Deccan Education Society's
Fergusson College (Autonomous), Pune
Department of Computer Science

A

Project Report
on

"DRIVER DROWSINESS DETECTION"

In partial fulfillment of Post Graduate course

in

M.Sc. Computer Science II

(Semester -III)

CSC4106 Computer Science Project - III

SUBMITTED BY

SRUSHTI PADALE - 226223
ASHWINI JAWALE - 226237
VRUSHALI SHEWALE - 226261

Deccan Education Society's
Fergusson College (Autonomous), Pune
Department of Computer Science

CERTIFICATE

This is to certify that the project entitled **Driver Drowsiness Detection** submitted by

1. SRUSHTI PADALE -226223
2. ASHWINI JAWALE -226237
3. VRUSHALI SHEWALE -226261

in partial fulfillment of the requirement of the completion of M.Sc.(C.S)-II [Semester-III], has been carried out by them under our guidance satisfactorily during the academic year 2023-2024.

Place: Pune
Date: 23/11/2023

Dr. Kavita Khobragade
Head,
Department of Computer Science
Fergusson College (Autonomous), Pune

Project Guide: - Aparna Vaidyanathan

Examiners Name **Sign**

1. _____ _____

2. _____ _____

Index

1. Introduction

1.1 Artificial Intelligence

Artificial Intelligence (AI) technologies exercise capabilities that were previously possessed only by human beings - like knowledge, perception, insight, and ability to swiftly solve highly precise jobs such as identifying structures or interpreting the meanings. AI incorporates various underlying technologies which have various alternative functions to perform.

Technology	Brief Description	Example Application
Statistical Machine Learning	Automates Processing and fitting models of data.	Highly granular marketing analyses on big data.
Neural Networks	Uses artificial “neurons” to weight inputs and relate them to outputs	Identifying credit fraud, weather prediction.
Deep Learning	Neural networks with many layers of variables or features	Image and voice recognition, extracting meaning from text.
Natural Language Processing	Analyses and “understands” human speech and text	Speech recognition, chatbots, intelligent agents.
Rule-based expert system	A set of logical rules derived from human experts.	Insurance underwriting, credit approval.
Physical Robots	Automates a physical activity.	Factory and warehouse tasks.
Robotic Process Automation	Automates structured digital tasks and interfaces with systems.	Credit card replacement, validating online credentials.

1.2 Advantages of Artificial Intelligence

Results achieved by the State-of-the-Art Neural Networks - The motivation for choosing the Neural Network (NN) approach for solving tasks like image classification boils down to the performance that the NN has achieved. In 2016, Microsoft developed a NN called ResNet (Residual Neural Network) which was the winner of the *ImageNet* Challenge. The dataset consisted of about 1.2 Million training images with 1000 classes, which gave an error rate of 4.94 percent or about approx. 97% accuracy. The Inception Network was one of the major breakthroughs in the field of NNs. In 2014, Google team developed a NN called Google Net.

Accuracy Advantage - Results achieved by the combination of new External Data sources and Machine Learning (ML) methods yield dramatic improvements in profitability. For example, employed ML-based models to assess supply and demand to price their charter trips, which incorporated External Datasets extensively for predicting demands. The company profited from it and revenue per occupied flight rose 5 percent and had fewer deadhead hours.

Preventing from Risks and Hazards - One of the biggest advantages of AI is developing AI Robots which can overcome risky limitations of human beings. For example, defusing bombs, going to mars, exploring oceans, mining coal and oil, et cetera, where human intervention can be hazardous.

1.3 Machine Learning

This research project is based on Machine Learning (ML) technology. The project utilizes one of the following ML techniques to train the neural network model. However, other learning techniques can also be inculcated depending on the task needed to be solved. Besides the algorithms used, another key dimension of ML is how the models learn. ML has mainly three learning approaches - Supervised, Unsupervised and Semi-supervised learning.

- **Supervised Learning** - This research project is primarily based on the supervised learning technique. Supervised learning algorithms account for substantial research activity in ML, especially in tasks like Classification. This method utilizes already labeled data to forecast events. Supervised training paradigm compares the predicted results with the actual results to identify errors and change the model-weights based on results. However, the selection of relevant data is crucial for supervised learning to work efficiently or there are possibilities of classification problems like Class Imbalance. The performance metrics used in this research project will take care of the class imbalance case and will penalize the performance, if the model has high bias towards any of the class.

- **Unsupervised Learning -** In unsupervised learning approaches, the model does not require any labeled data, the results are unknown as there is no target or expected output to compare to. This method discovers and writes observations from the unlabeled dataset to find hidden patterns and deduce a function to explain these hidden patterns. Unsupervised learning is crucial since it is more prevalent in the brain than supervised learning.

- **Semi-Supervised Learning -** Semi-supervised learning is advantageous in cases where the model takes in a smaller quantity of labeled data along with a larger quantity of unlabeled data. For example, a massive amount of email data is generated every day, most of which is uncategorized or 'unlabeled', and only a few amounts of email data is categorized or 'labeled' like emails marked spam or important by the users. It makes it difficult for supervised learning methods to automate applications like e-mail spam filtering, et cetera, with such type of data.

