

DROWSINESS DETECTION SYSTEM USING CNN

MINOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE AWARD OF DEGREE OF

BACHELOR OF TECHNOLOGY

INFORMATION TECHNOLOGY



SUBMITTED BY:

Ikrar Khan(1706857)

Kavish Thakur(1805520)

SUBMITTED TO:

Prof.Ranjodh Kaur

Assistant Professor

(Project Coordinator)

**Guru Nanak Dev Engineering College
(Information Technology Department)**

Abstract

"Drowsiness Detection system using CNN" is model which will be useful to tell us about a person if he is drowsy or not. Car accident is the major cause of death in which around 1.3 million people die every year. Majority of these accidents are caused because of distraction or the drowsiness of driver. Construction of high-speed highway roads had diminished the margin of error for the driver. The countless number of people drives for long distance every day and night on the highway. Lack of sleep or distractions like the phone call, talking with the passenger, etc may lead to an accident. To prevent such accidents we propose a system which alerts the driver if the driver gets distracted or feels drowsy. I think the stats on the accidents due to sleepiness will encourage us to pick this project.

Acknowledgement

We are highly grateful to Dr. Sehajpal Singh, Principal, Guru Nanak Dev Engineering College (GNDEC), Ludhiana, for providing this opportunity to carry out the minor project work. The constant guidance and encouragement received from Prof. Pankaj Bhambri, IT Department, GNDEC Ludhiana has been of great help in carrying out the project work and is acknowledged with reverential thanks. We would like to express a deep sense of gratitude and thanks profusely to Prof. Ranjodh Kaur without her wise counsel and able guidance, it would have been impossible to complete the project in this manner. We express gratitude to other faculty members of Information Technology Department of GNDEC for their intellectual support throughout the course of this work. Finally, we are indebted to all whosoever have contributed in this report work.

List of Figures

1	Face Detection	6
2	Gradient Descent	7
3	Facts to back our model	8
4	Drowsy state is dangerous while driving	10
5	Schematic Diagram of CNN	11
6	Kera Library	15
7	Multiple Face Detected	16
8	Architecture of our Model	17
9	Diagram of Pooling	18
10	Diagram of Flattening	19
11	Layers in Keras Library	20
12	CNN Block1	22
13	CNN Block2	23
14	Flowchart of our Model	25
15	First Shell of Code Defining Libraries	26
16	Second shell of code telling about image display	27
17	Third shell of code about Face Detection	27
18	Importing Libraries	28
19	Model Design	29
20	Adding optimizers and Loss	29
21	Training and Validating Model	30
22	To run and Give epochs	31
23	Result of our Model	32

Contents

1	Introduction	6
1.1	What is the idea?	6
1.2	Project Category?	6
1.3	Objective	6
1.3.1	Main Objectives	6
1.4	Problem and Identification	7
1.5	Some Facts!	8
1.6	What has been done:	10
1.7	Proposed System	11
1.8	Unique features of the system	11
2	Requirement Analysis and System Specification	13
2.1	Feasibility Study	13
2.1.1	Is this plan technically feasible?	13
2.1.2	Is this system Operationally feasible?	13
2.1.3	Is this system Economically feasible?	13
2.2	Software or Data Requirements	13
2.2.1	Images	13
2.2.2	Tensorflow framework	14
2.2.3	Keras library	14
2.3	Validation	15
2.4	Expected Hurdles	15
2.4.1	Classification of Images with different Positions:	15
2.4.2	Minor Drawbacks of CNN:	16
3	System Design	17
3.1	Model Architecture	17
3.1.1	Convolution:	18
3.1.2	Max pooling:	18
3.1.3	Dropout:	18
3.1.4	Flattening	19
3.1.5	Dense Layer	19

3.2	User Interface Design	20
3.3	Database designing	20
3.3.1	Requirement Analysis	20
3.3.2	Database	20
3.3.3	Implementation	21
3.4	Methodology	21
4	Implementation,Testing and Maintenance	23
4.1	Tools,Languages etc.	23
5	Coding standards of language	23
6	Result and Discussion	24
6.1	User Interface Representation	24
6.2	Screenshots of the Code	26
6.3	Back Ends Representation	28
6.4	Result of the model	32
7	Conclusion and Future scope	33

1 Introduction

1.1 What is the idea?

The term Driver Drowsiness Detection system refers to in-vehicle systems that monitor driver and/or vehicle behaviour. These systems monitor the performance of the driver, and provide alerts or stimulation if the driver seems to be impaired. It warns drivers when they are getting drowsy.

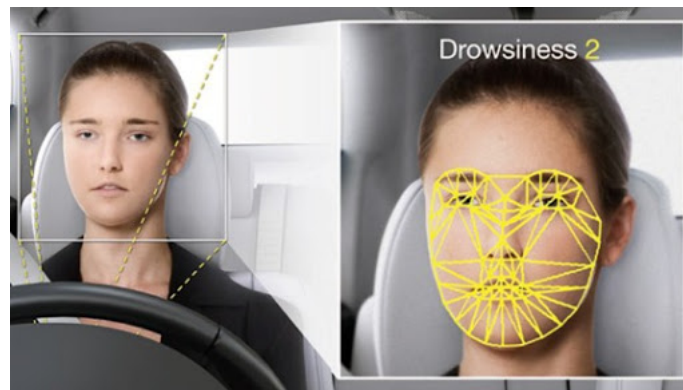


Figure 1: Face Detection

1.2 Project Category?

This Project is based on a very important part of Artificial Intelligence which is known as Deep Learning. There are many parts present in Deep Learning example- CNN, ANN, RNN. What we will be using is CNN (Convolutional Network) as it is best suited for image processing we will also be using Cascade classifier openCV for the training of our model. A image will be detected with the help of lot of filters which will be done by keras library.

1.3 Objective

1.3.1 Main Objectives

1. To filter the image:

We will filter the given image. A 2D image is converted into matrices with the

value of 0 and 1 and then we will apply filters on it to get its vertical and horizontal edges. These images will give us the whole image which we have to work upon.

2. To train the model:

We will train our model with the help of back propagation and with the help of optimizers like RMSprop which will help us to get our desired result by reducing the loss rate.

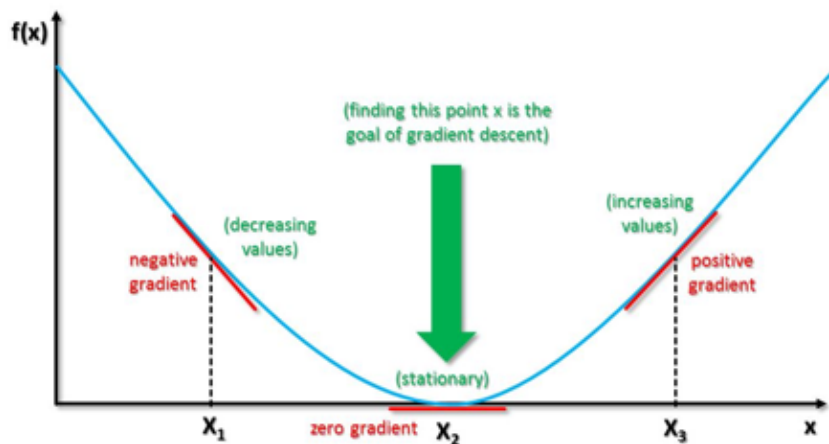


Figure 2: Gradient Descent

3. To tell the state:

At last our model will give output about Drowsiness or Undrowsiness with the help of the images which is entirely the main motive of our model.

