

**DRIVER DROWSINESS-DETECTION SYSTEM BASED
ON TRANSFER LEARNING AND MC-KCF**

*A Project report submitted in partial fulfillment of the requirements for
the award of the degree of*

**BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMMUNICATION ENGINEERING**

Submitted by

H. Dinesh (319126512085)

A. Pavan Kalyan (319126512067)

V. Shanmukha Raju (319126512126)

Ch. Kumar Charukesh (319126512076)

Under the guidance of

Mrs V. Shireesha

Assistant Professor



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)**

(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC)

Sangivalasa, Bheemili mandal, Visakhapatnam dist. (A.P)

2022-2023


DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES
(UGC AUTONOMOUS)
(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC)
Sangivalasa, Bheemili mandal, Visakhapatnam dist. (A.P)



CERTIFICATE

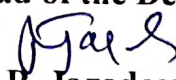
This is to certify that the project report entitled “DRIVER DROWSINESS-DETECTION SYSTEM BASED ON TRANSFER LEARNING AND MC-KCF” submitted by Hanumanthu Dinesh (319126512085), Ayyalasomayajula Pavan Kalyan (319126512067), Vysyaraju Shanmukha Raju (319126512126), Chavala Kumar Charukesh (319126512076) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering of Anil Neerukonda Institute of technology and Sciences(A), Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.

Project Guide


V. Shireesha
Assistant Professor
Department of E.C.E
ANITS

Assistant Professor
Department of E.C.E.
Anil Neerukonda
Institute of Technology & Sciences
Sangivalasa, Visakhapatnam-531 162

Head of the Department


Dr. B. Jagadeesh
Professor & HOD
Department of E.C.E
ANITS

Head of the Department
Department of E C E

Anil Neerukonda Institute of Technology & Sciences
Sangivalasa - 531 162

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **V. Shireesha**, Assistant professor, Department of Electronics and Communication Engineering, ANITS, for her guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Dr. B. Jagadeesh**, Head of the Department, Electronics and Communication Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, ANITS, Sangivalasa**, for their encouragement and cooperation to carry out this work.

We express our thanks to all **teaching faculty** of Department of ECE, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank **all non-teaching staff** of the Department of ECE, ANITS for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last, but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

PROJECT STUDENTS

H. Dinesh	(319126512085)
A. Pavan Kalyan	(319126512067)
V. Shanmukha Raju	(319126512126)
Ch. Kumar Charukesh	(319126512076)

ABSTRACT

The driver fatigue is an important issue in many vehicle accidents. According to the National Highway Safety Administration, drowsy driving causes more than 100,000 crashes, 71,000 injuries, and 1,550 death toll each year in India.

Earlier approach normally based on DriCare technique, detects the driver's fatigue status, such as yawning, blinking, and duration of eye closure, using video images, without equipping their bodies with devices. This approach uses KCF (Kernelized Correlation Filter) and CNN (Convolution neural network) algorithms.

The proposed work is based on the issues related to driver drowsiness detection and alert system. The proposed algorithm uses the features of deep convolutional neural network-like RESNET (Residual neural network) and MC-KCF (Multi Convolution neural network-KCF). These techniques are used to detect driver drowsiness by measuring yawning, head position and eye rotation.

Keywords: Deep Convolution Neural network, fatigue detection, face tracking, Drowsiness Detection, feature location, ResNet, MC-KCF.

CONTENTS

ACKNOWLEDGEMENT	ii
ABSTRACT	iii
LIST OF FIGURES	vii
CHAPTER 1: Introduction	1
1.1 Introduction	1
1.2 Problem Statement	1
1.3 Project Requirements	2
1.3.1 Software	2
1.3.2 Hardware	2
CHAPTER 2: LITERATURE SURVEY	3
2.1 Drowsiness Detection System Using Physiological Signals	3
2.2 Drowsiness Detection with OpenCV using EAR.	6
2.3 Driver Drowsiness detection using ANN image processing	8
CHAPTER 3: Methodology	10
3.1 Proposed Methodology	10
3.1.1 Existing System	10
3.1.2 Proposed System	10
3.2 Proposed Techniques:	11
3.2.1 Artificial Intelligence (A.I)	11
3.2.2 Machine Learning	13
3.2.3 Deep Learning	13
3.2.4 Neural Networks	14
3.2.5 Convolution Neural Network (CNN)	15
3.2.5.1 Convolutional Layer	16
3.2.5.2 Pooling Layer	18
3.2.5.2.1 Max Pooling	18
3.2.5.2.2 Average Pooling	18

3.2.5.2.3	Global Pooling	19
3.2.5.3	Fully Connected Layer	19
3.2.5.4	Dropout Layer	20
3.2.5.5	RESNET	20
3.2.5.5.1	Residual Blocks	20
3.2.5.5.2	Architecture Of RESNET	21
3.2.5.5.3	Using ResNet with Keras	23
3.2.6	Back Propagation	23
3.2.7	Activation Functions	24
3.2.8	Training	24
3.2.8.1	Test Loss	25
3.2.8.2	Test Accuracy	25
3.2.8.3	Validation Loss	26
3.2.8.4	Validation Accuracy	26
3.2.9	Train Dataset	26
3.2.10	Testing	27
CHAPTER 4:	Software Used	28
4.1	Python	28
4.2	Jupyter Notebook	28
4.3	Libraries	29
4.3.1	Open Source-Computer Vision Library	29
4.3.2	Numerical Python	30
4.3.3	Tensor Flow	31
4.3.4	Keras	31
4.3.5	Matplotlib	32
4.3.6	OS module in Python	33
4.3.7	Dlib	34
CHAPTER 5:	Experimental Results	35
5.1	Results through live face tracking	36
5.1.1	Frames recognized as drowsy	36
5.1.2	Frames recognised as not drowsy	37

CHAPTER 6: Conclusion and Future Scope	40
6.1 Conclusion	40
6.2 Future Scope	41
REFERENCES	42