1.4 Traditional Learning and Transfer Learning

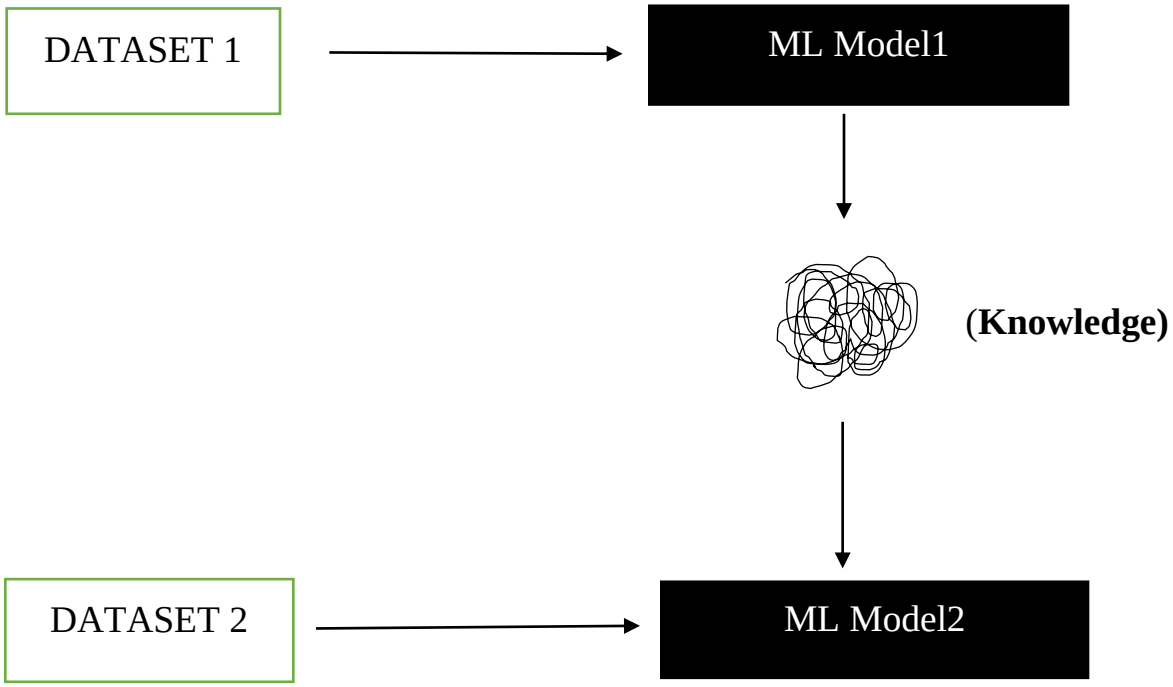
In Traditional Learning, accuracy is not that much high (Because it is learning from scratch) and the more time to train a model, less is accuracy.

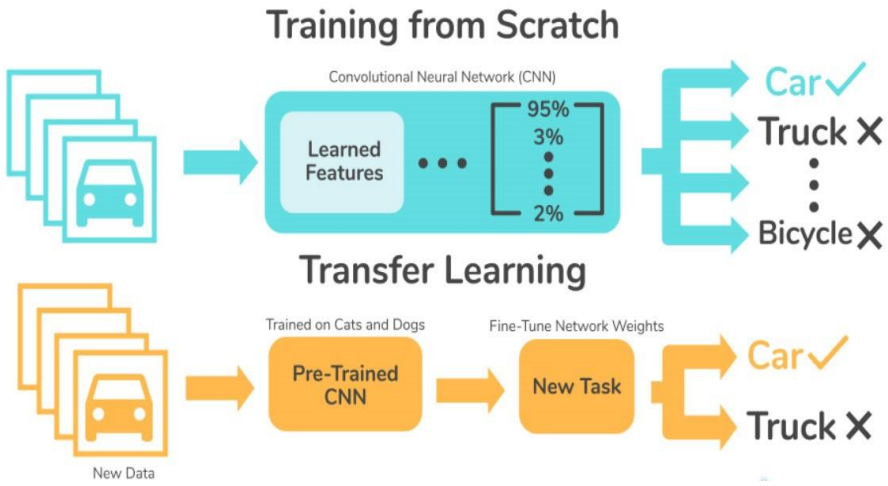
Train Model1 by scratch. Model1 will try to learn features and it will recognize.

Make Deep Learning Model2, train it from scratch and it will recognize how the person looks, eyes etc.

Transfer Learning

In Transfer Learning, whatever you have studied in Model1, all the knowledge of Model1 will be applied on Model2, you do not need to train your Model from scratch individually. So, it will not take much time because Model1 already knows how a person looks, his/her eye information etc. Here accuracy will be higher. It is basically transferring the knowledge of one project Model to another is called 'Transfer learning'. Our Project is mostly based on Transfer Learning.


DATASET 1
ML Model1
(Knowledge)
DATASET 2
ML Model2


Training from Scratch
Convolutional Neural Network (CNN)
Learned Features
95%
3%
2%
Car
Truck
Bicycle
Transfer Learning
Trained on Cats and Dogs
Pre-Trained CNN
Fine-Tune Network Weights
New Task
Car
Truck
New Data

2) Tasks solved using Neural Network.

Several tasks can be solved using neural networks depending on their use-case. However, these tasks are broadly classified into two major categories - Prediction of Categorical values vs Prediction of Continuous values.

2.1 Classification

Classification falls under the category of Predicting Categorical Values and this research project mainly deals with the task of Classification. It is a supervised learning technique since it categorizes data from prior information. Classification is done in two phases - First, a classification-based NN is applied to the training data set and then the trained model is validated against a labeled test data set to measure the model's accuracy and performance. Classification is essential for ML, data analytics and pattern recognition. Applications of classification include fraud detection, document and image classification, spam filtering, risk analysis, etc.

3) Dataset

As Neural Networks learn from the training dataset and make predictions on the test (unseen) dataset, it requires a large amount of training data. Since datasets can contain diversified information, it often leads to them being balanced or imbalanced.

- **Balanced Dataset -** A dataset is balanced if it has an equal number of images or samples for all the classes or categories present in the dataset.

- **Imbalanced Dataset** - Any data set that shows an unequal distribution between its classes or an unequal number of images for all the categories can be considered imbalanced. Learning from datasets that hold very few instances of the minority or interesting class tends to produce biased classifiers. Such classifiers usually show higher predictive accuracy over the majority classes and poorer predictive accuracy over the minority classes. There are several techniques to tackle imbalances in data level, for example - Synthetic Minority Oversampling Technique (SMOTE) which creates synthetic examples between two close vectors that lay together.

3.1 Drivers Drowsiness Detection Dataset

subject ID:

xxx

image number:

xxx

gender:

0 - male

1 - female

glasses:

0 - no

1 - yes

eye state:

0 - close

1 - open

reflections:

0 - none

1 - low

2 - high

lighting conditions/image quality:

0 - bad

1 - good

sensor type:

01 - RealSense SR300 640x480

02 - IDS Imaging, 1280x1024

03 - Aptina Imagin 752x480

example:

s001_00123_0_0_0_0_0_01.png

The Dataset has been taken from [MRL Eye Dataset | MRL (vsb.cz)]

In the dataset, we annotated the following properties (the properties are indicated in the following order):

subject ID; in the dataset, we collected the data of 37 different persons (33 men and 4 women)

image ID; the dataset consists of **84,898** images

gender [0 - man, 1 - woman]; the dataset contains the information about gender for each image (man, woman)

glasses [0 - no, 1 - yes]; the information if the eye image contains glasses is also provided for each image (with and without the glasses)

eye state [0 - closed, 1 - open]; this property contains the information about two eye states (open, close)

reflections [0 - none, 1 - small, 2 - big]; we annotated three reflection states based on the size of reflections (none, small, and big reflections)

lighting conditions [0 - bad, 1 - good]; each image has two states (bad, good) based on the amount of light during capturing the videos

sensor ID [01 - RealSense, 02 - IDS, 03 - Aptina]; at this moment, the dataset contains the images captured by three different sensors (Intel RealSense RS 300 sensor with 640 x 480 resolution, IDS Imaging sensor with 1280 x 1024 resolution, and Aptina sensor with 752 x 480 resolution)

