DROWSINESS DETECTION

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

B Surya Charan Teja [RA2011003010852]
P Praveen Reddy [RA2011003010886]
K Sai Manikanta [RA2011003010893]

Under the guidance of

MS. M Rajalakshmi

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

It is certified that the "DROWSINESS DETECTION" mini project report was completed under my supervision by B. Surya Charan Teja (RA2011003010852), P. Praveen Reddy (RA201103010886), and K. Sai Manikanta (RA2011003010893). Furthermore, I certify that, to the best of my knowledge, the work described here does not constitute any other project report or dissertation on the basis of which this candidate or any other candidate has previously received a degree or prize.

SIGNATURE
MS. M Rajalakshmi
GUIDE
Assistant Professor
Department of Computing Technologies

SIGNATURE
Dr. M. Pushpalatha
HEAD OF THE DEPARTMENT
Professor & Head
Department of Computing Technologies

ABSTRACT

This project suggests an in-vehicle drowsiness detection system based on computer vision. The technology uses a dashboard-mounted camera to continuously monitor the driver's facial expressions and eye movements. To identify tiredness, machine learning systems track the frequency and length of eyelid closure. The device informs the driver when drowsiness is found, helping to reduce accidents brought on by fatigued driving. For automakers and fleet operators looking to improve road safety, the system's simplicity, price, and ease of integration make it an appealing option.

Driving when fatigued puts lives at risk by increasing the likelihood of collisions, injuries, and fatalities. This research introduces a computer vision-based sleepiness detection system that uses a dashboard-mounted camera to record and examine the driver's facial expressions and eye movements. The technology can precisely identify tiredness by tracking the frequency and length of eyelid closure. The device instantly informs the driver when tiredness is found, offering a useful safeguard against accidents brought on by drowsy driving. This system has a tremendous deal of potential for incorporation into vehicles due to its simple implementation and low cost, which will help fleet operators and automakers increase driver safety.

TABLE OF CONTENTS

ABSTRACT

CHAPTER 1 INTRODUCTION

1.1. INTRODUCTION 6

1.2. PROBLEM STATEMENT 6

1.3. REQUIREMENTS 7

1.3.1. SOFTWARE REQUIREMENTS: 7

1.3.2. HARDWARE REQUIREMENTS: 7

CHAPTER 2 LITERATURE SURVEY 8

2.1 PAPER 1 8

2.2 PAPER 2 9

2.3 PAPER 3 10

CHAPTER 3 SYSTEM ARCHITETURE AND DESIGN 12

CHAPTER 4. PROPOSED METHODOLOGY 13

4.1 EXISTING SYSTEM 13

4.2 PROPOSED SYSTEM 13

4.3 PROPOSED TECHNIQUES 13

4.3.1 ARTIFICIAL INTELLIGENCE (AI) 14

4.32 MACHINE LEARNING 14

4.3.3 DEEP LEARNING 16

4.3.4 NEURAL NETWORKS 16

4.3.5 PYTHON 18

4.3.6 JUPYTER NOTEBOOK 20

4.3.7 CONVOLUTIONAL NEURAL NETWORK 21

TRAINING 22

TESTING 23

LIBRARIES USED 24

- OPENCV (OPEN-SOURCE COMPUTER VISION LIBRARY) 24
- NUMPY 24

- KERAS 25
- TENSORFLOW 25
- MATPLOTLIB 26
- SEABORN 26
- SKLEARN 26
- OS MODULE IN PYTHON 27
- PLAYSOUND MODULE IN PYTHON 28

CHAPTER 5 CODING AND TESTING

5.1 IMPORTING ALL THE REQUIRED LIBRARIES 29

5.2 CODE FOR FACE DETECTION 29

5.3 CODE FOR READING IMAGES AFTER FACE DETECTION 30

5.4 CODE FOR TRAINING THE CNN 31

5.5 CODE FOR EVALUATING THE MODEL 32

5.6 CODE FOR PREDICTING THE IMAGES CAPTURED FROM THE CAMERA 34

CHAPTER 6 SCREEN SHOTS AND RESULTS 35

CHAPTER 7 CONCLUSION AND FUTURE SCOPE 36

CONCLUSION

FUTURE SCOPE 37

REFERENCES 38

CHAPTER 1

INTRODUCTION

1.1. Introduction

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 ofdistraction or the drowsiness of driver. The countless number of peopledrives for long distance every day and night on the highway. Drowsinessappears in situations of stress and fatigue in an unexpected and inopportune way, and it may be produced by sleep disorders, certain type of medications, and even, boredom situations, for example, driving for a long time. In this way, drowsiness produces danger situations and increases the probability that an accident occurs. In this context, it is important to use new technologies to design and to build systems that will monitor drivers, and measure their level of attention throughout thewhole driving process.

To prevent such accidents, our team has come up with a solution for this. In this system, a camera is used to record user's visual characteristics. We use face detection and CNN techniques and try to detect the drowsiness of driver, if he/she is drowsy then alarm will be generated. So that the driver will get cautious and take preventive measures. Driver drowsiness detection contributes to the decrease in the number of deaths occurring in traffic accident.

1.2. Problem Statement

Traffic accidents due to human errors cause many deaths and injuries around the world. The major cause of these accidents is drowsiness of the driver due to sleeplessness or long driving hours. Thereis need for a system developed with the technologies that are available today which can overcome this situation. The aim of this system is to reduce the number of accidents by developing a model which can generate an alert if the driver is feeling drowsy so that the driver can become aware and take necessary actions.

1.3. Requirements

1.3.1. Software Requirements:

These are the Software Configurations that are required.

- Operating System: Linux, Windows, Mac OS
- Language: Python 3
- IDE: Jupiter Notebook

1.3.2. Hardware Requirements:

These are the Hardware Configurations that are required.

- Processor
- Ram: 4GB
- Webcam
- speaker

CHAPTER 2

LITERATURE SURVEY

2.1. Paper 1

Title: Drowsiness Detection System Utilizing Physiological Signals.

Author: Trupti K. Dange, T. S. Yengatiwar.

Year of publication: 2013.

Keywords: EOG, ECG, EEG, HRV, SVM, driver drowsiness detection.

The Physiological parameters-based techniques detect drowsiness based on drivers' physical conditions such as heart rate, pulse rate, breathing rate, respiratory rate and body temperature, etc. These biological parameters are more reliable and accurate in drowsiness detection as they are concerned with what is happening with driver physically. Fatigue or drowsiness, change the physiological parameters such as a decrease in blood pressure, heart rate and body temperature, etc. Physiological parameters-based drowsiness detection systems detect these changes and alert the driver when he is in the state, near to sleep.

A list of physiological condition-based drowsiness detection system. These measures are invasive, so require electrodes to be directly placed on the driver's body.