1.4 Problem and Identification

What is the Need?:

Drowsy driving is an important, but often unrecognized, traffic safety problem. NHTSA estimates that drowsiness contributes to more than 100,000 collisions each year, resulting in over 1,500 deaths and 40,000 injuries.

Drowsiness increases the impairment caused by alcohol. Teenagers, professional drivers (including truck drivers), military personnel on leave, and shift workers are at particular risk. Drowsiness appears in situations of stress and fatigue in an

unexpected and inopportune way and may be produced by sleep disorders, certain types of medications, and even boredom, for example, driving for long periods of time. The sleeping sensation reduces the level of vigilance producing dangerous situations and increases the probability of an accident occurring.

It has been estimated that drowsiness causes between 10percent and 20percent of traffic accidents, causing both fatalities and injuries, whereas within the trucking industry 57percent of fatal truck accidents are caused by this problem. Fletcher et al. have stated that 30percent of all traffic accidents have been caused by drowsiness, and Brandt et al. have presented statistics showing that 20percent of all accidents are caused by fatigue and lack of attention. In the USA, drowsiness is responsible for 100000 traffic accidents yearly producing costs of close to 12,000 million dollars. In Germany, one out of four traffic accidents originate from drowsiness, while in England 20percent of all traffic accidents are produced by drowsiness, and in Australia 1500 million dollars has been spent on fatalities resulting from this problem.

1.5 Some Facts!

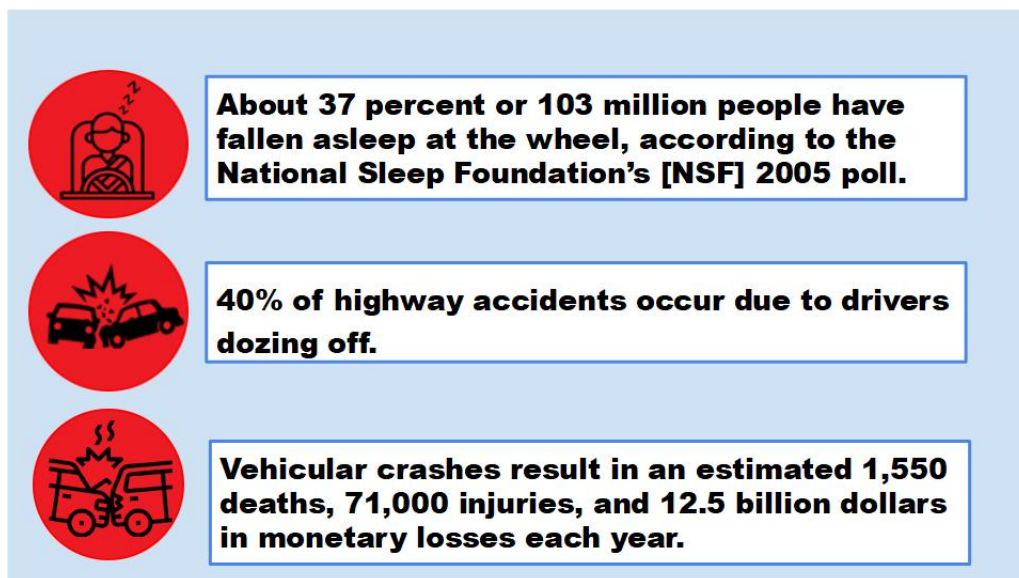


Figure 3: Facts to back our model

Drowsy Driving Problem:

Drowsy drivers may cause nearly a third of all deadly car crashes according to federal statistics and a study at the University of Pennsylvania Health System proves the danger. According to Dr. Michael Grandner “A lot of people don’t realize that more serious crashes are caused by falling asleep at the wheel than alcohol”. “Though we’ve all heard of drunk driving, we haven’t heard much about drowsy driving, but it’s a major health problem and safety problem.” Dr. Grandner, a member of the Center for Sleep and Circadian Neurobiology, says a survey of more than 17,000 people showed that most people need at least seven hours of sleep each night. While distracted driving has been getting a lot of attention lately, drowsy driving remains a major risk for motor vehicle crashes.

Two out of every five drivers (41percent) admit to having fallen asleep at the wheel at some point. One in ten said they have done so in the past year according to a new AAA Foundation for Traffic Safety Study.

One in six (16.5percent) of deadly crashes, one in eight of crashes resulting in a hospitalization, and one in eight out of fourteen crashes in which a vehicle had to be towed involved a drowsy driver. The National Highway Safety Administration estimates that drowsy driving results in 1,550 deaths, 71,000 injuries and more than 100,000 accidents per year.

More than half (55percent) of drivers who reported falling asleep while driving in the past year said they had been driving for less than one hour before falling asleep.

Many traffic researchers believe drowsy driving has been under-reported and underestimated.



Figure 4: Drowsy state is dangerous while driving

1.6 What has been done:

1. Simple driver strategies:

Clever signs are one of the most popular ways that the government has used to mitigate drowsy driving (Border Roads Organisation, 2016). Their signs have fun sayings on them intended to catch drivers attention.

2. Vibrations and alerts:

Multiple studies have explored the use of vibrations or alarms to alert drivers of their drowsiness. One particular study of 25 participants was conducted using a prototype steering wheel with vibration capability and a loud buzzer. When a driver displayed signs of drowsiness, the wheel would vibrate and the buzzer would sound, alerting the driver of their drowsy behavior. In general, subjects reported that the buzzer was more effective in alerting drivers, but also significantly more annoying than the vibrations. The study was based on a relatively small sample and produced mostly qualitative data, so the study is relatively incomplete if used on its own to propose a broader solution to drowsy driving. Despite the small sample, the study shows potential options for alert systems.

3. Blue light: With the introduction of blue light, the body becomes more alert

because the light suppresses the creation of melatonin and shifts circadian rhythms as much as three hours with the right amount of exposure (Harvard Health Letter, 2012). In a study, the introduction of a constant blue light with a wavelength of 468 nm led to a decrease in the amount of times the driver crossed the centerline while driving by around 45percent, as compared to the placebo given to participants.