An examples of image annotations of the proposed dataset:

s0012_03054_0_1_0_2_1_01

s0014_04371_0_0_1_0_1_03

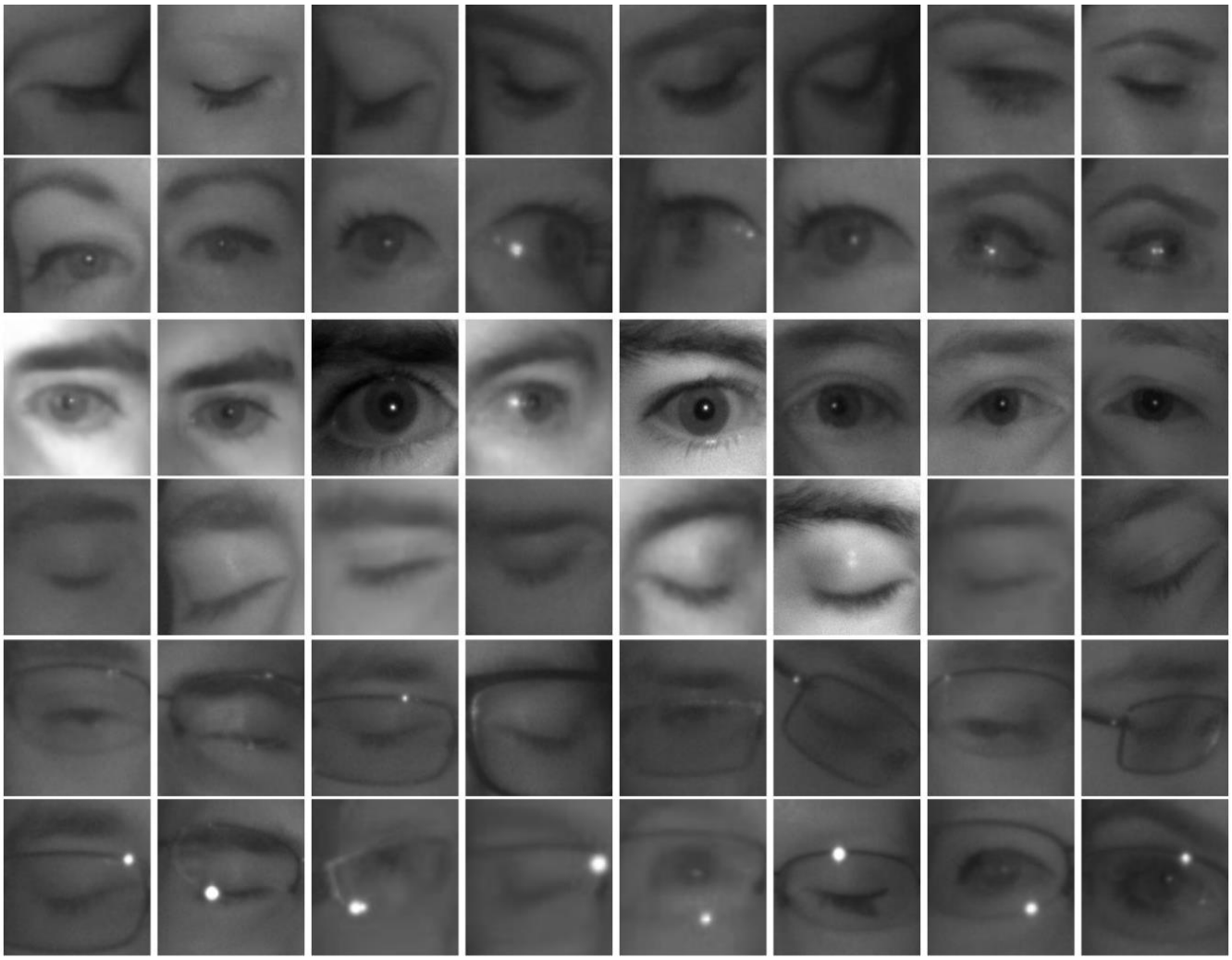
s0037_08976_1_1_1_0_0_01

s0014_03559_0_0_0_0_1_02

s0014_07350_0_0_1_1_1_02

s0036_02165_1_0_1_0_0_01

THE EXAMPLE IMAGES FROM THIS DATASET OF OPEN AND CLOSED EYES ARE SHOWN BELOW:



The dataset consists of 84,898 images

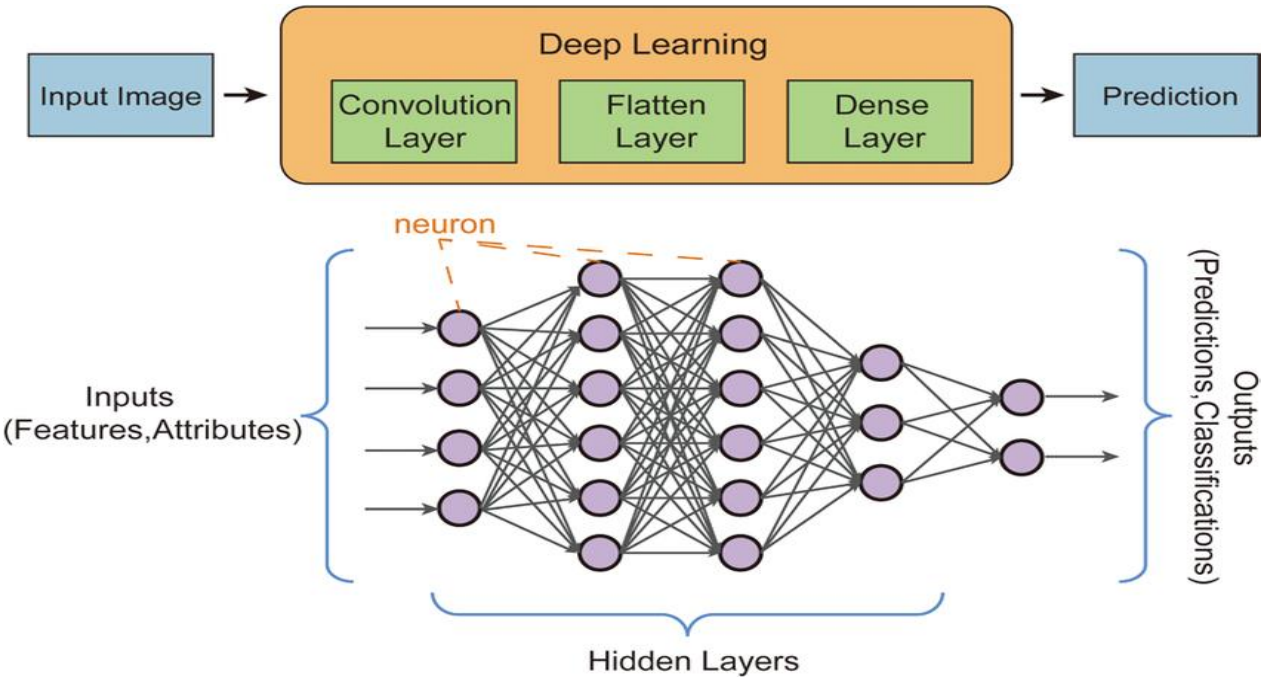
Class Labels	Class Description	Total Number of Images in each Class
Test Dataset	Close	2,010
	Open	2,010
Train Dataset	Close	39,936
	Open	40,942