LIST OF FIGURES

Figure 1 Venn Diagram of A.I	12
Figure 2 Layers in Neural Network	15
Figure 3 Operation of Convolution	17
Figure 4 Stide	17
Figure 5 Operation of Max Pooling	18
Figure 6 Operation of Average Pooling	19
Figure 7 Residual Blocks	21
Figure 8 Architecture of ResNet	22
Figure 9 Back Propogation	24
Figure 10 Driver Input images	35
Figure 11 Alert Images	35
Figure 12 Drowsy images	35
Figure 13 Person 1 in drowsy state	36
Figure 14 Person 2 in drowsy state	36
Figure 15 Person 3 in drowsy state	36
Figure 16 Person 4 in drowsy state	36
Figure 17 Person 5 in drowsy state	37
Figure 18 Person 1 in alert state	37
Figure 19 Person 2 in alert state	37
Figure 20 Person 3 in alert state	38
Figure 21 Person 4 in alert state	38
Figure 22 Person 5 in alert state	38

LIST OF ABBREVIATIONS

AI	:	Artificial Intelligence
BSD	:	Berkely Source Distribution
CNN	:	Convolution Neural Network
CUDA	:	Computer Unified Device Architecture
EAR	:	Eye Aspect Ratio
ECG	:	Electrocardiography
EEG	:	Electro Encephalogram
EOG	:	Electrooculography
FC	:	Fully-Connected
FFT	:	Fast Fourier Transform
GB	:	Giga Byte
GPU	:	Graphics processor unit
GUI	:	Graphic User Interface
HF	:	High Frequency
HRV	:	Heart Rate Variability
I/O	:	Input/Output
IDE	:	Integrated Development Environment
IT	:	Information Technology
KCF	:	Kernelized Correlation Filter
LF	:	Low Frequency
LSST	:	Large Synoptic Survey Telescope
MATLAB	:	Matrix Laboratory
MEMS	:	Micro-Electromechanical Systems
OS	:	Operating System
PPG	:	Plethysmo Graphy
PyPI	:	Python Package Index

RBF	:	Radial Basic Function
ReLU	:	Rectified Linera Activation unit
RESNET	:	Residual Neural Network
ROC	:	Receiver Operation Curve
SVM	:	Support Vector Machine
TV	:	Television
URL	:	Unifrom Resource Locator
USB	:	Universal Serial Bus
VGG	:	Visual Geometry Group
YawDD	:	Yawning Detection Dataset

CHAPTER 1

Introduction

1.1 Introduction

Around 1.3 million individuals pass away each year due to car accidents, which are primarily caused by driver distraction and drowsiness. Many individuals travel long distances on highways, which can lead to fatigue and stress. Drowsiness can arise unexpectedly, resulting from sleep disorders, medication, or for instance, boredom can arise while driving for long periods. Therefore, drowsiness can create hazardous situations and elevate the likelihood of accidents.

Given the circumstances, it is crucial to employ modern technologies to develop and construct systems capable of monitoring drivers and assessing their attentiveness throughout the entirety of their time on the road.

Project team has developed a solution to prevent such accidents. The system involves utilizing a camera to capture the user's visual features, with the use of face detection and CNN techniques to identify any signs of drowsiness in the driver. When drowsiness is detected, an alarm will sound to alert the driver, prompting them to take precautionary measures. The detection of driver drowsiness is instrumental in reducing the number of fatalities caused by traffic accidents.

1.2 Problem Statement

Road accidents caused by human errors are responsible for numerous fatalities and injuries worldwide. The primary reason behind such accidents is the driver's drowsiness, which could result from sleep deprivation or prolonged driving hours. To address this issue, it is imperative to develop a system that leverages the latest available technologies to minimize the likelihood of accidents. The main objective of this system

is to create a model that can issue an alert in case the driver shows signs of drowsiness. This alert will help the driver become aware of their condition and take the necessary measures to prevent an accident.

1.3 Project Requirements

1.3.1 Software

Software required are:

- Windows, Linux, MacOS used for operating system.
- Python 3.10(recent version) is used as language.
- Python IDE, Jupiter Notebook used as IDE's.

1.3.2 Hardware

Minimum Hardware required are:

- High computational processor
- Minimum 4 GB RAM
- Webcam which supports night vision
- Alarm

CHAPTER 2

LITERATURE SURVEY

2.1 Drowsiness Detection System Using Physiological Signals

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

Publication year: 2013

Drowsiness can be detected by measuring physiological parameters like heart rate, pulse rate, breathing rate, respiratory rate, and body temperature are considered more precise and dependable in identifying drowsiness since they exhibit measurable physiological changes directly related to the driver's physical condition. When a person becomes drowsy, their physiological parameters tend to alter, for instance, a drop in blood pressure, heart rate, and body temperature. Drowsiness detection systems that rely on these physiological indicators can identify such changes and caution the driver when they are at risk of falling asleep. However, since these systems necessitate electrodes to be attached to the driver's body, they are considered invasive. Here is a compilation of drowsiness detection systems that rely on physiological conditions.

2.1.1. EEG-BASED DRIVER FATIGUE DETECTION

A system has been suggested for identifying driver fatigue and exhaustion prevent Car accidents resulting from drivers who were sleepy or drowsy. This system uses Electroencephalogram (EEG) signals to determine the degree of sleepiness or drowsiness experienced by a driver. The system first identifies an index that corresponds to various levels of drowsiness. A cheap neuro signal acquisition device with a single electrode is used to obtain the EEG signal from the driver. A collection of data designed for simulated car drivers experiencing different levels of drowsiness was collected locally to evaluate the system. The findings indicated that the system proposed was successful in detecting fatigue in all the subjects.

2.1.2. PULSE SENSOR METHOD

Previous studies have primarily focused on using drivers' physical conditions to detect drowsiness. To address this issue, Rahim developed a system that uses infrared heart rate or pulse sensors to detect drowsy drivers. The pulse sensor gauges the heart's pulse rate by detecting the driver's finger or hand. By detecting the quantity of blood circulating through the finger, the sensor can determine the amount of oxygen in the blood, causing reflecting infrared light and transmit the data to the Arduino microcontroller. The variation in oxygen levels is then processed by HRV frequency domain software to visualize the driver's heart pulse rate. The results of the experiment showed that the LF/HF As drivers shift from being alerted to feeling drowsy, the ratio of oxygen tends to decline. By issuing timely warnings, numerous car accidents can be averted.