1) EEG-BASED DRIVER FATIGUE DETECTION

The drivers' fatigue detection system using Electroencephalogram (EEG) signals is proposed to avoid the road accidents usually caused due to drivers' fatigue. The proposed method firstly finds the index related to different drowsiness levels. The system takes EEG signal as input which is calculated by a cheap single electrode neuro signal acquisition device. To evaluate the proposed method, data set for simulated car driver under the different levels of drowsiness is collected locally. And result shows that the proposed system can detect all subjects of tiredness.

2) WAVELET ANALYSIS OF HEART RATE VARIABILITY & SVM CLASSIFIER

Li and Chung [21] proposed the driver drowsiness detection that uses wavelet analysis of Heart Rate Variability (HRV) and Support Vector Machine (SVM) classifier. The basic purpose is to categorize the alert and drowsy

drivers using the wavelet transform of HRV signals over short durations. The system firstly takes Photo Plethysmo Graphy (PPG) signal as input and divideit into 1-minute intervals and then verify two driving events using average percentage of eyelid closure over pupil over time (PERCLOS) measurement over the interval. Secondly, the system performs the feature extraction of HRV time series based on Fast Fourier Transform (FFT) and wavelet. A Receiver Operation Curve (ROC) and SVM classifier is used for feature extraction and classification respectively. The analysis of ROC shows that the wavelet-based method gives improved results than the FFT-based method.
Finally, the real time requirements for drowsiness detection, FFT and wavelet features are used to train the SVM classifier extracted from the HRV signals.

3) PULSE SENSOR METHOD

Mostly, previous studies focus on the physical conditions of drivers to detect drowsiness. That's why Rahim detects the drowsy drivers using infrared heart-rate sensors or pulse sensors. The pulse sensor measures the heart pulse rate from drivers' finger or hand. The sensor is attached with the fingeror hand, detects the amount of blood flowing through the finger. Thenamount of the blood's oxygen is shown in the finger, which causes the infrared light to reflect off and to the transmitter. The sensor picks up the fluctuation of oxygen that are connected to the Arduino as microcontroller. Then, the heart pulse rate is visualizing by the software processing of HRV frequency domain. Experimental results show that LF/HF (Low to highfrequency) ratio decreases as drivers go from the state of being awake to thedrowsy and many road accidents can be avoided if an alert is sent on time.

2.2. Paper 2

Title: Drowsiness Detection with OpenCV (Using Eye Aspect Ratio)

Author: Adrian Rosebrock.

Year of publication: 2017.

Keywords: EAR, SVM, eye blink detection.

A real-time algorithm to detect eye blinks in a video sequence from a standard camera is proposed. Recent landmark detectors, trained on in-the wild datasets exhibit excellent robustness against a head orientation with respect to a camera, varying illumination and facial expressions. We show that the landmarks are detected precisely enough to reliably estimate the level of the eye opening. The proposed algorithm therefore estimates the landmark positions, extracts a single scalar quantity – eye aspect ratio (EAR)
– characterizing the eye opening in each frame.

Many methods have been proposed to automatically detect eye blinks in a

video sequence. Several methods are based on motion estimation in the eye region. Typically, the face and eyes are detected by a Viola-Jones type detector. Next, motion in the eye area is estimated from optical flow, by sparse tracking, or by frame-to-frame intensity differencing and adaptive thresholding. Finally, a decision is made whether the eyes are or are not covered by eyelids. Nowadays, robust real-time facial landmark detectors that capture most of the characteristic points on a human face image, including eye corners and eyelids, are available, Most of the state-of-the-artlandmark detectors formulate a regression problem, where a mapping froman image into landmark positions or into other landmark parametrization islearned. These modern landmark detectors are trained on "in-the-wild datasets" and they are thus robust to varying illumination, various facial expressions, and moderate non-frontal head rotations

Proposed method

The eye blink is a fast closing and reopening of a human eye. Each individualhas a little bit different pattern of blinks. The pattern differs in the speed of closing and opening, a degree of squeezing the eye and in a blink duration. The eye blink lasts approximately 100-400 ms. We propose to exploit state-of-the-art facial landmark detectors to localize the eyes and eyelid contours.From the landmarks detected in the image, we derive the eye aspect ratio (EAR) that is used as an estimate of the eye-opening state. Since the per frame EAR may not necessarily recognize the eye blinks correctly, a classifier that takes a larger temporal window of a frame into account is trained. The EAR is mostly constant when an eye is open and is getting close to zero while closing an eye.

2.3. Paper 3

Title: Real Time Driver Fatigue Detection Based on SVM Algorithm.

Authors: Burcu Kir Savas, Yasar Becerkli.

Year of publication: 2018.

Keywords: fatigue detection , SVM , driving safety , video processing

PROPOSED SYSTEM

In this study, SVM based driver fatigue prediction system is proposed to increase driver safety. The proposed system has five stages: PERCLOS, countof yawn, internal zone of the mouth opening, count of eye blinking and headdetection to extract attributes from video recordings. The classification stage is done with Support Vector Machine (SVM). While the Yard dataset is used during the training phase of the classification, real-time video recordings areused during the test phase.

SVM is a classification algorithm separating data items. This algorithm proposed by Vladimir N. Vapnik based on statistical learning theory. SVM, one of the machine learning methods, is widely used in the field of pattern recognition. The main purpose of the SVM is to find the best hyperplane to distinguish the data given as two-class or multiclass. The study was carried out in two classes. Whereas label 0 means that the driver is tired, label 1 means the driver is non-tired. Thus, it is aimed to distinguish tired drivers (driver fatigue) from non- tired drivers. In training driving simulation experiments, we had 10 volunteers (5 men and 5 women) whose ages werebetween 18-30. All the volunteers drove training simulation during theexperiments. Attributions obtained in real time during the driving when fatigue detection system was working are indicated. The results of driver fatigue detection were classified using SVM classification algorithm. The total 713 datas line each of which has five stage features were sampled from 10 volunteers. All the data in the study are divided into two groups: 80% training data set (568 data set) and 20% test data set (145 data set). In the study, cross validation was selected as 10.

Measurements are as follows:

- TP means that a real fatigue is detected;
- TN means that a non-fatigue situation is correctly detected as non-fatigueby the system;
- FP means that a non-fatigue situation is incorrectly detected as real fatigue;
- FN means that a real fatigue situation is incorrectly detected as non-fatigue.