1.7 Proposed System

A model which after training alarms the user with the help of some beep sound or with some message.CNN(Convolutional neural network) of Deep learning will be used with the help of tensorflow framework and keras library

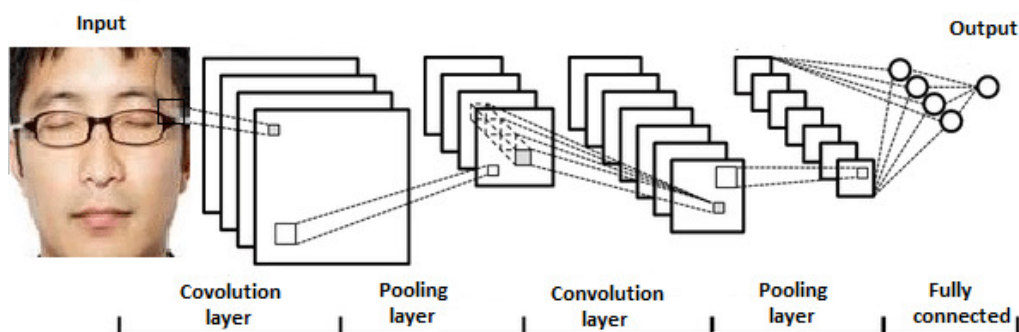


Figure 5: Schematic Diagram of CNN

1.8 Unique features of the system

1. 3D volumes of neurons. The layers of a CNN have neurons arranged in 3 dimensions: width, height and depth.[60] Where each neuron inside a convolutional layer is connected to only a small region of the layer before it, called a receptive field. Distinct types of layers, both locally and completely connected, are stacked to form a CNN architecture.
2. Local connectivity: following the concept of receptive fields, CNNs exploit spatial locality by enforcing a local connectivity pattern between neurons of adjacent layers. The architecture thus ensures that the learned "filters" produce the strongest response to a spatially local input pattern. Stacking many such layers

leads to non-linear filters that become increasingly global (i.e. responsive to a larger region of pixel space) so that the network first creates representations of small parts of the input, then from them assembles representations of larger areas.

3. Shared weights: In CNNs, each filter is replicated across the entire visual field. These replicated units share the same parameterization (weight vector and bias) and form a feature map. This means that all the neurons in a given convolutional layer respond to the same feature within their specific response field. Replicating units in this way allows for the resulting activation map to be equivariant under shifts of the locations of input features in the visual field, i.e. they grant translational equivariance - given that the layer has a stride of one.

4. Pooling: In a CNN's pooling layers, feature maps are divided into rectangular sub-regions, and the features in each rectangle are independently down-sampled to a single value, commonly by taking their average or maximum value. In addition to reducing the sizes of feature maps, the pooling operation grants a degree of local translational invariance to the features contained therein, allowing the CNN to be more robust to variations in their positions

2 Requirement Analysis and System Specification

2.1 Feasibility Study

2.1.1 Is this plan technically feasible?

Drowsiness detection system will need different images for image recognition method and can be implemented with the help of advance algorithms of Deep learning so it is quite feasible to make and it is definitely technically feasible.

2.1.2 Is this system Operationally feasible?

Will Drowsiness detection system completely solve the problem which we are intended to solve? We can't say for sure that it will solve all the problems but it will definitely help in reducing the number of accidents and other problems caused by drowsiness during driving.

2.1.3 Is this system Economically feasible?

Initial this system will only need software things and algorithm which doesn't take any of our expenses and for complete project we will need raspberry pi which will cost some money but overall it is quite economically feasible.

2.2 Software or Data Requirements

2.2.1 Images

For initial part of project we will train our model with the help of some images and we will give some images as input to get our desired result. After upload image path we will get result as 'Drowsiness or Undrowsiness' written over the particular image and that's how we will know if a person is drowsy or not.

2.2.2 Tensorflow framework

TensorFlow is a framework created by Google for creating Deep Learning models. Deep Learning is a category of machine learning models (=algorithms) that use multi-layer neural networks.

Machine Learning has enabled us to build complex applications with great accuracy. Whether it has to do with images, videos, text or even audio, Machine Learning can solve problems from a wide range. Tensorflow can be used to achieve all of these applications.

The reason for its popularity is the ease with which developers can build and deploy applications. The GitHub projects which we'll look closer at due the next parts are very powerful but also so easy to work with. Moreover, Tensorflow was created with processing power limitations in mind. The library can be ran on computers of all kinds, even on smartphones (yes, even on that overpriced thing with half an apple on it). I can guarantee you, working on a Intel Core i3 with 8 GB of RAM, you won't have performance issues.

2.2.3 Keras library

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear actionable error messages. It also has extensive documentation and developer guides.

Keras is the most used deep learning framework among top-5 winning teams on Kaggle. Because Keras makes it easier to run new experiments, it empowers you to try more ideas than your competition, faster. And this is how you win.

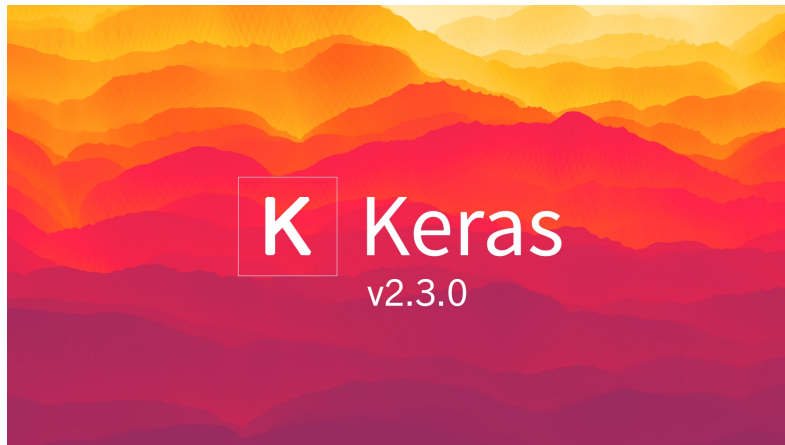


Figure 6: Kera Library

2.3 Validation

We will label the images of training data and also give an another dataset know as validation where we will validate the result of the data which will created by our model

2.4 Expected Hurdles

2.4.1 Classification of Images with different Positions:

One of many challenges in the field of computer vision is to deal with the variance in the data present in the real world. Human visual system can identify images :

1. Under different angles
2. Under different backgrounds
3. Under several different lighting conditions.

Convolutional Neural networks (CNN) have great performance while classifying images which are very similar to the dataset . However, If the images contain some degree of tilt or rotation then CNNs usually have difficulty in classifying the image



Figure 7: Multiple Face Detected

2.4.2 Minor Drawbacks of CNN:

1. A Convolutional neural network is significantly slower due to an operation such as maxpool.
2. If the CNN has several layers then the training process takes a lot of time if the computer doesn't consist of a good GPU.
3. A ConvNet requires a large Dataset to process and train the neural network.

The second category includes methods based on driver behaviour, which evaluate variations in the lateral position of the vehicle, in the velocity, in the steering wheel angle and in other signals recorded. The advantage of these approaches is that the signal is meaningful and the signal acquisition is quite easy. These procedures require a considerable amount of time to analyse user behaviour and, therefore, they do not work with the so called micro-sleeps—when a drowsy driver falls asleep for a few seconds on a very straight road section without changing the vehicle signals.