2.1.3. WEARABLE DRIVER DROWSINESS DETECTION SYSTEM

In the past, Applications designed for mobile devices have been created to identify and detect driver drowsiness. However, these applications can distract drivers and lead to accidents. To address this issue, Lenget developed a drowsiness detection system in the shape of a custom-designed wristband that can be worn has been developed, featuring a PPG signal and galvanic skin response sensor. The information gathered by these sensors is sent to a mobile device, which functions as the primary assessment unit. Motion sensors in the mobile device analyse the data, and five features (heart rate, breathing rate, level of stress, variability in pulse rate, and the count of adjustments made) are extracted for computation. These characteristics are subsequently employed as calculation parameters for an SVM classifier, which is utilized to assess the driver's level of drowsiness. The findings of the experiment indicated an accuracy up to 98.02% for the proposed system. In the event of drowsiness, the mobile device generates a warning system that makes use of both visual and vibrational alerts to notify the driver.

2.1.4. WIRELESS WEARABLES METHOD

Warwick has proposed a drowsiness detection system that utilizes a wearable biosensor known as Bio-harness to reduce the likelihood of road accidents. The system is comprised of two stages. During the first stage, the Bio-harness gathers the driver's physiological data, such as ECG, heart rate, and posture, among other metrics. This data is then analysed to determine key parameters linked to drowsiness. In the second stage, a drowsiness detection algorithm is established, and a mobile application is developed to warn drowsy drivers.

2.2 Drowsiness Detection with OpenCV using EAR.

Author: Adrian Rosebrock

Publication year: 2017

The paper proposes an algorithm that can detect eye blinks in real-time using video footage from a standard camera. The algorithm utilizes this paper proposes the use of landmark detectors that are trained on datasets containing images of people in everyday settings (in-the-wild datasets) for the purpose of detecting eye blinks in real-time from a video sequence captured by a standard camera, which are highly robust against different factors such as head orientation, varying illumination, and facial expressions. These detectors can precisely detect facial landmarks and the algorithm is designed to estimate the degree of eye opening, which is essential for detecting eye blinks accurately.

Several techniques have been proposed for automatic identification of eye blinks in video sequences, and some of these methods depend on analysing the motion within the eye region. Various techniques have been proposed to automatically identify eye blinks in video sequences, including methods that rely on analysing the motion in the eye region. Typically, these methods involve detecting the face and eyes using a detector such as the Viola-Jones algorithm. Then, the movement in the eye area is evaluated using techniques such as estimating optical flow, sparse tracking, or frame-to-frame intensity differences with adaptive thresholding. Finally, the algorithm determines whether the eyes are covered by eyelids or not.

At present, there are facial landmark detectors available that can accurately capture various key points on a human face image, including the corners of the eyes and the eyelids, with high reliability in real-time. These landmark detectors are advanced and use a regression approach, where a mapping is learned from an image to the positions of the landmarks or another landmark parametrization. They are trained on datasets that contain images taken in diverse settings, which makes them robust to

challenges like changes in illumination, various facial expressions, and moderate non-frontal head rotations.

Proposed method:

To blink is to quickly shut and then reopen one's eyes., and the pattern of blinks varies slightly from person to person, including differences in speed, degree of eye squeezing, and blink duration. Typically, an eye blink lasts anywhere between 100 to 400 milliseconds. In this paper, it is proposed to use advanced facial landmark detectors to locate the eyes and define the shape of the eyelids in an image. Based on the landmarks detected, the eye aspect ratio (EAR) is computed as an indicator of the degree of eye-opening. However, since the EAR value in each frame may not be able to accurately detect eye blinks, a classifier is trained to analyse a longer sequence of frames. When an eye is open, the EAR value remains relatively stable, but it gradually decreases towards zero as the eye closes.

The article introduced an algorithm capable of detecting eye blinks in real-time. It was shown that facial landmark detectors based on regression techniques can accurately estimate the degree of eye openness. These detectors are also highly robust against various challenges, such as low image quality (mostly due to low resolution) and real-world factors like non-frontal head positions, poor lighting, and facial expressions. The proposed approach, which employs a Support Vector Machine (SVM) and considers a temporal window of the eye aspect ratio (EAR), performs better than the EAR thresholding method.

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

2.3 Driver Drowsiness detection using ANN image processing

Authors: S. Moca¹, T. Vesselenyi¹, B. Tătaru¹, A. Rus¹, T. Mitran¹.

Publication year: 2017.

This study aimed to explore the feasibility of developing a drowsiness detection system for car drivers using three methods: processing of EEG and EOG signals, and analysis of driver images based on eye state (open or closed) classification. The EEG and EOG methods are used to measure brain activity and signals from the muscles responsible for eye movement, respectively. On the other hand, the eye image analysis method involves observing the state of the eye, whether it is open or closed.

The EEG and EOG sensors Electrodes need to be positioned on specific parts of the body and be connected through conductive gel or wires, causing discomfort to the user. However, advancements the problems associated with traditional EEG methods may be addressed by utilizing advancements in materials science and MEMS technology, including the application of dry electrodes for EEG.

The advancement of EEG technology has been largely fuelled by the development of brain-computer interfaces for various applications, including devices to assist people with disabilities. One of the primary goals of recent EEG research is to differentiate between low and high alpha rhythm peaks, which can be used to determine a person's level of alertness. The study involved using EOG signals from three sensors (EOG1, EOG2, EOG3) to identify four distinct signal types after pre-processing. By combining these signal types, researchers were able to determine the direction of eye movements (upward, downward, leftward, and rightward), providing the necessary information to differentiate between alert and drowsy states.

The researchers used MATLAB Neural Network Toolbox and Deep Learning Toolbox's autoencoder module to determine if these tools could be used to classify driver drowsiness based on images. They acquired 200 images of a driver during normal driving, with half showing open or half-open eyes and the other half showing closed

eyes. The hypothesis was that closed eyes would indicate drowsiness, while open or half-open eyes would indicate an alert state. They used a one-layer artificial neural network for analysis.

CHAPTER 3

Methodology

3.1 Proposed Methodology

3.1.1 Existing System:

There are several existing systems for detecting driver drowsiness, and they are:

- Various technologies can detect drowsiness in drivers to prevent accidents. Video-based systems use cameras to monitor the driver's face and identify signs of drowsiness such as drooping eyelids, head nodding, or yawning.
- Infrared-based systems use infrared sensors to detect changes in skin temperature, which can indicate drowsiness.
- EEG-based systems utilize electrodes on the driver's scalp to measure brain activity and identify changes in brain waves that correspond to drowsiness.
- Wearable devices that monitor changes in heart rate, breathing, and movement can also indicate drowsiness.
- Steering-based systems use sensors in the steering wheel to detect changes in grip strength or steering behaviour that can indicate drowsiness.