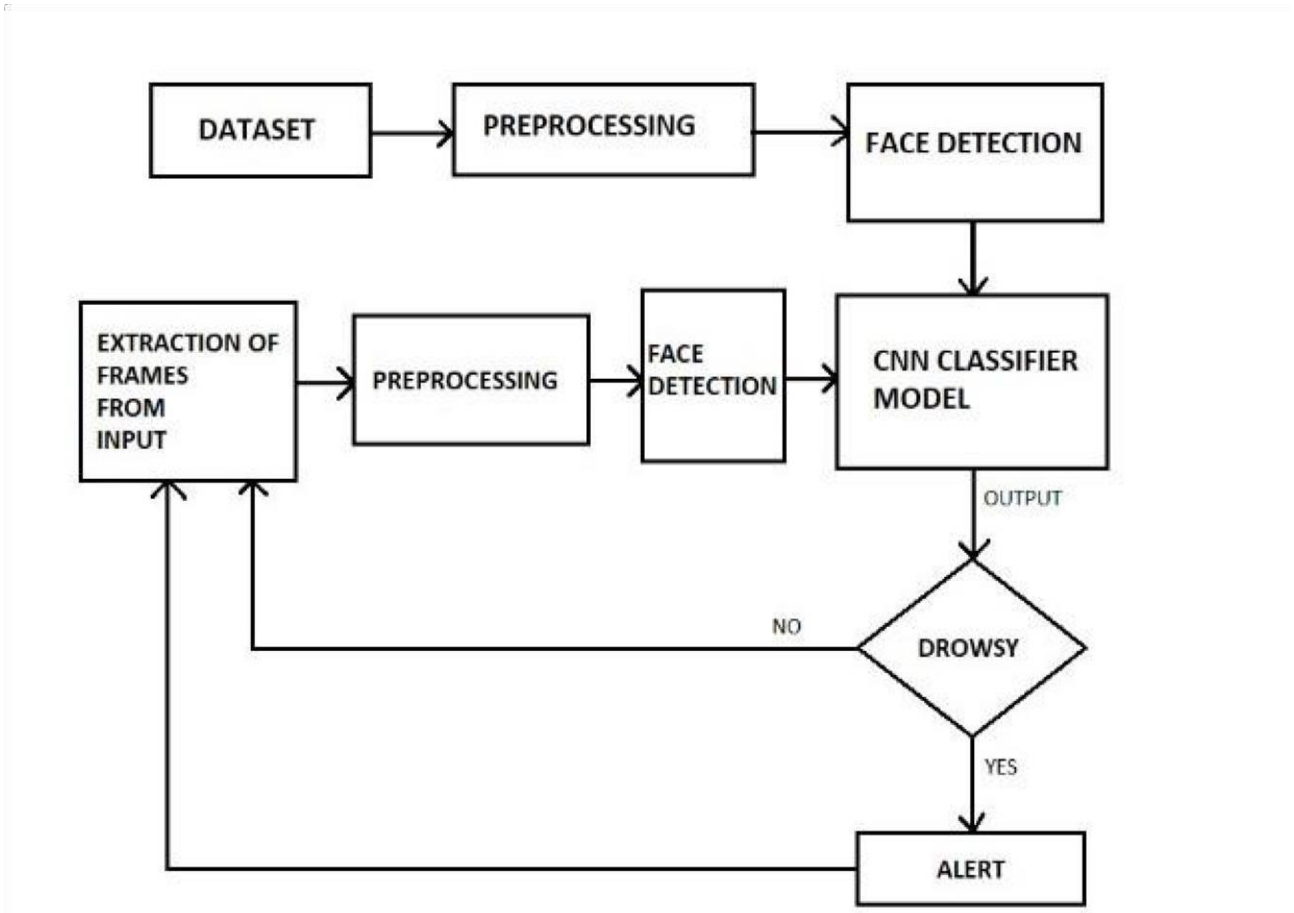
The accuracy of driver fatigue calculated as :

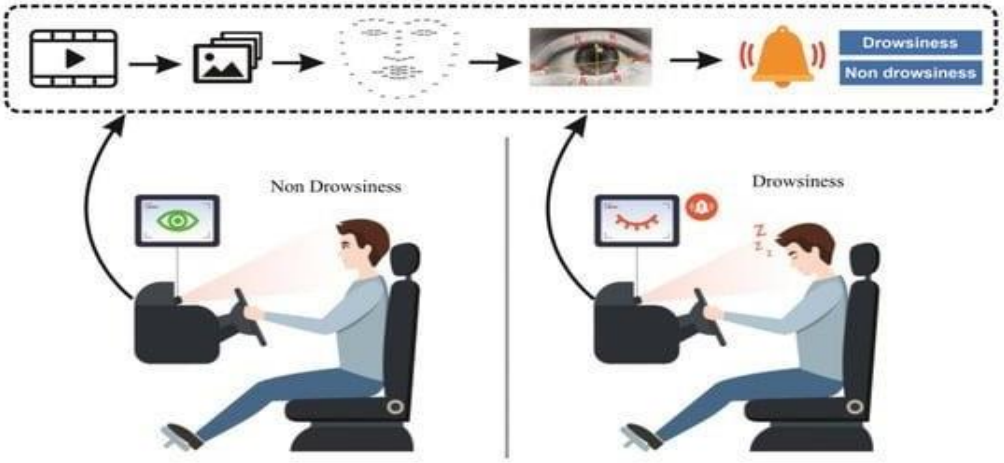
Accuracy= (TP+TN)/(TP+TN+FP+FN)

As a result of the study, a system was designed to investigate the effects of fatigue and insomnia on drivers physical transient behaviors, and to simulate drowsy drivers as a result of research. In this system, behavioral detection model is used for detecting fatigue drivers. The proposed system has five stages: PERCLOS, countof yawn, internal zone of the mouth opening, count of eye blinking and headdetection to extract attributes from video recordings. As a result, the system is classified as SVM in two classes (not fatigue / fatigue). In this study a Real Time Driver Fatigue Detection SVM Based on SVM Algorithm is presented whose test] results showed that the accuracy rate of fatigue detection is up to 97.93%

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN





3.1 DIFFERERNCE BETWEEN NON-DROWSINESS AND DROWSINESS

CHAPTER 4

METHODOLOGY

4.1. Existing System

The existing system used Support Vector Machine (SVM) for classifying the face (if the eyes ae closed or not) as drowsy or not drowsy. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. The objective of SVM is to find a plane that has the maximum margin, i.e., the maximum distance between data points of both classes.

4..2. Proposed System

In this system, instead of Support Vector Machine (SVM) we use a Classification Model based on Convolutional Neural Networks (CNN). Deep Learning is concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. ConvolutionalNeural Networks are a type of Artificial Neural Networks which are widely used for image classification and even Multi-Class classification of images. Convolution layers in a CNN consist of a set of learnable filters. During forward propagation, we slide each filter across the whole input step by step where each step is called stride. We propose CNN since the accuracy of the system is improved by using a CNN. The driver's face is continuously captured using a camera. The image frames are extracted and by face detection, the face of the driver is detected. Aclassification model is built based on CNN to classify the face as drowsy
/ not drowsy.

