix for Open eye and Closed eye

The trained and optimized CNN model predicts all the eight closed-eye images correctly but it fails to predict one dark open-eye image. The number of images that the model fails to predict the open-eye image is called a false negative. The number of images that the model fails to predict the closed eye images is called a false positive. The trained and optimized CNN model doesn't wrongly predict the closed eye image, thus there is no false positive. In order to determine the prediction capability of the model, accuracy has been calculated.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$= \frac{7 + 8}{7 + 8 + 0 + 1} = \frac{15}{16}$$

$$= 0.9375$$

$$\% \ Accuracy = 93.75\%$$

Thus, an accuracy of 93.75% is achieved from the trained and optimized model.

## 8. MODEL DEPLOYMENT ON WINDOWS OS

Many ML developers suggest that google colab is the smartest option for training and testing the TensorFlow model. But working with OpenCV in GoogleColab is difficult because it is very difficult to access the local hardware like webcam and the delay between each frame is very high. Thus, many ML developers suggest offline tools like an anaconda to overcome these drawbacks.

OpenCV is a library that carries out image processing using languages like python. OpenCV Library has been utilized to make Real-Time Face Detection using a webcam as a primary camera. The dlib is an open-source library for implementing various machine learning algorithms. The haar cascade frontal face detector is used for face detection, where the cascade function is already trained with lots of positive and negative images. For detecting key facial features, facial landmark detectors are used. The process flow involved in model deployment on windows os is shown in Figure 28.

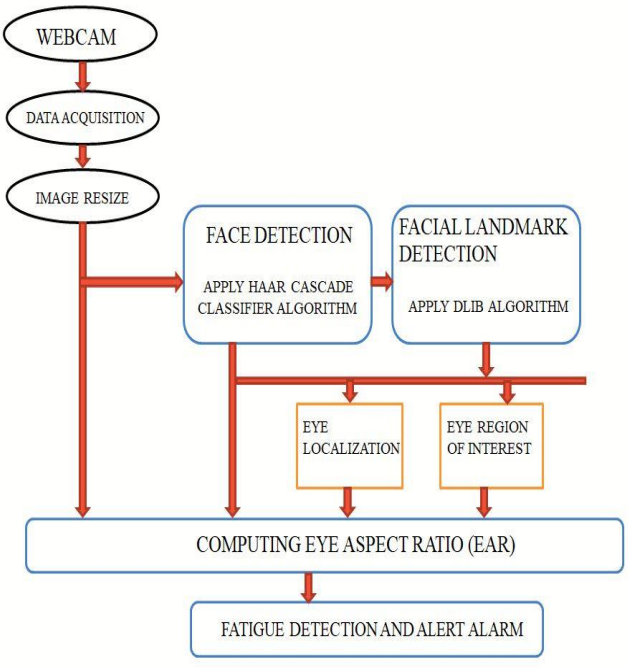



Figure 28: Process Flow involved in Model Deployment

Volume 10, Issue 3
June 2021

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X



From the landmarks detected in the image, the eye aspect ratio (EAR) [13] is used in the estimation of the eye-opening state. For every video frame, eye landmarks are detected. The ratio between the height and width of the eye has been computed. The P1, P2,…, P6 are the landmarks of the eye as shown in Figure 29 and Figure 30.

$$EAR = \frac{||P2-P6|| + ||P3-P5||}{2||P1-P4||}$$

Where,

P1-P4 →Horizontal distance
P2-P6 →Vertical distance1
P3-P5 →Vertical distance2

These landmarks [14] are used to compute the ratio of vertical and horizontal distances. The first important step in EAR computation is to perform facial landmark detection for localization of eyes in a given frame from the video stream. Once the facial landmarks for both eyes have been detected, the corresponding EAR for each eye will be calculated [15]. Thus, the EAR is the singular value that relates the distances between the vertical eye landmark points to the distances between the horizontal landmark points [16].