3.1.2 Proposed System:

In this system, instead of existing systems, an alternative approach was used:

- Artificial intelligence and machine learning algorithms are utilized in machine learning-based systems to detect drowsiness. These systems use data from multiple sources, such as video and sensor data, to identify patterns and indications of drowsiness.

A Convolutional Neural Network (CNN) is the model used in this scenario, which is frequently used for image classification and multi-class classification of images.

A Convolutional Neural Network (CNN) is the model used in this scenario, which is frequently used for image classification and multi-class classification of images. The CNN comprises convolution layers that contain adaptable filters. The filters are moved across the input in a forward propagation process, with each movement known as a stride. The CNN model enhances the accuracy of the system.

A camera captures continuous images of the driver's face, and a face detection process identifies the driver's face. The driver's face is then classified as either drowsy or not drowsy using a CNN-based classification model. The KCF and RESNET CNN are utilized to construct the classification model.

3.2 Proposed Techniques:

3.2.1 Artificial Intelligence (A.I):

Artificial Intelligence (AI) is a field of computer science that aims to create intelligent machines that can learn from experience, reason, and make decisions based on data. AI has made significant progress in recent years and is now being used in a wide range of applications, from self-driving cars to medical diagnosis and treatment.

There are different types of AI, including rule-based or symbolic AI, machine learning, and deep learning. Rule-based or symbolic AI uses pre-defined rules and logic to make decisions based on a set of if-then statements. Machine learning is a type of AI that allows machines to learn from data without being explicitly programmed. It uses algorithms to identify patterns in data and make predictions based on those patterns. Deep learning is a type of machine learning that uses neural networks to simulate the way the human brain works. It can process vast amounts of data and make predictions with high accuracy.

AI is used in many applications, including natural language processing (NLP), computer vision, robotics, healthcare, and finance. In natural language processing, AI is used to analyse and understand human language, enabling machines to interact with humans more effectively. In computer vision, AI is used to analyse images and videos,

allowing machines to recognize and identify objects, people, and other visual information. In robotics, AI is used to create intelligent robots that can perform tasks autonomously. In healthcare, AI is used to analyse medical data and make predictions about diseases, allowing doctors to provide better diagnoses and treatments. In finance, AI is used to analyse financial data and make predictions about markets, allowing investors to make better decisions.

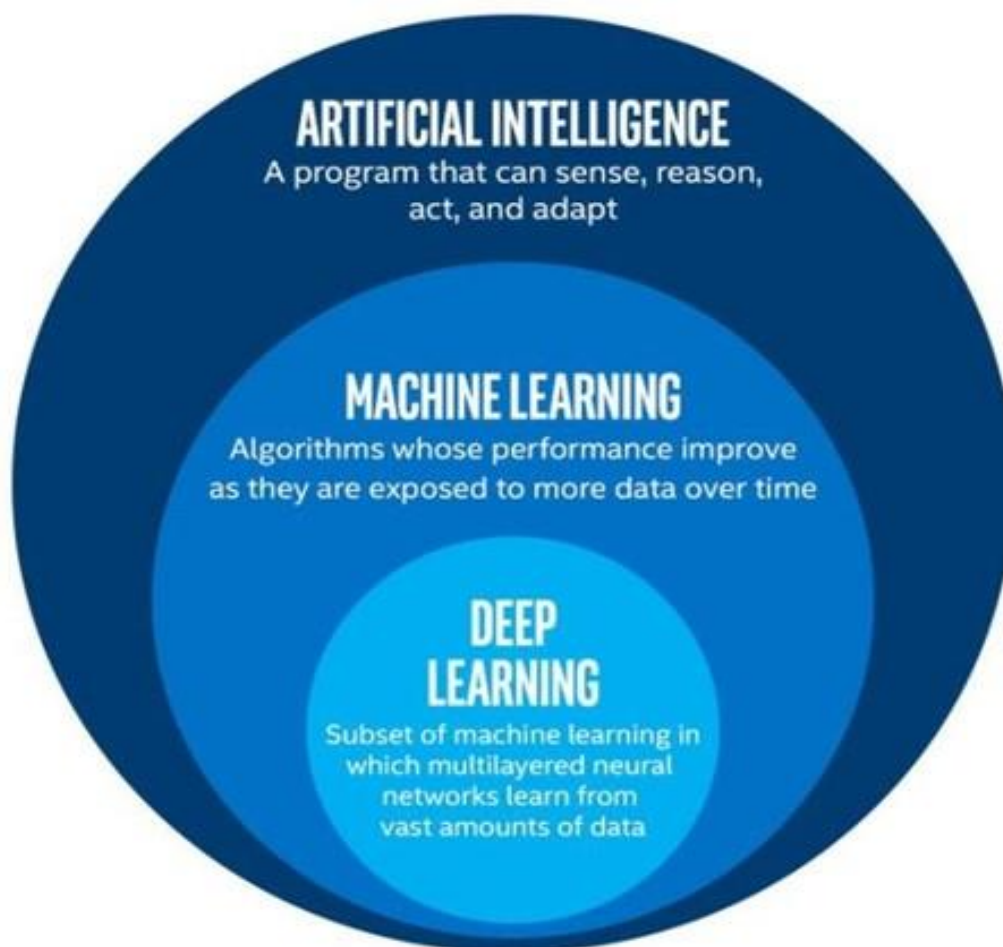


Figure 1 Venn Diagram of A.I

3.2.2 Machine Learning:

Machine learning is a subset of artificial intelligence (AI) that allows computer systems to automatically learn and enhance their performance by learning from experience without the need for explicit programming. It is a field of computer science that involves developing computer programs that utilize algorithms and statistical models to recognize and learn patterns from data. In recent years, significant progress has been made in this field, and the main aim of machine learning is to create programs that can access data and use it to improve their performance autonomously. Although there are various machine learning algorithms available, currently the three primary techniques being used are supervised, unsupervised, and reinforcement learning.

Although there are various types of machine learning algorithms that are utilized for use-cases, currently, the three primary techniques being used are:

- Supervised ML Algorithm
- Unsupervised ML Algorithm Reinforcement
- ML Algorithm

Out of the various types of machine learning algorithms available, the one utilized in this system is a supervised machine learning algorithm.

3.2.3 Deep Learning:

Deep learning is a form of artificial intelligence that emulates the human brain's ability to analyse data and detect patterns to support decision-making. It is a subdivision of machine learning in AI that employs networks capable of unsupervised learning from unstructured or unlabelled data. This technique is also known as deep neural learning or deep neural networks.