The third category includes methods based on driver visual analysis using image processing techniques. Computer vision can be a natural and non-intrusive technique for monitoring driver's sleepiness from the images taken by some cameras placed in front of the user

3 System Design

3.1 Model Architecture

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 64)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
dropout (Dropout)	(None, 14, 14, 64)	0
conv2d_1 (Conv2D)	(None, 14, 14, 32)	8224
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 32)	0
dropout_1 (Dropout)	(None, 7, 7, 32)	0
flatten (Flatten)	(None, 1568)	0
dense (Dense)	(None, 256)	401664
dropout_2 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 10)	2570
Total params: 412,778		
Trainable params: 412,778		
Non-trainable params: 0		

Figure 8: Architecture of our Model

Here we are continuously doing steps conv2d and max pooling to get the image with the help of filters which will give us vertical edges and horizontal edges.

3.1.1 Convolution:

Convolutional neural networks apply a filter to an input to create a feature map that summarizes the presence of detected features in the input.

Filters can be handcrafted, such as line detectors, but the innovation of convolutional neural networks is to learn the filters during training in the context of a specific prediction problem.

3.1.2 Max pooling:

Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

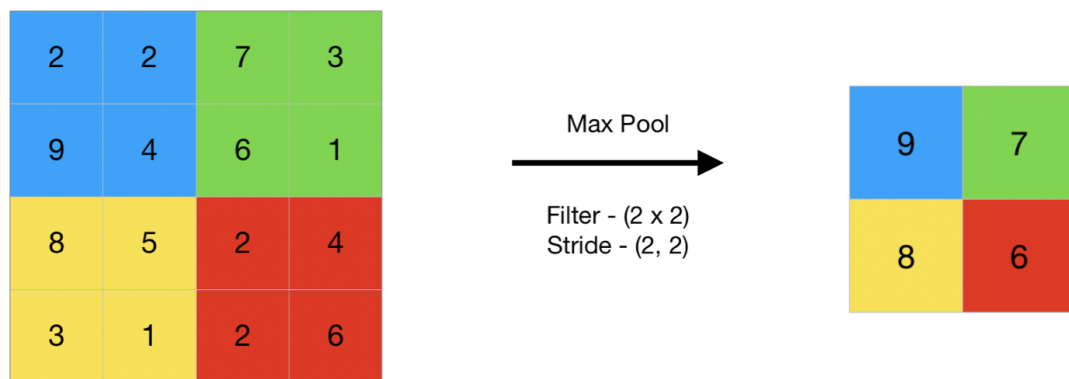


Figure 9: Diagram of Pooling

3.1.3 Dropout:

The term “dropout” refers to dropping out units (both hidden and visible) in a neural network.

Simply put, dropout refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random. By “ignoring”, I mean these units are not considered during a particular forward or backward pass.

More technically, At each training stage, individual nodes are either dropped out of

the net with probability $1-p$ or kept with probability p , so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed.

3.1.4 Flattening

Once the pooled featured map is obtained, the next step is to flatten it. Flattening involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing.

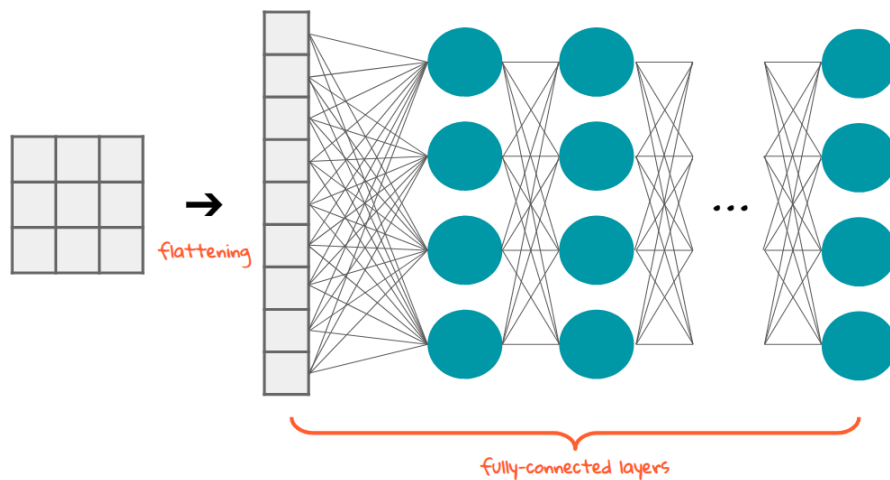


Figure 10: Diagram of Flattening

3.1.5 Dense Layer

What is a Dense Layer in Neural Network? The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most commonly used layer in the models.

Dense Layer in Keras

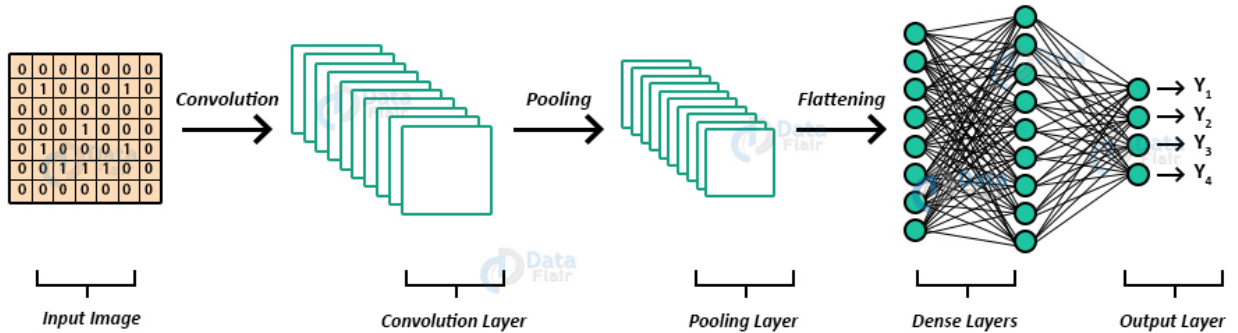


Figure 11: Layers in Keras Library

3.2 User Interface Design

This Model is generally based on the input output basis. That means we will provide some input as images and get some output in which 'Drowsiness and Undrowsiness' will be written on the images itself. So, there is not user interface or webpage for this but it can be expanded once we are done with our basic training of data.

3.3 Database designing

3.3.1 Requirement Analysis

Planning:

We need a powerful and loaded system to train our model and launch it if there is not availability of powerful system we can also use Google colaboratory.

3.3.2 Database

Logical Model:

We will train our data through various step of convolution,max pooling,flattening,dropout

etc to build a logical thinking to our model to perform the task we have given it.

Physical Model:

A large dataset should be provided to the model to train the model the more the merrier to make model understand each and every circumstance and perform better.

3.3.3 Implementation

Testing:

We will test our model with giving the path of testing image in the code itself and then we will get our desired result if the model is trained properly.

3.4 Methodology

In 2012, a revolution occurred: during the annual ILSVRC computer vision competition, a new Deep Learning algorithm exploded the records! It's a convolutional neural network called Alexnet.

Convolutional neural networks have a methodology similar to that of traditional supervised learning methods: they receive input images, detect the features of each of them, and then drag a grader on it.

However, features are learned automatically! The CNN themselves carry out all the tedious work of extracting and describing features: during the training phase, the classification error is minimized to optimize the parameters of the classifier AND the features!