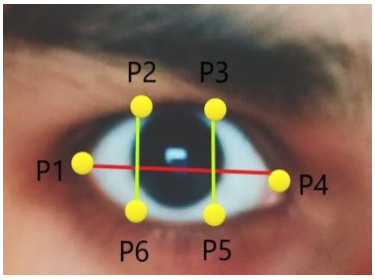



Figure 29: Landmarks on Open eye

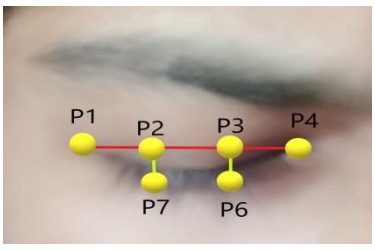



Figure 30: Landmarks on Closed eye

The eye blink lasts approximately 100-400ms. But drowsiness eye blink may last for 1-3s. So, the monitoring of EAR takes place to monitor a blink of an eye whether a drowsy blink of an eye takes place or not. Thus, EAR plays an important role in the drowsiness detection of the eye [17]. The audio warning runs when the fatigue, drowsiness of the driver is detected. The alarm runs and thus alerts the driver. The model has been tested under different conditions. The results which are obtained without wearing spectacles are shown in Figure 31 and Figure 32.

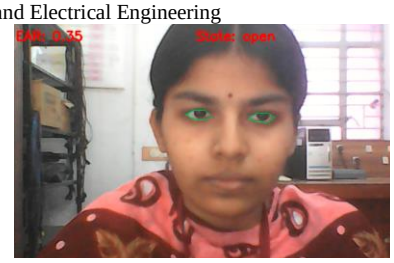


Figure 31: Open eye result – without spectacles

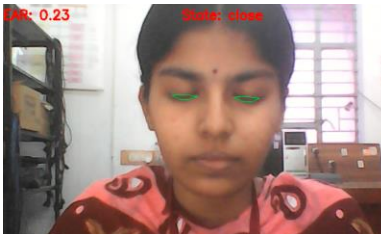



Figure 32: Closed eye result – without spectacles

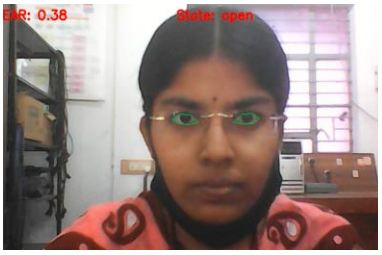



Figure 33: Open eye result – with spectacles

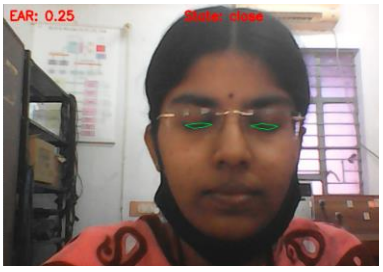



Figure 34: Closed eye result – with spectacles

The results which are obtained with spectacles are shown in Figure 33 and Figure 34. The results which are obtained on focusing light in front are shown in Figure 35 and Figure 36.


Figure 35: Open eye result – Focusing light in front

Volume 10, Issue 3
June 2021

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X




Figure 36: Closed eye result – Focusing light in front

The results which are obtained with varying distances are shown in Figure 37 and Figure 38.


Figure 37: Open eye result with varying distances

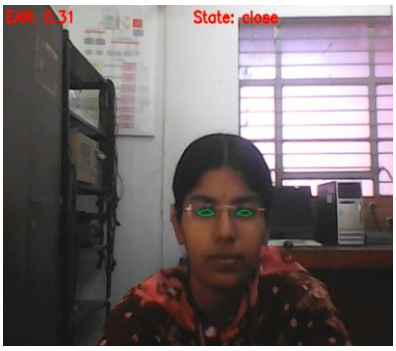



Figure 38: Closed eye result with varying distances

## 9. MODEL DEPLOYMENT ON ANDROID OS

Firebase ML Kit is a google machine learning mobile SDK for Android and IOS [18]. It is a straightforward way of implementing the machine learning functionality with few lines of code and also it provides space to import custom TensorFlow Lite models in the mobile apps. ML kit needs the TensorFlow lite model as input to be implemented in the mobile app. Hence the TensorFlow model must be converted to the TensorFlow Lite model. Tensor flow lite converter has been used here for the conversion process. The tensor flow converter is a python API tool to convert the tensor flow model to the TensorFlow lite model format. The converted TensorFlow lite model is in reduced size of the file and thus it has been deployed in the mobile app easily. The converted TensorFlow lite model is imported into the firebase ML kit. The ML kit supports many machine learning features, the face detection feature from Vision processing has been used.