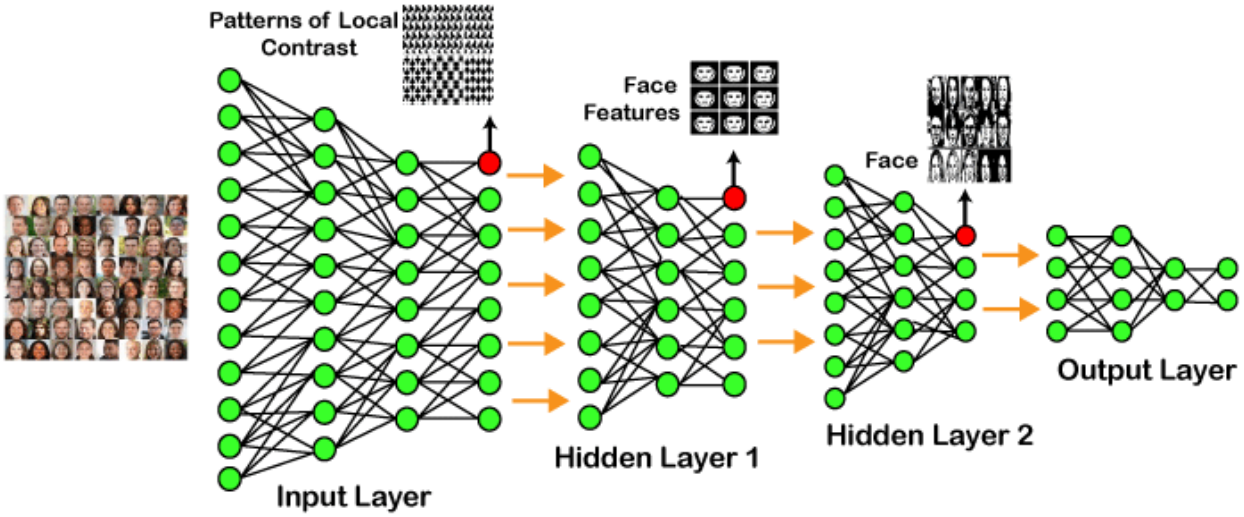
4) Deep Learning

Deep learning is a branch of machine learning which is based on artificial neural networks. It is capable of learning complex patterns and relationships within data. In deep learning, we don't need to explicitly program everything. It has become increasingly popular in recent years due to the advances in processing power and the availability of large datasets. Because it is based on artificial neural networks (ANNs) also known as deep neural networks (DNNs). These neural networks are inspired by the structure and function of the human brain's biological neurons, and they are designed to learn from large amounts of data.

Deep Learning is a subfield of Machine Learning that involves the use of neural networks to model and solve complex problems. Neural networks are modeled after the structure and function of the human brain and consist of layers of interconnected nodes that process and transform data.

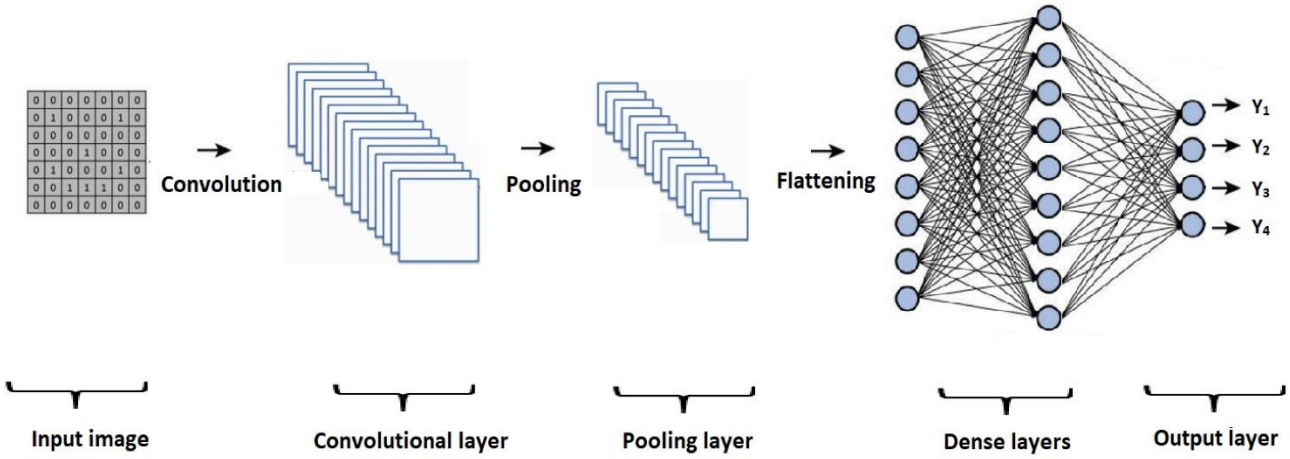
The key characteristic of Deep Learning is the use of deep neural networks, which have multiple layers of interconnected nodes. These networks can learn complex representations of data by discovering hierarchical patterns and features in the data.