Deep learning is a machine learning technique that uses complex algorithms capable of processing and making decisions based on unstructured data without supervision. This approach enables deep learning systems to recognize patterns and carry out tasks such as object and speech recognition, and language translation.

Deep learning is a subfield of artificial intelligence that emulates the way human brains process information to perform various tasks, including speech recognition, object detection, language translation, and decision-making. Unlike traditional machine learning methods, deep learning can learn autonomously without human intervention and can handle unstructured and unlabelled data. This technology has diverse applications, such as preventing fraud and money laundering, among other use cases.

3.2.4 Neural Networks:

Neural networks are artificial systems that are modelled after the structure and function of biological neural networks. These networks are capable of learning and adapting to new information without the need for explicit instructions. Instead, they analyse datasets and examples to identify patterns and relationships on their own.

A neural network is made up of various components such as neurons, connections, biases, weights, a propagation function, and a learning rule. The neurons receive input from previous neurons, and each has an activation function, threshold, and output function. The connections between neurons have weights and biases, which control how information is passed between them. The propagation function calculates the input and output of each neuron based on the function of the preceding neurons and their corresponding weights. The learning rule is responsible for adjusting the weights and thresholds of the network's variables.

In neural networks, units across multiple layers are connected to one another, with each connection carrying a weight that determines the impact of one unit on the other. The network receives data at the input layer, which is then transmitted through various layers before producing the final output at the output layer. Along the way, the network learns from the data and gains a deeper understanding of it, allowing it to make accurate predictions or classifications.

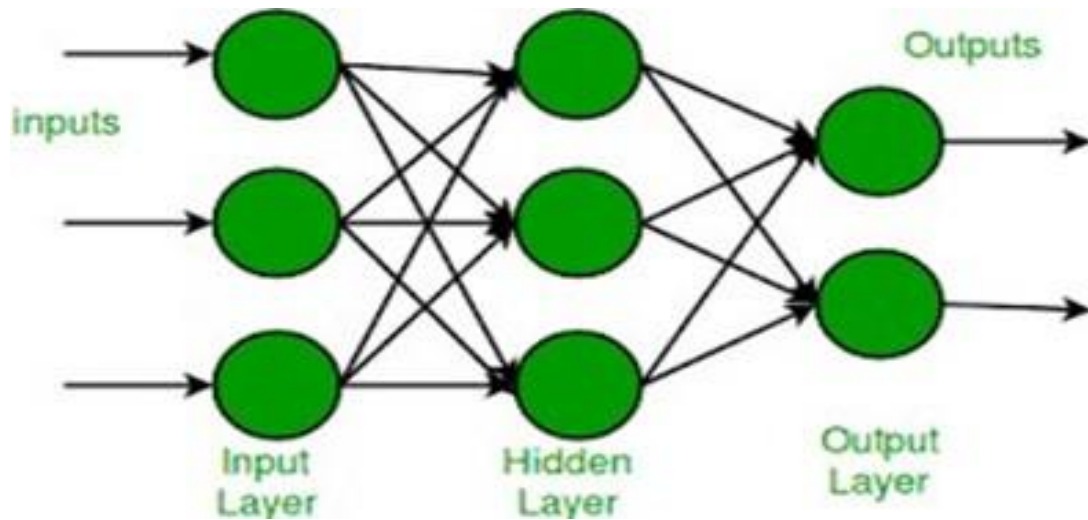


Figure 2 Layers in Neural Network

3.2.5 Convolution Neural Network (CNN):

A Convolutional Neural Network (ConvNet/CNN) is a type of Deep Learning algorithm that can analyse an image input and assign significance to different features or objects within the image using learnable weights and biases. Unlike other classification methods, ConvNets require less pre-processing. In traditional techniques, filters are manually designed, whereas a ConvNet can learn these filters with enough training.

A Convolutional Neural Network (ConvNet/CNN) can effectively capture the spatial and temporal connections within an image by using appropriate filters. Its structure allows for better fitting of the image dataset by reducing the number of parameters and reusing weights. In simpler terms, the network can be trained to comprehend the complexity of the image more accurately.

The term "Convolution" in Convolutional Neural Network (CNN) refers to a specific type of linear operation called convolution in mathematics. Convolution is when two functions are multiplied together to produce a third function that shows how one function changes the shape of the other. In simpler terms, in CNNs, two matrix representations of images are multiplied to create an output that extracts feature from the image.

3.2.5.1 Convolutional Layer:

In neural networks, a convolution layer is a type of layer that performs the convolution operation on input data. It is typically used in image recognition and computer vision tasks.

The convolution layer consists of a set of filters, also called kernels or feature maps, that slide over the input data and perform the convolution operation. Each filter detects a specific feature or pattern in the input data, such as edges or corners. The output of the convolution layer is a set of feature maps, where each map represents the response of one filter to the input data. The feature maps are typically down sampled using a pooling operation, such as max pooling or average pooling, to reduce their size and computational complexity. The parameters of the convolution layer include the size of the filters, the number of filters, and the padding and stride values used during the convolution operation. These parameters are learned during training using backpropagation, allowing the network to learn the best set of filters for a given task.

The Convolution Operation

The convolution operation is a mathematical operation that is commonly used in signal processing and image processing. In the context of neural networks, convolution is used to extract features from input data, such as images or audio signals.

The convolution operation involves sliding a small matrix, called a kernel or filter, over the input data. At each position of the kernel, the values of the kernel and the corresponding values of the input data are multiplied together and then summed. The result is a single output value, which represents the degree of similarity between the input data and the kernel at that position. The process of sliding the kernel over the input data is repeated for every position in the input data, resulting in a new output data structure. The size of the output data structure is typically smaller than the input data structure, depending on the size of the kernel and the amount of padding used.