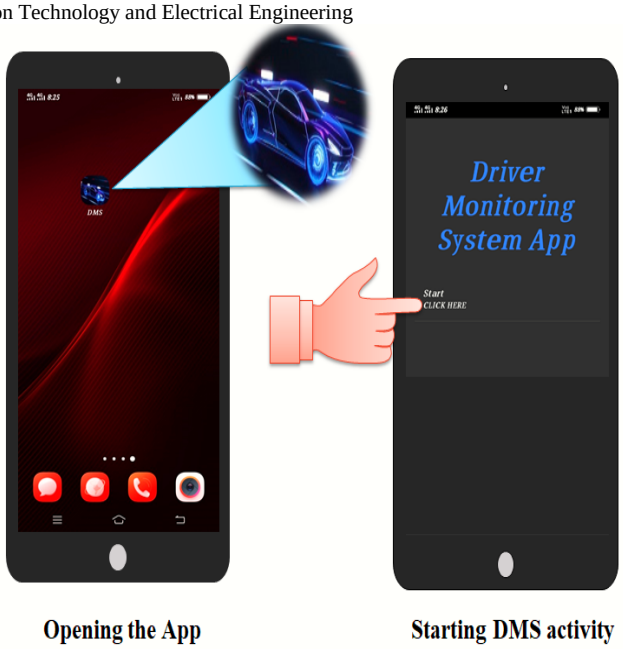



Figure 39: Opening the app and starting the DMS activity

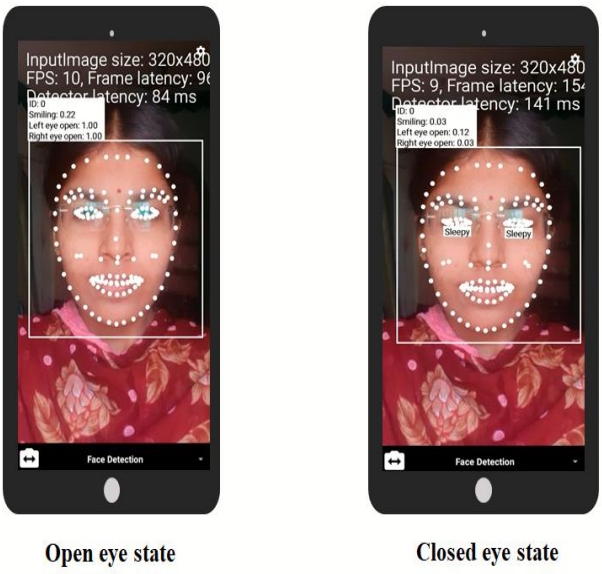



Figure 40: Model deployment in Android OS

The face detection API of the firebase ML kit provides the following features such as face detection, facial features, and contour extraction. Video frames from primary or secondary cameras are fed as an input based upon the selection. Face detection API detects the face and shows the bounding box. Also, it detects facial features like an eye in this scenario. The contour detection extracts the facial eye contours. Based upon this probability from the left and right eye, the eye-opening has been determined. If the probability of right and left eye-opening mean is lesser than 3.0 the closeness of the eye has been detected and thus the app provides an audio alert as shown in Figure 39 and Figure 40.

## 10. FINAL RESULTS

Thus, the CNN model that has been constructed and trained reaches 96.24% accuracy after optimization. This model works well with both .csv and jpg images. For 16 jpg images, this model has produced 93.75% depicted from the confusion matrix. Unlike other models, deploying this trained and optimized CNN model in windows OS has produced accurate results under different conditions such as in presence of spectacles, in absence of spectacles, focusing light in front, and with varying distances. Thus, this model has been deployed in android OS. Unlike other researches, it doesn't require additional hardware devices except for the mobile phone.

## 11. FUTURE SCOPE

In this covid world, during this pandemic situation, many states and countries have shifted to online education and work from home. Monitoring a high number of learners and listeners becomes very difficult. Alerting them through an automated system is necessary. Thus, the developed model for DMS can also be extended in the mere future for achieving self-alertness.

## REFERENCES

[1] 'Drowsiness Detection with Machine Learning _ by Grant Zhong _ Towards Data Science'https://towardsdatascience.com/drowsiness-detection-with-machine-learning-765a16ca208a.

[2] W. Han, Y. Yang, G. Bin Huang, O. Sourina, F. Klanner, and C. Denk, 'Driver Drowsiness Detection Based on Novel Eye Openness Recognition Method and Unsupervised Feature Learning', *Proc. - 2015 IEEE Int. Conf. Syst. Man, Cybern. SMC 2015*, no. October, pp. 1470–1475, 2016, doi: 10.1109/SMC.2015.260.