Input Image
Deep Learning
Convolution Layer
Flatten Layer
Dense Layer
Prediction
neuron
Inputs
(Features,Attributes)
Outputs
(Predictions,Classifications)
Hidden Layers

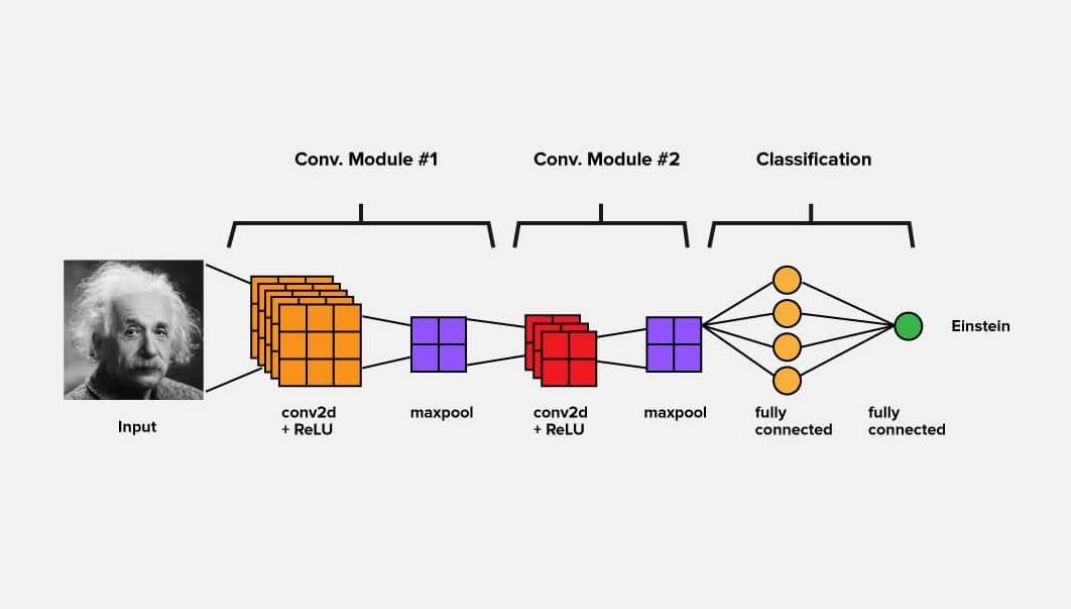

Patterns of Local Contrast
Face Features
Face
Input Layer
Hidden Layer 1
Hidden Layer 2
Output Layer

4.1 Convolution Neural Network

Convolutional Neural Network (CNN) is a type of Deep Learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers and fully connected layers. A CNN is a Deep Learning algorithm which takes in input images, assigns learnable biases and weights to various objects or aspects in the image and differentiate one object from another. CNN is made up of multiple convolution layers which serve as feature extractors - each layer responsible for identifying distinct visual receptive field patterns.



4.1 Working Principle of Convolution Neural Network

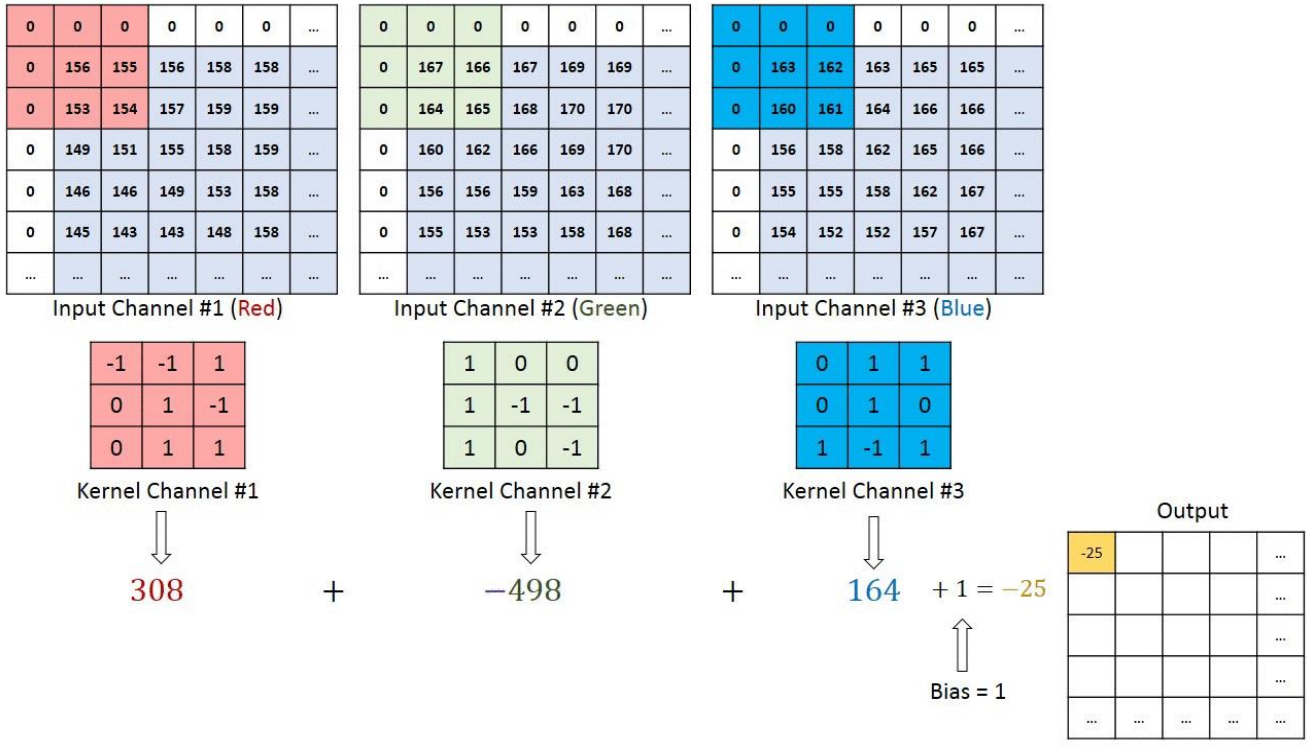


A convolutional neural network consists of an input layer, multiple hidden layers, and an output layer. A single image or multiple images are passed as inputs to the input layer. The dimensions of the input image - height, width, channels, et cetera, are passed as the parameters of the input layer.

The hidden layers accommodate a series of convolution layers which acquires the understanding of image characteristics and provide a compact representation in the form of Feature Maps with the help of inherent hierarchical patterns found in the image data. The weights of the filter are automatically adjusted based on the error at the output during the learning process. Activation Functions are used in CNNs to make it capable of learning and performing more complex tasks as it introduces non-linearity into the output of a neuron. Activation functions such as Rectified Linear Unit (ReLU), Softmax, LeakyReLU, Sigmoid, and Tanh are used for this purpose.