Convolutional neural networks refer to a sub-category of neural networks: they, therefore, have all the characteristics of neural networks. However, CNN is specifically designed to process input images. Their architecture is then more specific: it is composed of two main blocks.

The first block makes the particularity of this type of neural network since it functions as a feature extractor. To do this, it performs template matching by applying convolution filtering operations. The first layer filters the image with several convolution kernels and returns "feature maps", which are then normalized (with an activation function) and/or resized.

This process can be repeated several times: we filter the features maps obtained with new kernels, which gives us new features maps to normalize and resize, and we can filter again, and so on. Finally, the values of the last feature maps are concatenated into a vector. This vector defines the output of the first block and the input of the second.

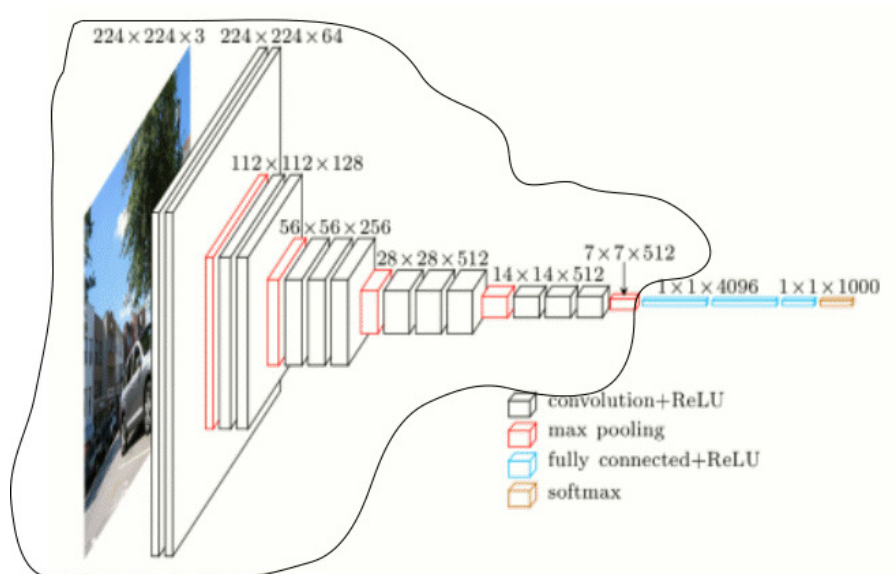


Figure 12: CNN Block1

The second block is not characteristic of a CNN: it is in fact at the end of all the neural networks used for classification. The input vector values are transformed (with several linear combinations and activation functions) to return a new vector to the output. This last vector contains as many elements as there are classes: element i represents the probability that the image belongs to class i . Each element is therefore between 0 and 1, and the sum of all is worth 1. These probabilities are calculated by the last layer of this block (and therefore of the network), which uses a logistic function (binary classification) or a softmax function (multi-class classification) as an activation function.

As with ordinary neural networks, the parameters of the layers are determined by gradient backpropagation: the cross-entropy is minimized during the training phase. But in the case of CNN, these parameters refer in particular to the image features.

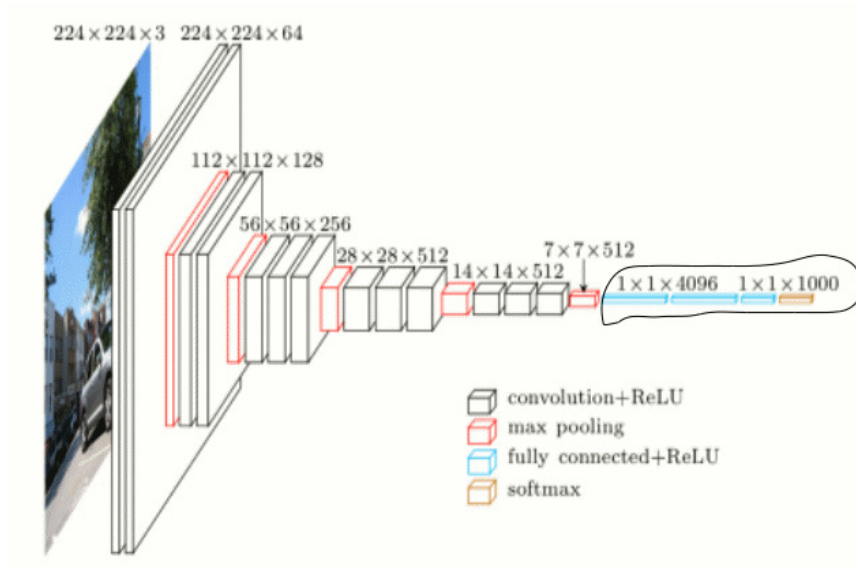


Figure 13: CNN Block2

4 Implementation, Testing and Maintenance

4.1 Tools, Languages etc.

We will use language python and to get multidimensional array we will use tensorflow and on tensorflow we will use library keras which performs almost all the tasks needed in convolutional neural network .Python

.Tensorflow

.Keras

.Matplotlib

5 Coding standards of language

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability

and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

6 Result and Discussion

6.1 User Interface Representation

We will get result based on our input and a output will be shown. This all will be happening in Google Colaboratory as it is a very good platform to do Deep learning and machine learning projects as it can provide us required GPU which we needed for this project and also less powerful systems are able to do such high level computational work.