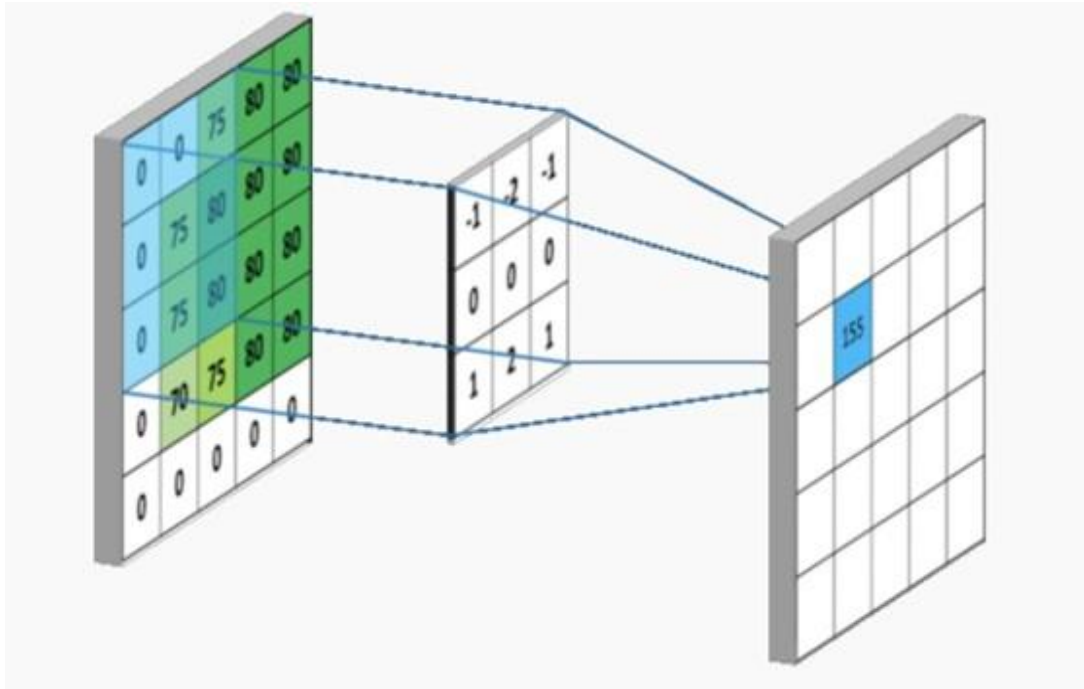


Figure 3 Operation of Convolution

Stride

Stride refers to the number of pixels shifts that a filter makes across the input matrix. The filter moves from left to right across the width of the image using a specified stride value. Once the filter reaches the end of the row, it moves down to the beginning of the image (left) with the same stride value and repeats the process until it has traversed the entire image. If the stride value is 1, the filters move one pixel at a time. If the stride value is 2, the filters move two pixels at a time. The diagram below illustrates how convolution operates with a stride value of 2.

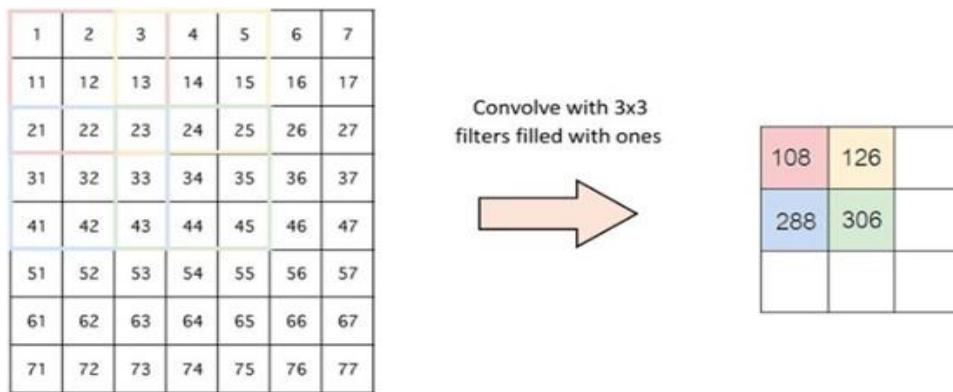


Figure 4 Stide

3.2.5.2 Pooling Layer:

A pooling layer is a common layer in Convolutional Neural Networks (CNNs) that is typically inserted after a convolutional layer. Its main function is to reduce the spatial dimensions of the output feature maps generated by the convolutional layer while retaining the important features learned by the filters. There are two main types of pooling layers: average pooling and max pooling.

3.2.5.2.1 Max Pooling:

In max pooling, the input feature map is divided into non-overlapping regions or windows, typically of size 2x2 or 3x3. For each window, the maximum value within that region is selected and placed in the output feature map, while the other values are discarded. This process is repeated for each window, effectively down sampling the feature map and retaining only the strongest activations.

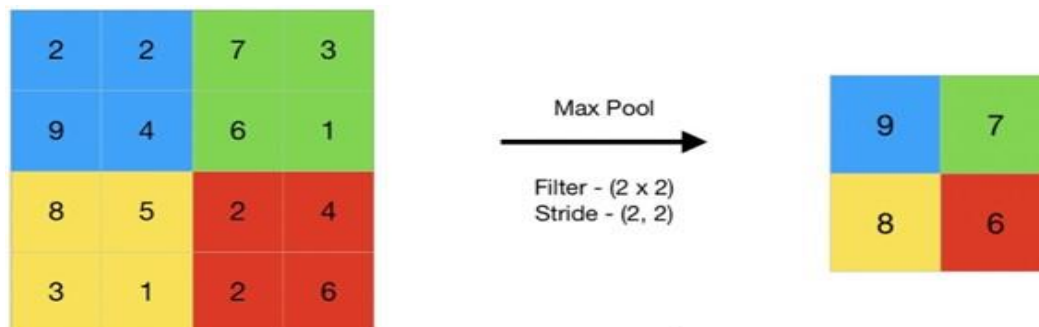


Figure 5 Operation of Max Pooling

3.2.5.2.2 Average Pooling:

In average pooling, the input feature map is divided into non-overlapping regions or windows, typically of size 2x2 or 3x3. For each window, the average value of all the activations within that region is computed and placed in the output feature map. This process is repeated for each window, effectively down sampling the feature map while retaining information about the distribution of activations within each region.