4.3. Proposed Techniques

4.3.1. Artificial Intelligence (AI)

Definition: Artificial Intelligence (AI) is the study and creation of computer systems that can perceive, reason and act. The primary aim of Al is toproduce intelligent machines. The intelligence should be exhibited by thinking, making decisions, solving problems, more importantly by learning. Al is an interdisciplinary field that requires knowledge in computer science,linguistics, psychology, biology, philosophy and so on for serious research.2According to the father of Artificial Intelligence, John McCarthy, it is the science and engineering of making intelligent machines, especially intelligent computer programs. It is a way of Making a Computer, a Computer- Controlled Robot, or a Software Think Intelligently in the similar manner the intelligent humans think. Artificial intelligence (AI), the ability of a digital

computer or computer controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from experience. Since the development of the digital computer in the 1940s, it has been demonstrated that computers can be programmed to carry out very complex tasks—as, for example, discovering proofs for mathematical theorems or playing chess— with great proficiency. Still, despite continuing advances in computer processing speed and memory capacity, there are as yet no programs that can match human flexibility over wider domains or in tasks requiring much everyday knowledge. On the otherhand, some programs have attained the performance levels of human experts and professionals in performing certain specific tasks, so that artificial intelligence in this limited sense is found in applications as diverse as medical diagnosis, computer search engines, and voice or handwriting recognition.

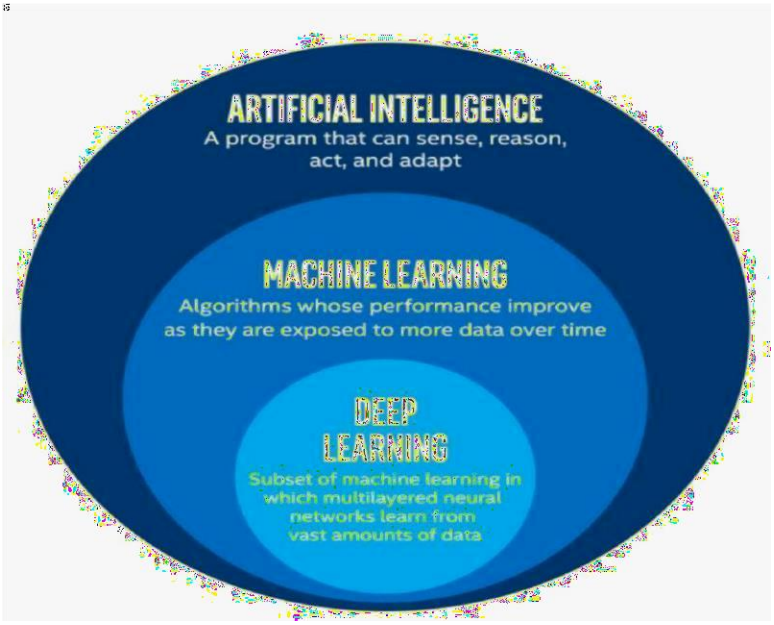


Fig. 4.1Artificial Intelligence

4.3.2. Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

Machine Learning is defined as the study of computer programs that leverage algorithms and statistical models to learn through inference and patterns without being explicitly programed. Machine Learning field has undergone significant developments in the last decade. Machine learning focuses on the development of computer programs that can access data anduse it to learn for themselves. There are also some types of machine learningalgorithms that are used in very specific use-cases, but three main methodsused today are:

- Supervised Machine Learning Algorithm
- Unsupervised Machine Learning Algorithm
- Reinforcement Machine Learning Algorithm

Among these, the type of machine learning algorithm we used in our system is supervised machine learning algorithm.

1. Supervised machine learning

This can apply what has been learned in the past to new data using labelled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is ableto provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the modelaccordingly.

2. Unsupervised machine learning

It holds the advantage of being able to work with unlabeled data. This means that human labor is not required to make the dataset machine- readable, allowing much larger datasets to be worked on by the program. Insupervised learning, the labels allow the algorithm to find the exact nature ofthe relationship between any two data points. However, unsupervised learning does not have labels to work off of, resulting in the creation of hidden structures. Relationships between data points are perceived by the algorithm in an abstract manner, with no input required from human beings.The creation of these hidden structures is what makes unsupervised learning algorithms versatile. Instead of a defined and set problem statement, unsupervised learning algorithms can adapt to the data by dynamically changing hidden structures. This offers more post deployment developmentthan supervised learning algorithms.

3. Reinforcement Machine Learning

It directly takes inspiration from how human beings learn from data intheir lives. It features an algorithm that improves upon itself and learns from new situations using a trial and-error method. Favorable outputs are encouraged or ‘reinforced’, and no favorable outputs are discouraged or ‘punished’. Based on the psychological concept of conditioning, reinforcement learning works by putting the algorithm in a work environment

with an interpreter and a reward system. In every iteration of the algorithm, the output result is given to the interpreter, which decides whether the outcome is favorable or not. In case of the program finding the correct solution, the interpreter reinforces the solution by providing a reward to the algorithm. If the outcome is not favorable, the algorithm is forced to reiterate until it finds a better result. In most cases, the reward system is directly tied to the effectiveness of the result.

4.3.3. Deep Learning

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

It is a form of machine learning, with functions that operate in a nonlinear decision-making process. Deep learning occurs when decisions are made on unstructured data without supervision. Object recognition, speech recognition, and language translation are some of the tasks performed through deep learning.

Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions. Deep learning AI is able to learn without human supervision, drawing from data that is both unstructured and unlabeled. It is a form of machine learning, can be used to help detect fraud or money laundering, among other functions.

4.3.4. Neural Networks

Neural networks are artificial systems that were inspired by biological neural networks. These systems learn to perform tasks by being exposed to various datasets and examples without any task-specific rules. The idea is that the system generates identifying characteristics from the data they have been passed without being programmed with a preprogrammed understanding of these datasets. Components of a typical neural network involve neurons, connections, weights, biases, propagation function, and a learning rule. Neurons will receive an input from predecessor neurons that have an activation, threshold, an activation function f, and an output function. Connections consist of connections, weights and biases which rules how neuron i transfers output to neuron j. Propagation computes the input and outputs the output and sums the predecessor neurons function with the weight. The learning rule modifies the weights and thresholds of the variables in the network. Artificial Neural Networks contain artificial neurons which are called **units**. These units are

arranged in a series of layers that together constitute the whole Artificial Neural Networks in a system. A layer can have only a dozen units or millions of units as this depends on the complexity of the system. Commonly, Artificial Neural Network has an input layer, output layer as well as hidden layers. The input layer receives data from the outside world which the neural network needs to analyze or learn about. Then this data passes through one or multiple hidden layers that transform the input into data that is valuable for the output layer. Finally, the output layer provides an output in the form of a response of the Artificial Neural Networks to input data provided.