[3] R. Jabbar, M. Shinoy, M. Kharbeche, K. Al-Khalifa, M. Krichen, and K. Barkaoui, 'Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application', *2020 IEEE Int. Conf. Informatics, IoT, Enabling Technol. ICIoT 2020*, pp. 237–242, 2020, doi: 10.1109/ICIoT48696.2020.9089484.

[4] S. K. Vasudevan, J. Anudeep, G. Kowshik, and P. R. Nair, 'An AI Approach for Real-Time Driver Drowsiness Detection—A Novel Attempt with High Accuracy', *Lect. Notes Networks Syst.*, vol. 127, pp. 305–316, 2021, doi: 10.1007/978-981-15-4218-3_30.

[5] V. R. Reddy Chirra, S. R. Uyyala, and V. K. Kishore Kolli, 'Deep CNN: A machine learning approach for driver drowsiness detection based on eye state', *Rev. d'Intelligence Artif.*, vol. 33, no. 6, pp. 461–466, 2019, doi: 10.18280/ria.330609.

[6] A. Ignatov *et al.*, 'AI Benchmark: Running Deep Neural Networks on Android Smartphones Radu Timofte Luc Van Gool'.

[7] N. K. Sancheti, 'Camera-based driver monitoring system using deep learning Camera based driver monitoring system using deep learning'.

[8] F. Song, X. Tan, X. Liu, and S. Chen, 'Eyes closeness detection from still images with multi-scale histograms of principal oriented gradients', *Pattern Recognit.*, vol. 47, no. 9, pp. 2825–2838, 2014, doi: 10.1016/j.patcog.2014.03.024.

[9] W. Deng and R. Wu, 'Real-Time Driver-Drowsiness Detection System Using Facial Features', *IEEE Access*, vol. 7, pp. 118727–118738,2019,doi:10.1109/access.2019.2936663.

[10] E. R. Anas, P. Henriquez, and B. J. Matuszewski, 'Online eye status detection in the wild with convolutional neural networks', *VISIGRAPP 2017 - Proc. 12th Int. Jt. Conf. Comput. Vision, Imaging Comput. Graph. Theory Appl.*, vol. 6, no. Visigrapp, pp. 88–95, 2017, doi: 10.5220/0006172700880095.

[11] J. Nabi, 'Hyper-parameter Tuning Techniques in Deep Learning | by Javaid Nabi | Towards Data Science', pp. 1–16, 2019,[Online].Available:https://towardsdatascience.com/hyper-parameter-tuning-techniques-in-deep-learning-4dad592c63c8.

[12] M. Hasnain, M. F. Pasha, I. Ghani, M. Imran, M. Y. Alzahrani, and R. Budiarto, 'Evaluating Trust Prediction and Confusion Matrix Measures for Web Services Ranking', *IEEE Access*, vol. 8, pp. 90847–90861, 2020, doi: 10.1109/ACCESS.2020.2994222.

[13] C. B. S. Maior, M. J. das C. Moura, J. M. M. Santana, and I. D. Lins, 'Real-time classification for autonomous drowsiness detection using eye aspect ratio', *Expert Syst. Appl.*, vol. 158, no. September, 2020, doi: 10.1016/j.eswa.2020.113505.

[14] M. Xu, 'Robust object detection with real-time fusion of multiview foreground silhouettes', *Opt. Eng.*, vol. 51, no. 4, p. 047202, 2012, doi: 10.1117/1.oe.51.4.047202.

[15] A. Dongre, A. Kumawat, A. S. Kushwah, and P. R. Jain, 'REAL-TIME DRIVER FATIGUE DETECTION USING EYE DETECTION', no. 05, pp. 957–960, 2020.

[16] A. Kumar, 'ISCAIE 2014 - 2014 IEEE Symposium on Computer Applications and Industrial Electronics', *ISCAIE 2014 - 2014 IEEE Symp. Comput. Appl. Ind. Electron.*, p. 237, 2015.

[17] Adrian Rosebrock, 'Training a custom dlib shapepredictor-PyImageSearch', *Pyimagesearch*, pp. 1–64, 2019, [Online]. Available:https://www.pyimagesearch.com/2019/12/16/training-a-custom-dlib-shape-predictor/.

[18] G. Poovarasan, S. Susikumar, S. Naveen, and 'International Journal of Engineering Technology Research & Management', *Academia.Edu*, no. 03, pp. 131–134, 2020, [Online].Available:http://www.academia.edu/download/62142316/Jan-2020-17-1579239734-820200219-79933-ihqi97.pdf.