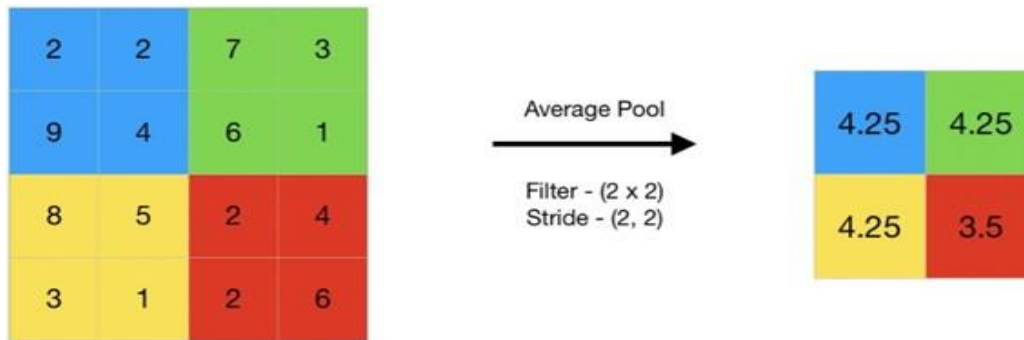


Figure 6 Operation of Average Pooling

3.2.5.2.3 Global Pooling:

Global pooling is a specific type of pooling layer where the entire feature map is reduced to a single value, instead of dividing the feature map into regions. The most common type of global pooling is global average pooling, where the average value of all the activations in the feature map is computed and placed in the output. Global pooling is often used in the final layers of a CNN for classification tasks, where the output of the network needs to be a fixed-size vector representing the probability distribution over the possible classes.

3.2.5.3 Fully Connected Layer:

In deep learning, a fully connected layer (FC layer) is a type of layer in a neural network where all the neurons in one layer are connected to all the neurons in the next layer. This means that every input neuron is connected to every output neuron, and each connection has a corresponding weight and bias.

A fully connected layer is also known as a dense layer or a linear layer. It is typically used as the last layer in a neural network, where it performs the final classification or regression of the input data. The output of a fully connected layer is calculated by taking a weighted sum of the inputs and adding a bias term, followed by an activation function. The weights and biases in the fully connected layer are learned through a process called backpropagation, which is a type of supervised learning algorithm.

3.2.5.4 Dropout Layer:

Dropout is a regularization technique used in Neural Networks to prevent overfitting. It is implemented as a layer in the network architecture, called the dropout layer.

The dropout layer randomly selects a subset of neurons in the previous layer and sets their outputs to zero during training. This means that the information flow through those neurons is temporarily removed from the network, and the remaining neurons must learn to work together to compensate for the missing information. The dropout layer is applied during the training phase only, and the full set of neurons is used during testing.

3.2.5.5 RESNET

ResNet (short for "Residual Network") is a type of deep neural network architecture that was introduced in 2015 by Microsoft Research. It was designed to address the problem of vanishing gradients in deep neural networks, which can make it difficult for the network to learn effectively.

The key idea behind ResNet is the use of residual connections, which allow the network to "skip" over layers and make it easier for gradients to flow back through the network during training. In a standard neural network, each layer applies a set of transformations to the input, but in a ResNet, some of the layers have a "shortcut" connection that adds the input to the output of the layer. This creates a "residual" that can be passed forward to the next layer, allowing the network to learn more complex and deeper representations.

3.2.5.5.1 Residual Blocks:

A Residual Block is the fundamental building block of a Residual Network (ResNet) architecture. It consists of one or more convolutional layers, followed by a set of shortcut connections that allow the network to bypass one or more layers and pass information directly from one layer to another.

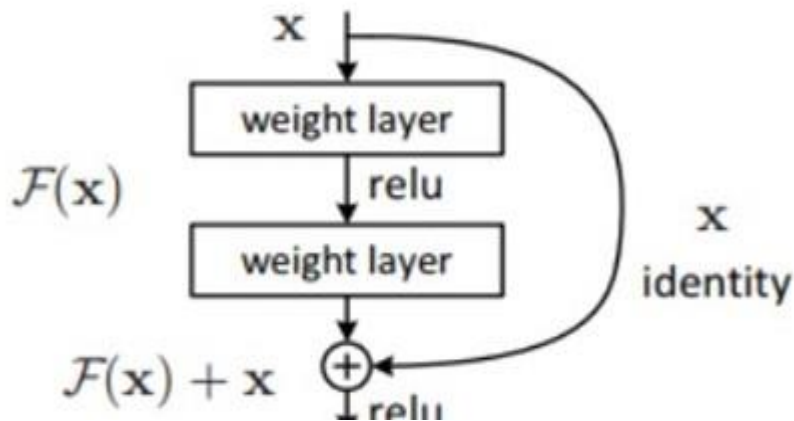


Figure 7 Residual Blocks

The shortcut connection in a Residual Block is a skip connection that adds the output of one or more layers to the output of the block. The idea behind this is that if the input of a layer can be represented by the sum of its output and the input to the layer, then the layer should learn a residual mapping instead of a direct mapping. By doing so, the network can learn more efficient and accurate representations, as it is easier for the network to learn the difference between the output and input rather than learning the direct mapping.

The residual block has been shown to be highly effective in enabling the training of very deep neural networks. By adding shortcut connections between the layers, the gradient can flow more easily through the network during backpropagation, which reduces the vanishing gradients problem and enables the training of deeper networks.

3.2.5.5.2 Architecture Of RESNET:

The architecture of a 34-layer plain network is based on VGG-19, with the addition of skip connections or shortcut connections. These connections are implemented through residual blocks, which convert the architecture into a residual network. A diagram of this can be seen in the figure below.