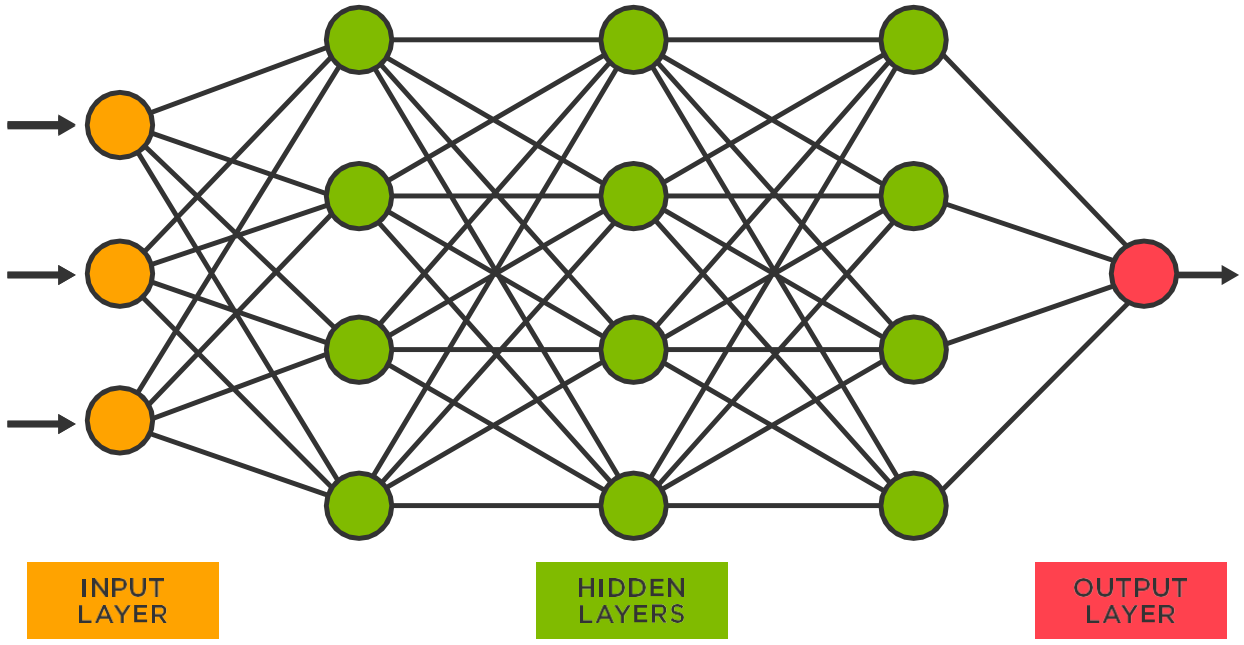


Fig.4.2Neural NetworksTypes:

1. Feedforward Neural Network

The feedforward neural network is one of the most basic artificial neural networks. In this ANN, the data or the input provided ravels in a single direction. It enters into the ANN through the input layer and exits through theoutput layer while hidden layers may or may not exist. So the feedforward neural network has a front propagated wave only and usually does not have backpropagation.

2. Recurrent Neural Network

The Recurrent Neural Network saves the output of a layer and feeds this output back to the input to better predict the outcome of the layer. The firstlayer in the RNN is quite similar to the feed-forward neural network and therecurrent neural network starts once the output of the first layer is computed. After this layer, each unit will remember some information fromthe previous step so that it can act as a memory cell in performing computations.

3. Convolutional Neural Network

A Convolutional neural network has some similarities to the feed-forward neural network, where the connections between units have weights that determine the influence of one unit on another unit. But a CNN has one or more than one convolutional layers that use a convolution operation on theinput and then pass the result obtained in the form of output to the next layer. CNN has applications in speech and image processing which is particularly useful in computer vision.

4. Modular Neural Network

A Modular Neural Network contains a collection of different neural networksthat work independently towards obtaining the output with no interaction between them. Each of the different neural networks performs a different sub-task by obtaining unique inputs compared to other networks. The advantage of this modular neural network is that it breaks down a large andcomplex computational process into smaller components, thus decreasing its complexity while still obtaining the required output.

5. Radial basis function Neural Network

Radial basis functions are those functions that consider the distance of a point concerning the center. RBF functions have two layers. In the first layer,the input is mapped into all the Radial basis functions in the hidden layer and then the output layer computes the output in the next step. Radial basisfunction nets are normally used to model the data that represents anyunderlying trend or function.

4.3.5. Python

Python is an interpreted, object-oriented, high-level programming languagewith dynamic semantics. It’s 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.

Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints,

stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edittest-debug
cycle makes this simple approach very effective. It's easy to learn syntax and portability capability makes it popular these days.

The followings facts give us the introduction to Python:

1. Python was developed by Guido van Rossum at Stichting Mathematisch Centrum in the Netherlands.
2. It was written as the successor of programming language named 'ABC'.
3. It's first version was released in 1991.
4. The name Python was picked by Guido van Rossum from a TV show named Monty Python's Flying Circus.
5. It is an open source programming language which means that we can freely download it and use it to develop programs. It can be downloaded from www.python.org.
6. Python programming language is having the features of Java and C both. It is having the elegant 'C' code and on the other hand, it is having classes and objects like Java for objectoriented programming.
7. It is an interpreted language, which means the source code of Python program would be first converted into bytecode and then executed by Python virtual machine.

Python for Data Science

Whether you're doing straightforward data analysis or full-on data science, you'd be hardpressed to find a better suite of tools than those in Python. The Pandas library is a quantum-leap improvement over the dusty Excel spreadsheets in which financial analysis was done for so long. If Pandas isn't fast enough for you, most of the basic vector operations can be one with NumPy. Numpy also offers the ability to do linear algebra, scientific computing, and a host of other highly technical things. It is, therefore, a great tool to learn how to use well. Python is the fifth most important language as well as most popular language for Machine learning and data science. The following are the features of Python that makes it the preferred choice of language for data science:

1. Extensive set of packages
2. Easy prototyping
3. Collaboration feature
4. One language for many domains

Artificial Intelligence and Machine Learning

Python community has developed many modules to help programmers implement machine learning. Artificial intelligence and machine learning have become buzzwords these days, but the truth is that it all comes downto algorithms, code, and logic. Given the scope and power of Python, it's no surprise that some truly world-class tools exist for generating intelligent behavior in Python. Arguably the most popular is the ubiquitous machine learning library, Scikit-Learn. Speaking from experience, Sklearn makes the process of building everything from classifiers to regression models orders of magnitude simpler than it
otherwise would be.If neural networks are more your jam, there's always TensorFlow. Adding in the new Keras API, building a state-of-the-art neural network is easier than it has ever been.

4.3.6. Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizationsand narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

The Jupyter notebook has two components. Users input programming code or text in rectangular cells in a front-end web page. The browser then passesthat code to a back-end 'kernel', which runs the code and returns the results(see our example at go.nature.com/2yqq7ak). By Pérez's count, more than 100 Jupyter kernels have been created, supporting dozens of programming languages. Normally, each notebook can run only one kernel and one language, but workarounds exist.

Importantly, the kernels need not reside on the user's computer. When future users of the LSST use Jupyter notebooks to analyse their data, the code will be running on a supercomputer in Illinois, providing computational muscle no desktop PC could match.
Notebooks can also run in the cloud. Google's Colaboratory project, forinstance, provides a Google-themed front-end to the Jupyter notebook. It enables users to collaborate and run code that exploits Google's cloud resources — such as graphical processing units — and to save their documents on Google Drive.