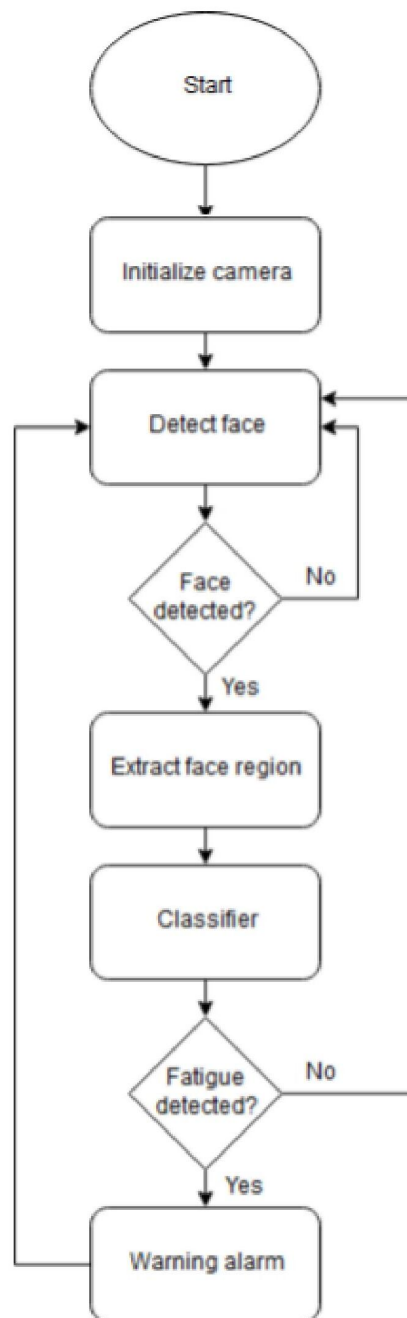


Figure 14: Flowchart of our Model

6.2 Screenshots of the Code



The screenshot shows a Jupyter Notebook titled 'loadmodel.ipynb'. The code in the first cell imports tensorflow as tf, cv2, matplotlib.pyplot as plt, and numpy as np. The second cell imports drive from google.colab and mounts the drive at /content/drive. The third cell defines labels as ['drowsiness', 'undrowsiness'], loads a model from a Google Drive path, and prints the input shape, which is (None, 150, 150, 3). The fourth cell calls model.summary(). The output shows the model name 'sequential_1' and a table with columns 'Layer (type)', 'Output Shape', and 'Param #'. The table is empty. The interface includes a top bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help' menus, and a bottom bar with 'Connect', 'Editing', and 'Activate Windows' buttons.

```
[ ] import tensorflow as tf
import cv2
import matplotlib.pyplot as plt
import numpy as np

from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[ ] labels = ["drowsiness", "undrowsiness"]
model = tf.keras.models.load_model('/content/drive/MyDrive/Minor project/model/model_10June_15epoch.h5')
print(model.input.shape)

(None, 150, 150, 3)

[ ] model.summary()

Model: "sequential_1"
Layer (type) Output Shape Param #
```

Figure 15: First Shell of Code Defining Libraries

Importing tensorflow which is the library, Matplotlib which is a library for numerical calculation and plotting graphs, cv2 for image processing and numpy which is the extension of Matplotlib

```

def get_label_index(pre):
    if pre[0][0] > 0.5:
        return 1
    else:
        return 0

[ ]
def display_multiple_img(images, rows = 1, cols=1):
    figure, ax = plt.subplots(nrows=rows,ncols=cols,figsize=(10,10))
    # figure.tight_layout(pad=2.0)
    for ind,detail in enumerate(images):
        # print(ind,detail)
        ax.ravel()[ind].imshow(detail["img"])
        ax.ravel()[ind].set_title(detail["label"])
        ax.ravel()[ind].set_axis_off()
    plt.tight_layout()
    plt.savefig("out.png")
    plt.show()

[ ] from tensorflow import keras
import os

```

Figure 16: Second shell of code telling about image display

```

[ ] from tensorflow import keras
import os
DIR = "/content/drive/MyDrive/Minor project model/test images"
face_cascade = cv2.CascadeClassifier("/content/drive/MyDrive/Minor project model/haarcascade/haarcascade_frontalface_alt2.xml")
IMG_label = []
for imgFile in os.listdir(DIR):
    img_path = os.path.join(DIR,imgFile)
    img = keras.preprocessing.image.load_img(img_path,target_size =(150,150))

    img_array = keras.preprocessing.image.img_to_array(img)
    gray = cv2.cvtColor(np.float32(img_array), cv2.COLOR_BGR2GRAY)
    gray = np.array(gray, dtype='uint8')
    faces = face_cascade.detectMultiScale(gray, scaleFactor = 1.1, minNeighbors = 5,minSize = (30,30))
    for (x,y,w,h) in faces:
        # print(x,y,w,h)
        roi_image = img_array[y:y+h,x:x+w]
        roi_image = cv2.resize(roi_image , (150 , 150))
        roi_image = np.array(roi_image,dtype="float64")
        roi_image = roi_image.reshape((-1,150,150,3))
        pred = model.predict([roi_image])
        label_index = get_label_index(pred)
        label = labels[label_index]
        IMG_label.append({"img" : img,"label" : label})

print(len(IMG_label))

```

Figure 17: Third shell of code about Face Detection

In the Last part of code we are importing opencv for face detection and to

append the image and label it accordingly we can use multiple images and just have provide proper rows and columns thats it our model will be up and running

6.3 Back Ends Representation

Steps To Train our Model:

The screenshot shows the Jupyter Notebook interface for a file named "train-model.ipynb". The top bar includes standard menu items like File, Edit, View, Insert, Runtime, Tools, and Help, along with a status indicator "Last saved at 4:13 PM". On the right side of the top bar are icons for Comment, Share, Settings, and a user profile icon.

The left sidebar contains navigation icons for a home-like view, a file explorer, a search function, and code navigation symbols (back, forward, up, down). Below these are icons for running all cells, running the current cell, and interrupting the kernel.

The main area displays the notebook's code cells. The first cell contains import statements for tensorflow as tf, os, cv2, and pickle. The second cell defines directory paths d_dir and ud_dir, lists their contents into d_img and ud_img, and prints some of them. The third cell shows a list of image filenames. The fourth cell begins defining a Keras Sequential model with Conv2D and MaxPooling2D layers.

A watermark "Activate Windows Go to Settings to activate Windows." is visible in the bottom right corner of the application window.

Figure 18: Importing Libraries

Importing all important libraries tensorflow,os,cv2,pickle and getting data from libraries drowsiness and undrowsiness.

```

model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(.2),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

[ ] model.summary()

```

Layer (type)	Output Shape	Param #

conv2d_4 (Conv2D)	(None, 148, 148, 32)	896

max_pooling2d_4 (MaxPooling2D)	(None, 74, 74, 32)	0

Figure 19: Model Design

Here we can see our model architecture to train the model

```

[ ] =====
Total params: 3,453,121
Trainable params: 3,453,121
Non-trainable params: 0
=====

[ ] from tensorflow.keras.optimizers import RMSprop
model.compile(optimizer=RMSprop(lr=0.001), loss = 'binary_crossentropy', metrics = ['accuracy'])

# model.compile(optimizer=RMSprop(lr=0.001),
#               loss='binary_crossentropy',
#               metrics = ['accuracy'])

[ ] from tensorflow.keras.preprocessing.image import ImageDataGenerator
base_dir = "data_faces2"
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')

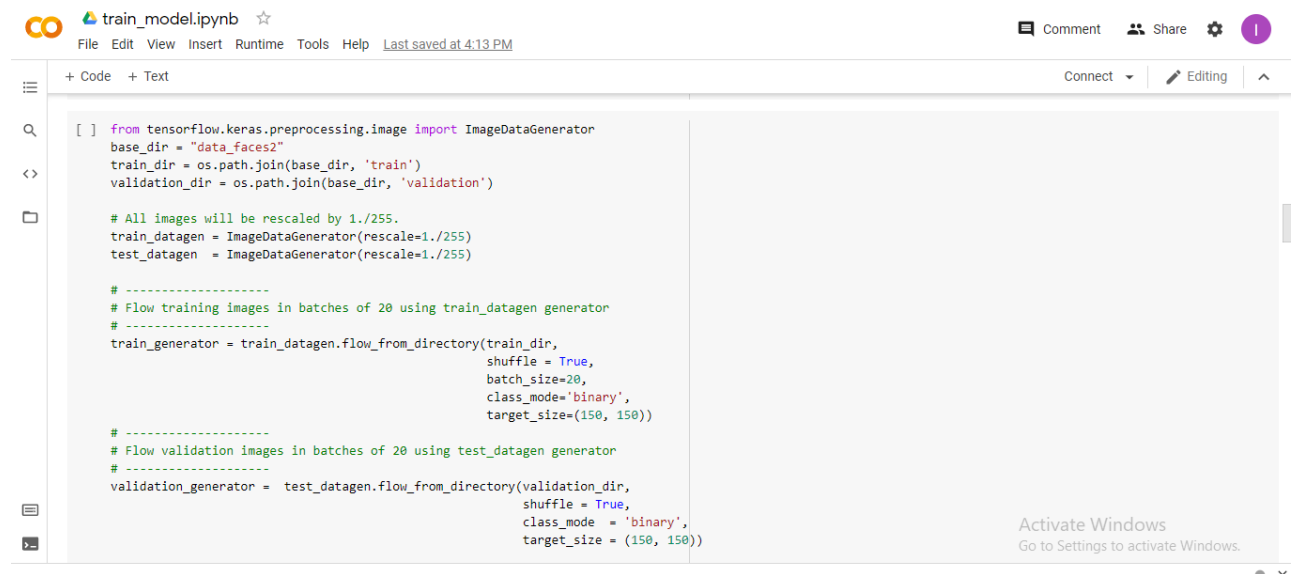
# All images will be rescaled by 1./255.
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

# -----
# Flow training images in batches of 20 using train_datagen generator
# -----
train_generator = train_datagen.flow_from_directory(train_dir,