The output layer of the CNN architecture is a fully connected layer. All the neurons in the fully connected layer are connected to the activation function of the previous layer. The activation values are summed for each neuron, and the neuron with the highest value is given as the output.

An input image of size (x × y) *px* and a convolution filter (typically, 1×1, 3×3, 5×5, and 7×7) is taken. The image contains pixel information, and the convolution filter consists of weights. These weights determine how important the pixel is at that location to learn and extract features. The Filter then slides across the entire x, y dimensions of the image. During the sliding operation, the dot product is computed between the filter's location and the image's pixels at that location. The resulting dot product generates a 2D feature map. All of the generated feature maps are stacked on top of each other to produce the final convolution output. One pass of the convolution operation is illustrated in Figure below:



In Figure above, Red, Blue, and Green are three different filters applied on the input image frame channel. These filters slide over the whole input channel and their results are summed up to produce the final output.

4.2.1 Neural Network Layers

- **Functions**

```
In [1]: import tensorflow as tf
        from tensorflow.keras.applications import InceptionV3
        from tensorflow.keras.models import Model
        from tensorflow.keras.layers import Dropout,Input,Flatten,Dense,MaxPooling2D
        from tensorflow.keras.optimizers import Adam
        from tensorflow.keras.preprocessing.image import ImageDataGenerator  # Data Augumentation
```

- **Tensorflow**

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

- **InceptionV3**

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9

InceptionV3 is a convolutional neural network for assisting in image analysis and object detection. This function returns a Keras image classification model, optionally loaded with weights pre-trained on ImageNet.

- **Arguments**

Inputs will be the 'knowledge' of the base model (bmodel), last layer of the two neurons is head model (hmodel) . We just have to train the last two layers and use knowledge of InceptionV3. So we just used them and their knowledge and not 'train' them. Hence, 'layer.trainable = false'.

```
In [7]: bmodel = InceptionV3(include_top=False, weights='imagenet', input_tensor=Input(shape=(80,80,3)))
        hmodel = bmodel.output
        hmodel = Flatten()(hmodel)
        hmodel = Dense(64, activation='relu')(hmodel)
        hmodel = Dropout(0.5)(hmodel)
        hmodel = Dense(2,activation= 'softmax')(hmodel)

        model = Model(inputs=bmodel.input, outputs= hmodel)
        for layer in bmodel.layers:
            layer.trainable = False
```

include_top: Boolean, whether to include the fully-connected layer at the top, as the last layer of the network. Defaults to true.

weights: One of None (random initialization), imagenet (pre-training), or the path to the weights file to be loaded. Defaults to imagenet.

input_tensor: Optional Keras tensor (output of layers.input()) to use as image input for the model. It is useful for sharing inputs between multiple different networks. Defaults to None.

input_shape: Optional shape tuple, only to be specified if include.top is false. It should have exactly 3 input channels, and width and height should be no smaller than 75.

Activation Function

The primary decision-making units in NN are activation functions as they calculate the output of NN nodes. Activation function decides whether a neuron should be activated or not by calculating weighted sum and further adding bias with it. The activation function employed in this research project is Rectified Linear Unit (ReLU).

- **ReLu** $f(x)$- It is a piecewise linear function that outputs the input directly if it is positive; otherwise, the output is zero [25].

$$f(x) = x^+ = \max(0, x)$$

Softmax- The most common use of the softmax function in applied machine learning is in its use as an activation function in a neural network model. Specifically, the network is configured to output N values, one for each class in the classification task, and the softmax function is used to normalize the outputs, converting them from weighted sum values into probabilities that sum to one. Each value in the output of the softmax function is interpreted as the probability of membership for each class.

Example- For example, we can turn the first value "1" in the list [1, 3, 2] into a probability as follows:

Probability = exp (1) / (exp (1) + exp (3) + exp (2))

probability = exp (1) / (exp (1) + exp (3) + exp (2))

probability = 2.718281828459045 / 30.19287485057736

probability = 0.09003057317038046

4.2.2 Fully Connected Layer / Dense Layer

When all the inputs from one layer are connected to every activation unit of the next layer, a Fully Connected (FC) or Dense Layer is formed. FC layers compile the data extracted by previous layers to give the final output. Not all layers are fully connected as it would be computationally expensive. In the Classification task, the model needs to classify the data into various classes after the feature extraction process. In this research project, after passing through the FC layers, the output layer uses the 'Softmax' activation function to get probabilities of the input object in the image belonging to the classes.

4.2.3 Dropout Layer

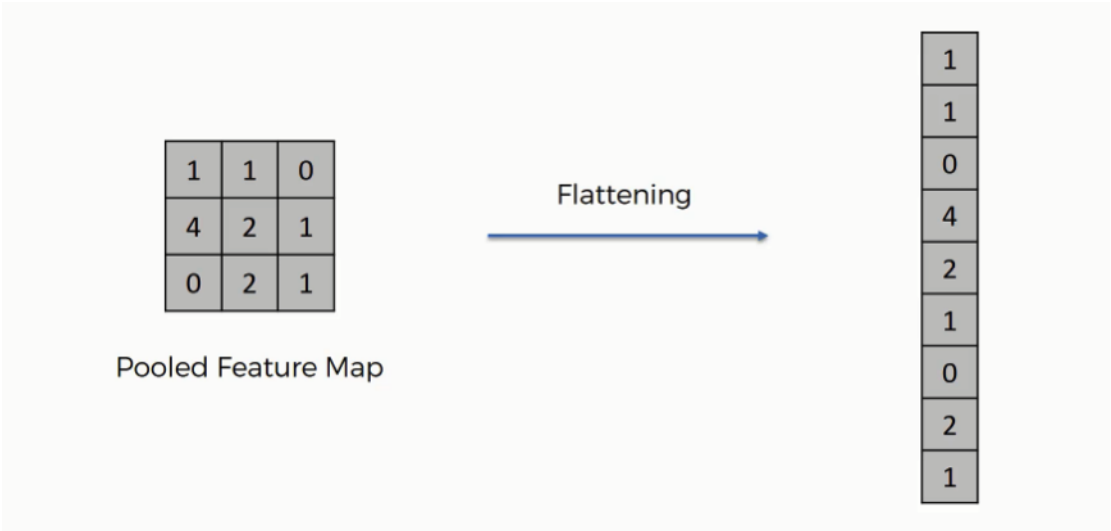
Deep Neural Networks with FC layers occupy many parameters due to which neurons develop co-dependency amongst each other and their individual power is restrained. This leads to overfitting of training data. Dropout layers 'randomly' drop units along with their connections to the next layer during training to prevent units from co-adapting. Dropping out unit means temporarily removing it from the network, along with all its incoming and outgoing connections. Dropout prevents overfitting.

```
In [8]: bmodel.summary()
```

```
Model: "inception_v3"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_1 (InputLayer)        [(None, 80, 80, 3)]       0         []

 conv2d (Conv2D)             (None, 39, 39, 32)        864       ['input_1[0][0]']

 batch_normalization (Batch  (None, 39, 39, 32)        96        ['conv2d[0][0]']
 Normalization)

 activation (Activation)     (None, 39, 39, 32)        0         ['batch_normalization[0][0]']

 conv2d_1 (Conv2D)           (None, 37, 37, 32)        9216      ['activation[0][0]']

 batch_normalization_1 (Bat  (None, 37, 37, 32)        96        ['conv2d_1[0][0]']
 chNormalization)

 activation_1 (Activation)   (None, 37, 37, 32)        0         ['batch_normalization_1[0][0]'
```