Two additional tools have enhanced Jupyter's usability. One is JupyterHub, a service that allows institutions to provide Jupyter notebooks to large pools of users. The IT team at the University of California, Berkeley, where Pérez is

a faculty member, has deployed one such hub, which Pérez uses to ensure that all students on his data-science course have identical computing environments. “We cannot possibly manage IT support for 800 students, helping them debug why the installation on their laptop is not working; that’s simply infeasible,” he says.

The other development is Binder, an open-source service that allows users to use Jupyter notebooks on GitHub in a web browser without having to install the software or any programming libraries. Users can also execute Jupyter notebooks on the Google cloud by inserting https://colab.research.google.com/github before the URL of a notebook on GitHub, or using the commercial service Code Ocean. In September, Code Ocean rolled out a new user interface for its cloud-based code-sharing and code-execution service, also based on Jupyter.

3.3.7. Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The preprocessing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

The term “Convolution” in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other. In simple terms, two images which can be represented as matrices are multiplied to give an output that is used to extract features from the image.

There are two main parts to CNN architecture:

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function which are defined below.

❖ *. Training*

Deep learning neural networks learn a mapping function from inputs to outputs. This is achieved by updating the weights of the network in response to the errors the model makes on the training dataset. Updates are made to continually reduce this error until either a good enough model is found or the learning process gets stuck and stops.

The stochastic gradient descent algorithm is used to solve the optimization problem where model parameters are updated each iteration using the backpropagation algorithm.

A neural network model uses the examples to learn how to map specific sets of input variables to the output variable. It must do this in such a way that this mapping works well for the training dataset, but also works well on new examples not seen by the model during training. This ability to work well on specific examples and new examples is called the ability of the model to generalize.

- **Loss Function**: The function used to estimate the performance of a model with a specific set of weights on examples from the training dataset.
- **Weight Initialization**: The procedure by which the initial small random values are assigned to model weights at the beginning of the training process.
- **Batch Size**: The number of examples used to estimate the error gradient before updating the model parameters.
- **Learning Rate**: The amount that each model parameter is updated per cycle of the learning algorithm.
- **Epochs**: The number of complete passes through the training dataset before the training process is terminated.

Train Dataset:

The sample of data used to fit the model. The part of data we use to train our model. This is the data which your model actually sees (both input and output) andlearn from

- *. Testing*

The usage of the word "testing " in relation to Machine Learning models is primarily used for testing the model performance in terms of accuracy/precision of the model. It can be noted that the word, "testing" means different for conventional software development and Machine Learning model development.

The goal of ML testing:

Quality assurance is required to make sure that the software system works according to the requirements. Were all the features implemented as agreed? Does the program behave as expected? All the parameters that you test the program against should be stated in the technical specification document.

Moreover, testing has the power to point out all the defects and flaws during development. We don't want your clients to encounter bugs after it is released and come to you waving their fists. Different kinds of testing allowus to catch bugs that are visible only during runtime.

However, in machine learning, a programmer usually inputs the data and the desired behavior, and the logic is elaborated by the machine. This is especially true for deep learning. Therefore, the purpose of machine learning testing is, first of all, to ensure that this learned logic will remain consistent,no matter how many times we call the program.

First of all, we split the database into three non-overlapping sets. You use a trainingset to train the model. Then, to evaluate the performance of the model, you use two sets of data:

Validation Dataset:

Having only a training set and a testing set is not enough if you do many rounds of hyper parameter-tuning (which is always). And that can result in overfitting. To avoid that, you can select a small validation data set to evaluate a model. Only after you get maximum accuracy on the validationset, you make the testing set come into the game.

Test Dataset:

Our model might fit the training dataset perfectly well. In order to assure

that it will do equally well in real-life, you select samples for a testing set from your training set —
examples that the machine hasn't seen before. It is important to remain unbiased during selection and draw samples at random. Also, you should not use the same set many times to avoid training on your test data. Your testset should be large enough to provide statistically meaningful results and be representative of the data set as a whole.

LIBRARIES USED

- **OpenCV (Open Source Computer Vision Library)**

OpenCV is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.
It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDA and OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlesslywith STL containers.

- **Numpy**

NumPy stands for Numerical Python. NumPy was created in 2005 by Travis Oliphant. NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, randomsimulation and much more.
At the core of the NumPy package, is the *ndarray* object. It provides a lot of supporting functions that make working with ndarray very easy. Arrays arevery frequently used in data science, where speed and resources are very important. NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behavior is called locality of reference in computer science. This is the main reason why NumPy is faster than lists. Also it is optimized to work withlatest CPU architectures.
NumPy is a Python library and is written partially in Python, but most of the

parts that require fast computation are written in C or C++.

- **Tensorflow**

TensorFlow is an open-source software library. **TensorFlow** was originally developed by
researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well. Google open-sourced TensorFlow in November 2015.

TensorFlow's popularity is due to many things, but primarily because of the computational graph concept, automatic differentiation, and the adaptabilityof the Tensorflow python API structure. This makes solving real problems with TensorFlow accessible to most programmers. Google's Tensorflow engine has a unique way of solving problems. This unique way allows for solving machine learning problems very efficiently.

What is Tensor in Tensorflow

TensorFlow, as the name indicates, is a framework to define and run computations involving tensors. A tensor is a generalization of vectors and matrices to potentially higher dimensions. Internally, TensorFlow represents tensors as n-dimensional arrays of base datatypes. Each element in the Tensor has the same data type, and the data type is always known. The shape(that is, the number of dimensions it has and the size of each dimension) might be only partially known. Most operations produce tensors of fullyknown shapes if the shapes of their inputs are also fully known, but in some cases it's only possible to find the shape of a tensor at graph executiontime.

- **Keras**

Keras is a python based open-source library used in deep learning (for neural networks).

Keras was released in March 2015. It can run on top of TensorFlow, Microsoft CNTK or Theano. It is very simple to understand and use, and suitable for fast experimentation. It is designed to be fast and easy for the user to use. Keras models can be run both on CPU as well as GPU.

Keras is the best platform out there to work on neural network models. The API that Keras has a user-friendly where a beginner can easily understand. Keras has the advantage that it can choose any libraries which support it forits backend support. Keras provides various pre- trained models which helpthe user in further improving the models the user is designing.

- **Matplotlib**

Matplotlib is a low level graph plotting library in python that serves as a visualization utility.
Matplotlib was created by John D. Hunter. Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi- platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

- **Seaborn**

Seaborn is a library for making statistical graphics in Python. It builds on topof matplotlib and integrates closely with pandas data structures. It providesa high-level interface for drawing attractive and informative statistical graphics.
Seaborn helps you explore and understand your data. Its plotting functions operate on dataframes and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots. Its datasetoriented, declarative API lets you focuson what the different elements of your plots mean, rather than on the detailsof how to draw them.