```

Figure 20: Adding optimizers and Loss

Here we are optimizing our data and giving values to loss and metrics.



The screenshot shows a Jupyter Notebook titled 'train_model.ipynb' with a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar (Last saved at 4:13 PM). The code in the notebook is as follows:

```
[ ] from tensorflow.keras.preprocessing.image import ImageDataGenerator
base_dir = "data_faces2"
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')

# All images will be rescaled by 1./255.
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

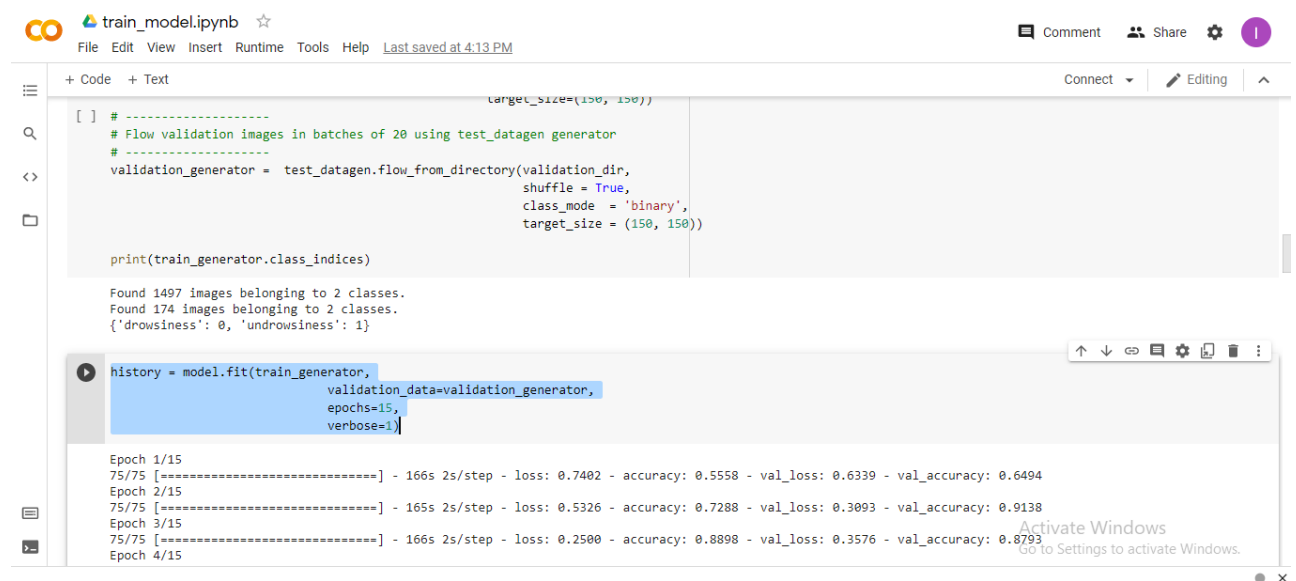
# -----
# Flow training images in batches of 20 using train_datagen generator
# -----
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    shuffle = True,
                                                    batch_size=20,
                                                    class_mode='binary',
                                                    target_size=(150, 150))

# -----
# Flow validation images in batches of 20 using test_datagen generator
# -----
validation_generator = test_datagen.flow_from_directory(validation_dir,
                                                        shuffle = True,
                                                        class_mode = 'binary',
                                                        target_size = (150, 150))
```

An 'Activate Windows' watermark is visible in the bottom right corner of the code editor.

Figure 21: Training and Validating Model

Here we are generating more kind of images for our model and also validating our image



The screenshot shows a Jupyter Notebook titled 'train_model.ipynb'. The code in the cell defines a validation generator and runs a model fit. The output shows the number of images found and the training progress over 4 epochs.

```
[ ] # -----
# Flow validation images in batches of 20 using test_datagen generator
# -----
validation_generator = test_datagen.flow_from_directory(validation_dir,
                                                        shuffle = True,
                                                        class_mode = 'binary',
                                                        target_size = (150, 150))

print(train_generator.class_indices)

Found 1497 images belonging to 2 classes.
Found 174 images belonging to 2 classes.
{'drowsiness': 0, 'undrowsiness': 1}

history = model.fit(train_generator,
                    validation_data=validation_generator,
                    epochs=15,
                    verbose=1)
```

Epoch 1/15
75/75 [=====] - 166s 2s/step - loss: 0.7402 - accuracy: 0.5558 - val_loss: 0.6339 - val_accuracy: 0.6494
Epoch 2/15
75/75 [=====] - 165s 2s/step - loss: 0.5326 - accuracy: 0.7288 - val_loss: 0.3093 - val_accuracy: 0.9138
Epoch 3/15
75/75 [=====] - 166s 2s/step - loss: 0.2500 - accuracy: 0.8898 - val_loss: 0.3576 - val_accuracy: 0.8793
Epoch 4/15

Figure 22: To run and Give epochs

Model fit is used for plot graphs if we want to show some data about our model

All this training is happening at the backend where are executing our architecture and optimizing our model with the help of Rmsprop to reduce the loss rate we are also using imagegenerator to get different kind of images to train our model even better way.

6.4 Result of the model

Now after all the training and all the coding we need to see some results as our model suggest if the value is given as 0-0.5 the peson will be drowsy and if the value is greater then 0.5 then the person will be undrowsy. So lets try our model on the input given as two images one is drowsy and other is undrowsy.