4.2.4 Flatten Layer

Flattens the input. does not affect the batch size.



5) Neural Network Hyper parameters

5.1 Loss Function

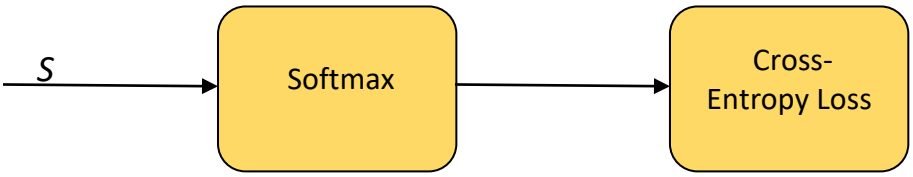
Loss function computes the distance between current output of the algorithm and the expected output. Error function is used to compute a model's loss so that the weights can be updated to reduce the loss on the next evaluation. There are multiple loss functions available to match or choose from when training NNs depending on the specific predictive modeling problem. As this research project is a typical Multi-Class Classification problem, a Multi-Class Cross-Entropy Loss is an appropriate loss function for it. The loss function employed in this project is Categorical Cross Entropy.

```
In [12]: model.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['accuracy'])
```

- **Categorical Cross Entropy –**

The Categorical Cross Entropy Loss function is a Softmax activation plus a Cross-Entropy loss. Cross-entropy calculates a score that summarizes the average difference between the actual and predicted probability distributions for all classes in the problem.

Cross-Entropy is specified as the loss function in Keras by specifying 'categorical_crossentropy' when compiling the model. The function requires that the output layer is configured with n nodes for each class respectively, in this case ten nodes, and a 'Softmax' activation in order to predict the probability for each class.



$$\boldsymbol{f(s)} = \frac{\boldsymbol{e^{s}}_{i}}{\sum_{j}^{c} \boldsymbol{e^{s}}_{i}} \qquad \boldsymbol{CE} = -\sum_{i}^{c} t_{\ i} log(f(s)_{\ i})$$

Where S_j are the scores inferred by the net for each class in C. Note that the Softmax activation for a class S_i depends on all the scores in s.

5.2 Optimizer

Optimizers change the attributes such as weights and learning rate to reduce the losses. There are many types of optimization methods available to choose from. Several of these can be found in the paper given by the authors in. This research project employs 'ADAM' Optimizer in order to provide the most accurate results possible.

ADAM (Adaptive Moment Estimation) - ADAM is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments**.** It is computationally efficient and has very little memory requirement.

6) Neural Network Optimization

This research project employs two kinds of Neural Network Optimisations - '**EarlyStopping**' and **'ReduceLROnPlateau'** from Keras, as shown in the code below:

```
earlystop = EarlyStopping(monitor = 'val_loss', patience=7, verbose= 3, restore_best_weights=True)

learning_rate = ReduceLROnPlateau(monitor= 'val_loss', patience=3, verbose= 3, )

callbacks=[checkpoint,earlystop,learning_rate]
```

6.1 Reduce Learning Rate on Plateau

- **Learning Rate (LR)** or Step Size is a hyperparameter that controls how quickly the model is adapted to the problem in response to the estimated error each time the model weights are updated. If LR is too large - the model converges too quickly to a suboptimal solution, whereas if LR is too small - the learning process gets stuck.

- The Reduce Learning Rate on Plateau approach is to adjust the LR only when the optimizer has reached some plateau (for example, the model cannot improve the performance over some number of epochs). To improve the performance, the optimizer needs the step size to be reduced. Keras provides the ReduceLROnPlateau function that adjusts the LR when a plateau in model performance is detected.

The ReduceLROnPlateau function requires to:

- Specify the metric to monitor during training via the **"monitor"** argument.
- Specify the number of training epochs to wait before triggering the change in LR via **"patience"** argument.

6.2 Early Stopping During Training

Too many epochs can lead to overfitting of the training dataset, whereas too few may result in underfitting of the model. Early Stopping allows an arbitrarily large number of training epochs and stops training once the model performance stops improving after a certain number of epochs on the validation dataset. Early Stopping requires that a validation dataset is evaluated during training. The model is not trained on the validation dataset. Instead, the model is evaluated on the validation dataset at the end of each training epoch.

```
Epoch 1/5
8087/8087 [==============================] - ETA: 0s - loss: 0.2082 - accuracy: 0.9189
Epoch 1: val_loss improved from inf to 0.17209, saving model to D:\FinalYearProject\models\model.h5
```

7) Performance Metrics for Evaluation

7.1 Accuracy

Classification Accuracy is defined as the percentage of correct predictions for the test data. It is the ratio of number of correct predictions to the total number of input samples. It works well only if there are an equal number of samples belonging to each class, it does not do well when you have a severe class imbalance.

$$\textbf{Accuracy} = \frac{\textbf{Number of accurate Predictions}}{\textbf{Total number of Predictions made}}$$

7.2 Precision

Precision is defined as the fraction of relevant examples (true positives) among all of the examples which were predicted to belong in a certain class (sum of true positives and false positives).

$$\textbf{Precision} = \frac{\textbf{TruePositives}}{\textbf{TruePositives + FalsePositives}}$$