- ***Seaborn Heatmap***

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common valuesor higher activities brighter colors basically reddish colors are used and to represent less common or activity values, darker colors are preferred. Heatmap is also defined by the name of the shading matrix. Heatmaps in Seaborn can be plotted by using the seaborn.heatmap() function.

- **Sklearn**

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.
It was originally called ***scikits.learn*** and was initially developed by David

Cournapeau.

Sklearn Metrics

Sklearn provides metrics for evaluating the performance of the model.

Classification Metrics

The **sklearn.metrics** module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values. Most implementations allow each sample to provide a weighted contribution to the overall score, through the sample weight parameter. The confusion matrix function evaluates classification accuracyby computing the confusion matrix with each row corresponding to the trueclass (Wikipedia and other references may use different convention for axes). By definition, entry i,j in a confusion matrix is the number of observations actually in group i, but predicted to be in group j. Here is an example: Intuitively, precision is the ability of the classifier not to label as positive a sample that is negative, and recall is the ability of the classifier to find all thepositive samples.

- **OS module in Python**

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you wantto manipulate paths, see the os.path module, and if you want to read all thelines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module. OS comes under Python's standard utility modules. The *os* and *os.path* modules include many functions to interact with the file system.

- **os.listdir()**

os.listdir() method in Python is used to get the list of all files and directories in the specified directory. If we don't specify any directory, then list of files and directories in the current working directory will be returned.

- **os.path module**

This module contains some useful functions on pathnames. The path parameters are either *strings or bytes*. These functions here are used for

different purposes such as for merging, normalizing and retrieving path names in python. All of these functions accept either only bytes or only stringobjects as their parameters. The result is an object of the same type, if a pathor file name is returned. As there are different versions of operating systemso there are several versions of this module in the standard library. ***os.path.join()*** method in Python join one or more path components intelligently. This method concatenates various path components with exactly one directory separator (‘/’) following each non-empty part except the last path component. If the last path component to be joined is empty thena directory separator (‘/’) is put at the end. If a path component represents an absolute path, then all previous components joined are discarded and joining continues from the absolute path component.

- **Playsound module in Python**

The playsound module is a cross platform module that can play audio files. This doesn’t have any dependencies, simply install with pip in your virtual environment and run. Implementation is different on platforms. It uses windll.winm on Windows , AppKit. NSSound on Apple OS X and GStreamer on Linux. The *playsound* module contains a function named **playsound()**. It works with both **WAV** and **MP3** files

CHAPTER 5

CODING AND TESTING

5.1. Importing all the required libraries

```
import tensorflow
as tfimport cv2
import os
import matplotlib.pyplot as plt
import numpy as np
```

We create a dictionary which contains labels for 2 sets of images.

```
data_path='D/'
categories=os.listdir(data_p
ath) labels=[i for i
in
range(len(categories))]
label_dict=dict(zip(categories,labe
ls)) print(label_dict)
print(categ
ories)print(labels)
```

```
{'alert': 0, 'drowsy': 1}
['alert',
'drowsy'][0,
1]
```

5.2. Code for Face Detection

```
img_size=100
data=[] target=[] fpath='D/' face_cascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml') for
categoryin
categories:
folder_path=os.path.join(data_path,category)
img_names=os.listdir(folder_path)
for img_name
inimg_names:
img_path=os.path.join(folder_path,img_name)
img=cv2.imread(img_path) try:
        gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
         #Coverting the image into gray scale faces =
```

```
face_cascade.detectMultiScale(gray, 1.1, 4)  for (x, y, w, h) in
faces:cv2.rectangle(img, (x, y), (x+w, y+h),
(255,  0,  0),  2)  roi  =  img[y:y  +  h,  x:x
  +w]resized=cv2.resize(roi,(img_size,img_size))
 #resizing the region of interest into 100x100, since we need a fixed
common size for all the images in the dataset
face_path=fpath+'/'+category+'/'+'image_'+img_name+'.jpg'
cv2.imwrite(face_path,resized)
 #Saving the resized images with detected
 facesexcept
 Exception as e: print('Exception:',e)
```

5.3. Code for reading images after Face Detection

```
data=[]

target=[
]        a=0d=0

for category in categories:  #Path  to  access  each
                             directory
folder_path=os.path.join(data_path,category)
   img_names=os.listdir(folder_path)

  for img_name in img_names: data.append(img)
    #Path  to  access each  image  in  the  particular
          directoryimg_path=os.path.join(folder_path,img_name)
#Read  the  image
img=cv2.imread(img_path)if(category=="alert"
): a+=1
 elif(category=="drowsy
 "):d+=1
 try:

    #Appending the image to data list  and its label to
                                       target  list
    target.append(label_dict[category])

  except  Exception  as  e:print('Exception:',e)
```

code for converting images intonumpy

```
data=np.array(data)/255
.0 newtarget=[] for
i intarget: if
            i=
=0:
newtarget.append([1,0])  elifi==1:
```

```
newtarget.append([0,1])
np.save('data',data)        np.save('target',newtarget)
                            data=np.load('data.npy')target=np.load('target.npy')
```

5.4.Code for training the CNN

```
model=Sequential() input_shape
= (100, 100, 3)

#The first Convolutional Layer (with ReLU) and a Pooling
Layer #It contains 200 filters of 3x3 size

model.add(Conv2D(200,(3,3),input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))#The second Convolutional
Layer (with ReLU) and a Pooling
Layer #It contains 100 filters of 3x3 size


model.add(Conv2D(100,(3,3)))
                                        model.add(Activation('r
elu'))model.add(MaxPooling2D(pool_size=(2,2)))

#The third Convolutional Layer (with ReLU) and a Pooling
Layer #It contains 100 filters of 3x3 size

model.add(Conv2D(100,(3,3)))
                                        model.add(Activation('r
elu'))model.add(MaxPooling2D(pool_size=(2,2)))

#The flatten layer

model.add(Flatten())
model.add(Dropout(0.5))#The dense layer
with 64 neurons

model.add(Dense(64,activation='relu'))

#The      output      layer      with      2      neurons      for      each
          class model.add(Dense(2,activation='sigmoid'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['a
ccuracy','Prec ision','Recall'])
```

model.summary()

Model: "sequential"

Layer (type) Output Shape Param #

==========

conv2d (Conv2D) (None, 98, 98, 200) 5600

activation (Activation) (None, 98, 98, 200) 0

max_pooling2d

(MaxPooling2D) (None, 49, 49, 200) 0
conv2d_1 (Conv2D) (None, 47, 47, 100) 180100
activation_1 (Activation) (None, 47, 47, 100) 0
max_pooling2d_1 (MaxPooling2 (None, 23, 23, 100) 0
conv2d_2 (Conv2D) (None, 21, 21, 100) 90100
activation_2 (Activation) (None, 21, 21, 100) 0
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 100) 0
flatten (Flatten) (None, 10000) 0
dropout (Dropout) (None, 10000) 0
dense_1 (Dense) (None, 64) 640064
dense_2 (Dense) (None, 2) 130