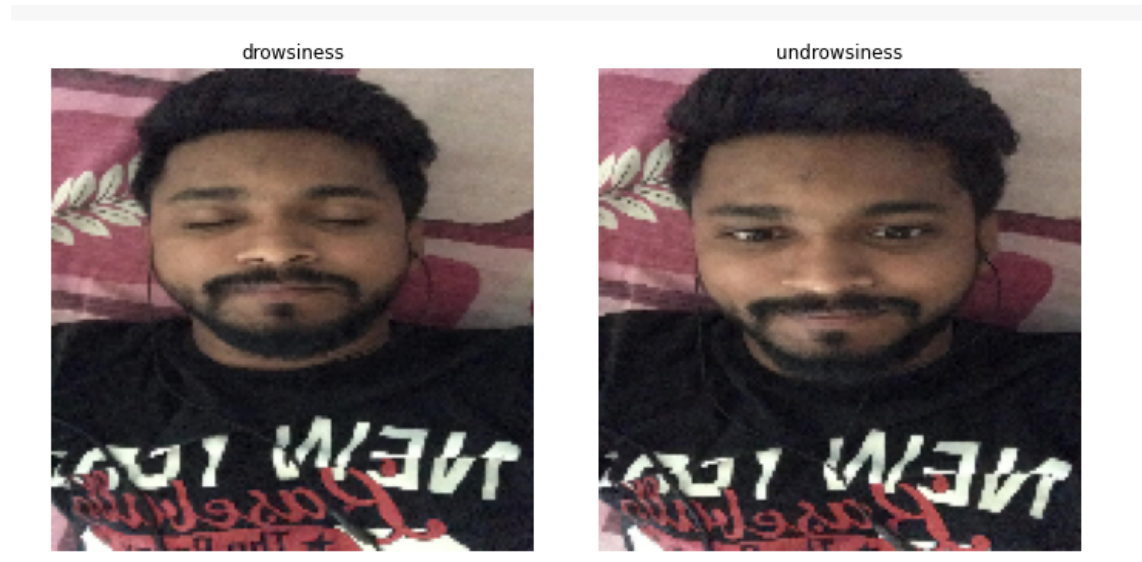


Figure 23: Result of our Model

So as we can see the model is correctly predicting the drowsy or undrowsy state of a person so we can say that it is working fine.

7 Conclusion and Future scope

Deep learning is a concept which is not so old and still growing rapidly and CNN(Convolutional Neural Network) is an important part of it which is very beneficial to recognize and image and solve some complex problems.

Our model here is trying to solve a problem which results in the death of thousands of people every year 'Drowsiness'. On roads even if you lose your concentration for one second disastrous things could happen so a model telling about the drowsy state of a person is a good and revolutionary system.

For future scope we want to make this model a fully functional device. Right now this model is a static model taking input and giving output.

We want to make it more dynamic a model which will capture the image with the help of a web cam and after recognizing the image gives us a result of either 'Drowsy' or 'Undrowsy'. We will also add an alarm beep which is a beeping sound in case the person is in a drowsy state.

We will also use a Raspberry Pi to make it in order to make it a fully functional device in which a bit of IOT will also be used.

References

- [1] Messaoud Doudou, Abdelmadjid Bouabdallah. Performance Specifications of Market Physiological Sensors for Efficient Driver Drowsiness Detection System. 7th International Conference on Sensor Networks (SENSORNETS 2018), Jan 2018, Funchal, Madeira, Portugal. Ffhal-01649081
- [2] Esra Vural and Mujdat Cetin and Aytul Ercil and Grew Littlewort and Marian Bartlett and Javier Movellan. Drowsy Driver Detection Through Facial Movement Analysis
- [3] Venkata Rami Reddy Chirra and Srinivasulu Reddy Uyyala and Venkata Krishna Kishore Kolli. A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State. IIETA
- [4] Kartik Dwivedi, Kumar Biswaranjan and Amit Sethi. Drowsy Driver Detection using Representation Learning
- [5] Kusuma Kumari B. M. Review on Drowsy Driving: Becoming a Dangerous Problem. International Journal of Science and Research (IJSR)
- [6] Sanjay Kumar Singh. Road Traffic Accidents in India: Issues and Challenges. World Conference on Transport Research -WCTR 2016 Shanghai.
- [7] Flores, Marco Armingol, J.M. de la Escalera, Arturo. (2008). Real-time drowsiness detection system for an intelligent vehicle. IEEE Intelligent Vehicles Symposium, Proceedings. 637 - 642. 10.1109/IVS.2008.4621125.

- [8] Bn, Manu. (2016). Facial Features Monitoring for Real Time Drowsiness Detection. 10.1109/INNOVATIONS.2016.7880030.
- [9] Hossain, Md. Yousuf George, Fabian Parsia. (2018). IOT Based Real-Time Drowsy Driving Detection System for the Prevention of Road Accidents. 190-195. 10.1109/ICIIBMS.2018.8550026.
- [10] Understanding Convolutional Neural Networks with A Mathematical Model - C.-C. Jay Kuo
- [11] Deep Convolutional Neural Network for Image Deconvolution- Li Xu
- [12] Recent Advances in Convolutional Neural Networks - Jiuxiang Gua, Zhenhua Wangb, Jason Kuenb
- [13] Real-Time Driver Drowsiness Detection for Embedded System Using Model Compression of Deep Neural Networks - Bhargava Reddy, Ye-Hoon Kim
- [14] Driver Fatigue Detection Based on Eye Tracking - Mandalapu Sarada Devi ; Preeti R. Bajaj
- [15] Eye tracking based driver fatigue monitoring and warning system - Hardeep Singh ; J. S. Bhatia ; Jasbir Kaur
- [16] Driver drowsiness recognition based on computer vision technology - Wei Zhang ; Bo Cheng ; Yingzi Lin

- [17] Head movement-based driver drowsiness detection: A review of state-of-art techniques - Ajay Mittal, Kanika Kumar, Sarina Dhamija , Manvjeet Kaur
- [18] Drowsy Driver Detection Through Facial Movement Analysis - Esra Vural- Mujdat Cetin Aytul Ercil Gwen Littlewort Marian Bartlett Javier Movellan
- [19] Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review - Waseem Rawat and Zenghui Wang
- [20] Neural network-based face detection - H.A. Rowley , S. Baluja , T. Kanade
- [21] P. Fischer, J. Adkins, D. Davila, Ch. DeWeese, V. Harper, and J. Stephen Higgins. "wake up call! understanding drowsy driving and what states can do", 2016.
- [22] P. Viola and M. Jones. Robust real-time face detection. In Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, volume 2, pages 747–747, 2001
- [23] <https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb>
- [24] <https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148>
- [25] <https://medium.com/@tifa2up/image-classification-using-deep-neural-networks-a-beginner-friendly-approach-using-tensorflow-94b0a090ccd4>

- [26] <https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html>
- [27] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3571819/>
- [28] <https://www.medindia.net/patientinfo/drowsy-driving.htm>
- [29] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3571819>
- [30] <https://www.dailymail.co.uk/sciencetech/article-2238581/How-blue-light-car-good-coffee-keeping-alert-wheel.html>
- [31] <https://timesofindia.indiatimes.com/india/india-way-off-road-safety-targets-for-2020-road-accidents-still-kill-over-a-lakh-a-year/articleshow/65765549.cms>
- [32] https://en.wikipedia.org/wiki/Driver_drowsiness_detection
architecture – of – convolutional – neural – networks – simplified – demystified