7.3 Recall

Recall is defined as the fraction of examples which were predicted to belong to a class with respect to all of the examples that truly belong in the class.

$$\textbf{Recall} = \frac{\textbf{TruePositives}}{\textbf{TruePositives + FalseNegatives}}$$

7.4 F1 Score

F1 Score is an overall measure of a model's accuracy as it is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1] - with score '1' the model is considered perfectly accurate and score '0' meaning the model is a total failure. A good F1 score means low false positives and low false negatives, and the model is correctly identifying real threats and not giving false alarms.

$$\textbf{F1 Score} = 2 \times \frac{\{\textbf{Precision} \cdot \textbf{Recall}\}}{\{\textbf{Precision + Recall}\}}$$

8) Results

Drivers Drowsiness Detection Dataset is an open-source dataset. Originally, the dataset contained a total of 84,898 images. Since, **'eyestate'** is an important attribute, the dataset is divided into two categories – Test-Set and Train-Set. The Test-Set now consists of 2,010 images each in Open Eyes and Close Eyes-Set respectively. And the Train-Set for Open Eyes consists of 40,942 and for Close Eyes, 39,936. To develop the model, TensorFlow Framework with Keras API is used. The model is trained and tested on Jupyter Notebook as the development IDE. The GPU used to train the model is randomly allocated, which generally allocates a memory capacity of 12 GBs. The Trained model stores the weights of the model.

8.1 Model's Loss and Accuracy Graph

Following are the graphs plotted for the model's accuracy and loss after the model is trained on the dataset. The accuracy graph is obtained after evaluating the model on the test dataset.

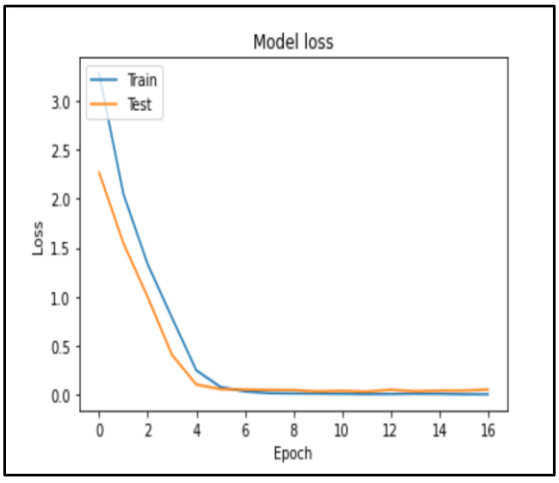


Figure 8.1.1: Result - Loss Graph

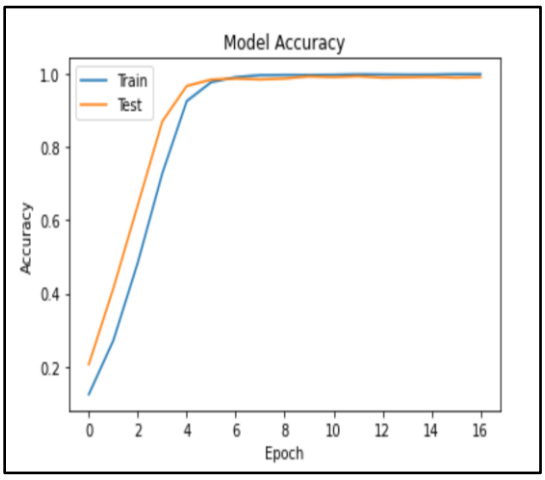


Figure 8.1.2: Result - Accuracy Graph

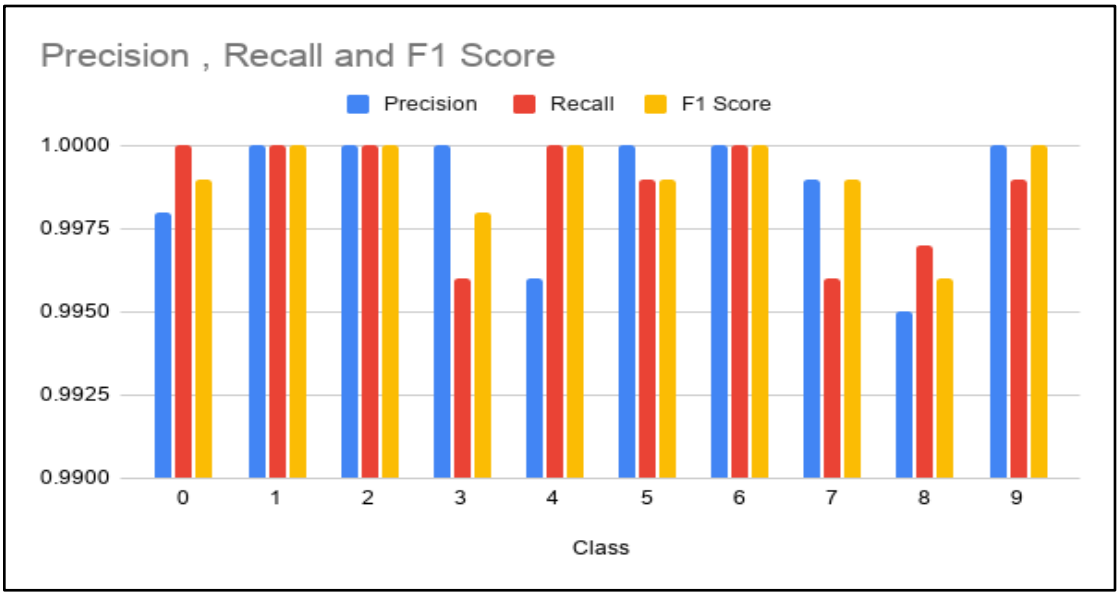


Figure 8.1.3: Result - Graph of Average values of Precision, Recall and F1 Score

9) Conclusion

In this research project, a novel CNN model was developed to detect distracted actions of the driver. The model was successfully able to detect and classify distracted actions. The CNN model developed is a Feed-Forward Neural Network with its core component being the convolution layers which are the feature extraction layers. The model was trained using a (80-10)-10 train-validation-test split. The result obtained from the model showed that on an average, the model was able to classify 99 out 100 cases of distracted actions.

9.1) Future Enhancement

Further improvements can be achieved by reducing the model size (number of parameters) and trying to maintain the same level of performance as the original model.

Further research can be done to check the performance of the model by training it on a different Multi-Class Dataset and monitor the predictions obtained by the model.