Total params: 915,994
Trainable params: 915,994
Non-trainable params: 0

Splitting data into train data and test data:

```
from sklearn.model_selection import train_test_split

train_data,test_data,train_target,test_target=train_test_split(data,target,test
_size=0.2,shuffl e=1)
```

Training:

```
checkpoint = ModelCheckpoint('model-
best.model',monitor='val_loss',verbose=1,save_best_only=True,mode='auto')
history=model.fit(train_data,train_target,epochs=20
,callbacks=[checkpoint],validation_sp  lit=0.1,verbose=1)
```

5.6. Code for predicting the images captured from the Camera

```
#importing the playsound module to play audio fromplaysound
import playsound

#Creating an object for capturing video vid =
cv2.VideoCapture(0)   img_size=100          i=0
data1=[] color=(225,225,225) while(i<=10):

    # Capture the video frame#
  by frame  ret, frame
= vid.read()

cv2.imshow('frame', frame) if
cv2.waitKey(1) & 0xFF == ord('q'):
break

    face_cascade                                                    =
    cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    #Coverting the image into gray scale faces =
face_cascade.detectMultiScale(gray, 1.1, 4)

 for (x, y, w, h) in faces: cv2.rectangle(frame, (x, y),(x+w,
y+h), color, 2)
  cv2.imshow("Frame",     frame)
roi       =       frame[y:y       +       h,       x:x       +       w]
    resized=cv2.resize(roi,(img_size,img_size))
    #resizing the gray scale into 100x100, since we need a fixed common size for allthe
images in the dataset

    data1.append(resized)
    i+=1

    if(i==5):
       data1=np.array(data1)/255.0
#data1=np.reshape(data1,(data1.shape[0],img_size,img_size,1))   input_shape
 = (100, 100, 3)
       #data=np.reshape(data,(img_size,img_size,1))
 prediction = (model.predict(data1))
       #print(prediction)          drowsycount=0
       new_prediction=np.argmax(prediction,axis=1)
```

```
        for  i       in      new_prediction: if(i==1):
                drowsycount+=1
  color=(0,0,225)          else:
  color=(0,225,0)
if(drowsycount==5):        print("Drowsy")
  print('playing              sound')
  playsound('alertsound1.mp3
  ') else:     print("Not  drowsy")
        cv2.rectangle(frame, (x, y), (x+w, y+h), color, 2)
        cv2.imshow("Frame", frame)
       data1=[]      i=0
  # After the loopvid.release()
  # Destroy all the   windows
  cv2.destroyAllWindows()
```

CHAPTER 6

SCREENSHOTS AND RESULTS

Input:

The input is an image which contains a human face. The following are some ofthe images from our dataset.

The following are the image after performing face detection and resizing

Fig. 6.1. Alert faces

Fig. 6.2. Drowsy faces

Output:

If the driver is drowsy, an alert (voice) is generated as the output.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

10.1. Conclusion

The paper described a drowsiness detection system based on CNN-based Machine Learning. We used OpenCV to detect faces and eyes using a haar cascade classifier and then we used a CNN model to predict the status. Finally, driver fatigue is evaluated. The experimental results demonstrate that when the drowsy counts reaches to 5, the driver can be considered in a fatigue state.The system was able to detect facial landmarks from images and pass it to a CNN-based trained Deep Learning model to detect drowsy driving behavior.

There are limitations to this technology, such as obstructing the view of facial features by wearing sunglasses and bad lighting conditions. However, given the current state, there is still room for performance improvement and better facial feature detection even in bad lighting conditions. Thus we have successfully designed a prototype drowsiness detection system using OpenCV software and Haar Classifiers. The system so developed was successfully tested, its limitations identified and a future plan of action developed.

The ultimate goal is to create a safer driving experience by significantly reducing the number of accidents caused by drowsy driving

10.2. Future Scope

The work can be extended by combing physiological approach techniques, so the driver can be detected as drowsy through those readings. In special situations like when a driver wares spectacles along with mask then we can use the physiological reading of the driver. We can detect the drowsiness of the driver more accurately with these readings. We can use the devices which are mentioned earlier and detect the drowsiness of the driver. While this is a research project, there is scope when this completely turns out to be developed into an application which can be run by the end users on their own for their own purposes on their own systems.

REFERENCES

[1] Rosebrock, A. (2017, May 8). Retrieved from Pyimagesearch: https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/

[2] Awais M, Badruddin N, Drieberg M. A Hybrid Approach to Detect Driver Drowsiness Utilizing Physiological Signals to Improve System Performance and Wearability. Sensors (Basel). 2017 Aug 31;17(9):1991. doi: 10.3390/s17091991. PMID: 28858220; PMCID: PMC5620623. https://pubmed.ncbi.nlm.nih.gov/28858220/

[3] Arefnezhad, S., Samiee, S., Eichberger, A., & Nahvi, A. (2019). Driver Drowsiness Detection Based on Steering Wheel Data Applying Adaptive Neuro-Fuzzy Feature Selection. *Sensors (Basel, Switzerland)*, *19*(4), 943. https://doi.org/10.3390/s19040943

[4] B. K. Savaş and Y. Becerikli, "Real Time Driver Fatigue Detection Based on SVM Algorithm," 2018 6th International Conference on Control Engineering & Information Technology (CEIT), 2018, pp. 1-4, doi: 10.1109/CEIT.2018.8751886. https://ieeexplore.ieee.org/abstract/document/8751886/

[5] Vesselenyi, T., Moca, S., Rus, A., Mitran, T., & Tătaru, B. (2017, October). Driver drowsiness detection using ANN image processing. In *IOP Conference Series: Materials Science and Engineering* (Vol. 252, No. 1, p. 012097). IOP Publishing. https://iopscience.iop.org/article/10.1088/1757-899X/252/1/012097/meta

[6] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[7] https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

[8] Chirra, V. R. R., ReddyUyyala, S., & Kolli, V. K. K. (2019). Deep CNN: A Machine Learning Approach for Driver Drowsiness Detection Based on Eye State. *Revue d'Intelligence Artificielle*, *33*(6), 461-466.

[9] L. B. Leng, L. B. Giin and W. Chung, "Wearable driver drowsiness detection system based on biomedical and motion sensors," 2015 IEEE SENSORS, 2015, pp. 1-4, doi: 10.1109/ICSENS.2015.7370355.

[10] asim AL-Anizy, G. J., Nordin, M. J., & Razooq, M. M. (2015). Automatic driver drowsiness detection using haar algorithm and support vector machine techniques. *Asian Journal of Applied Sciences*, *8*(2), 149-157.

[11] https://www.geeksforgeeks.org/metrics-for-machinelearningmodel/#:~:text=It%20is%20one%20of%20the,higher%20than%20a%2 0negati ve%20e xample. 41