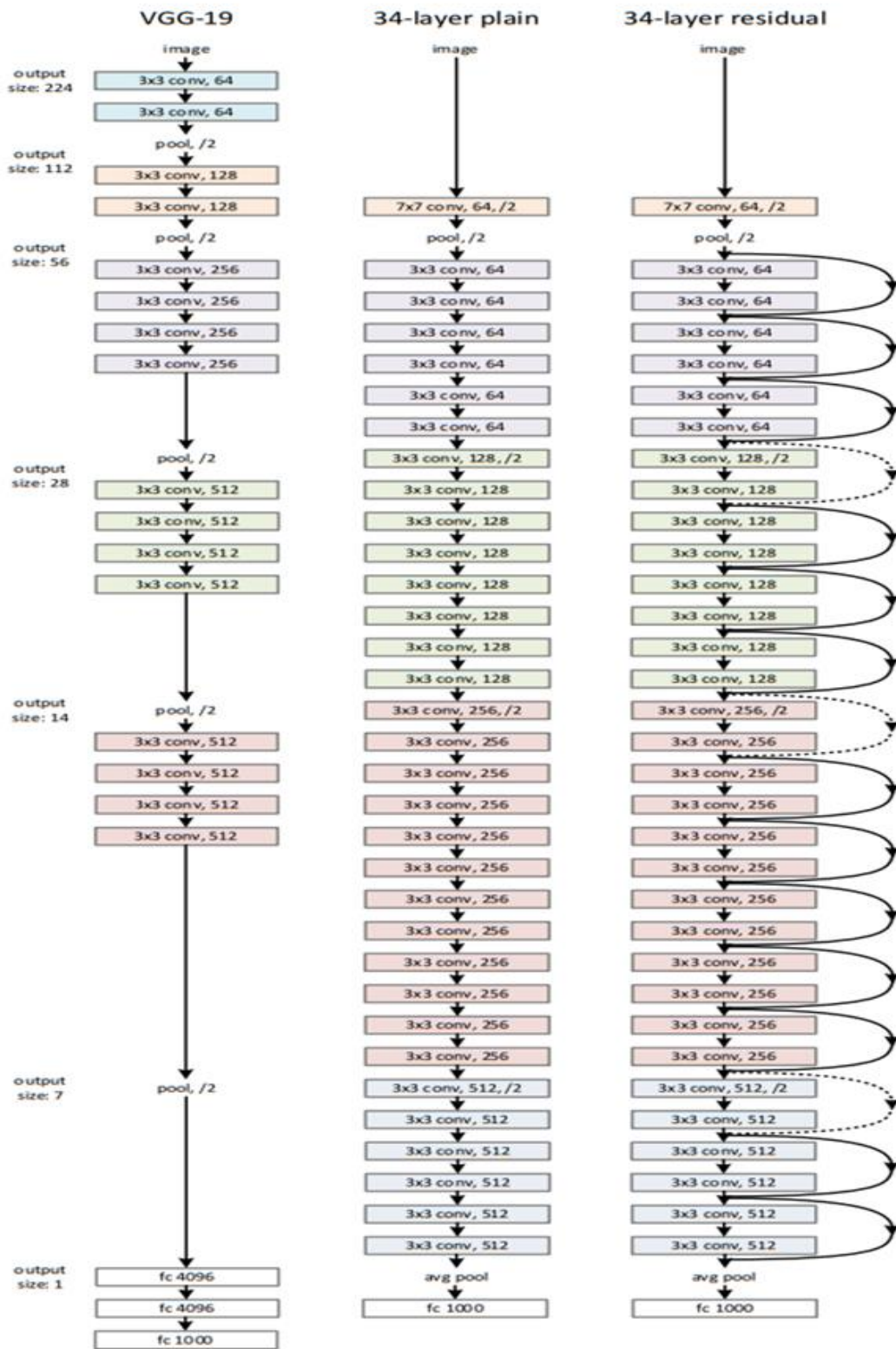


Figure 8 Architecture of ResNet

3.2.5.5.3 Using ResNet with Keras:

Keras is a deep-learning library that is available for free and can be used on top of TensorFlow. Within Keras, there is a feature called Keras Applications, which offers different versions of ResNet.

- ResNet50
- ResNet50V2
- ResNet101
- ResNet101V2
- ResNet152
- ResNet152V2

3.2.6 Back Propagation:

Backpropagation is a widely used algorithm for training artificial neural networks. It is an optimization algorithm that calculates the gradients of the loss function with respect to the weights of the network, allowing the weights to be updated in the direction that minimizes the loss.

The backpropagation algorithm works by first propagating the input forward through the network, calculating the output of each neuron, and finally calculating the loss between the predicted output and the actual output. It then propagates the error back through the network, calculating the gradient of the loss with respect to each weight in the network. This is done by applying the chain rule of calculus to calculate the derivative of the loss with respect to each intermediate output in the network, and then using these derivatives to calculate the derivative of the loss with respect to each weight. Once the gradients have been calculated, they are used to update the weights of the network using an optimization algorithm such as gradient descent. The optimization algorithm iteratively updates the weights in the direction that reduces the loss function.

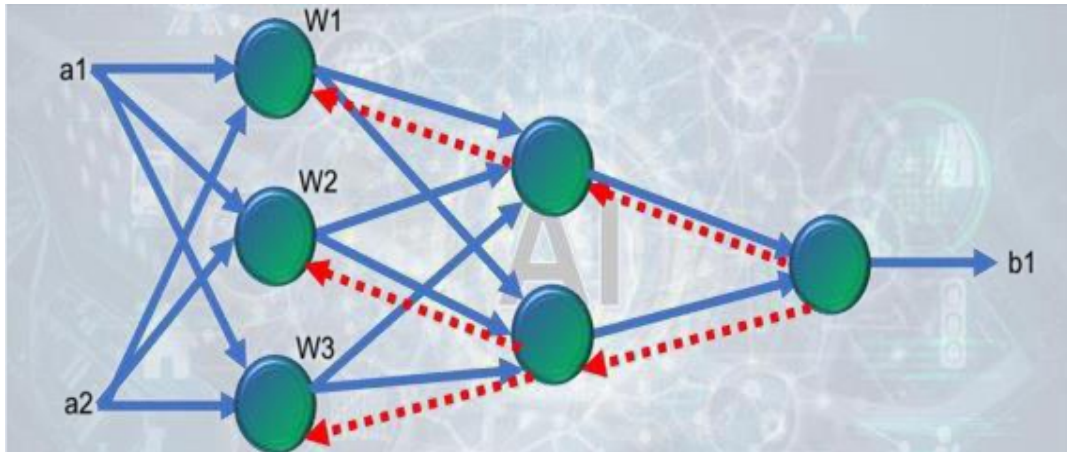


Figure 9 Back Propagation

3.2.7 Activation Functions:

Activation functions are a key component of artificial neural networks, used to introduce non-linearity into the output of a neuron. They are applied to the weighted sum of the inputs and bias of a neuron to produce its output. The choice of activation function has a significant impact on the performance and efficiency of a neural network.

There are several types of activation functions, including:

1. Sigmoid function
2. Rectified Linear Unit (ReLU) function.
3. Hyperbolic Tangent (tanh) function
4. SoftMax function

3.2.8 Training:

Deep learning neural networks are designed to learn how to map inputs to outputs. This is accomplished by adjusting the weights of the network in response to the errors made by the model on the training dataset. These adjustments are made continuously to minimize the error until the learning process either comes to a stop or an acceptable level of accuracy is achieved. In other words, the goal of the network is to continually refine its mapping function to produce more accurate and precise results.

The optimization problem in deep learning neural networks is typically solved using the stochastic gradient descent algorithm. This algorithm utilizes the backpropagation algorithm to update the model's parameters in each iteration. In other words, the stochastic gradient descent algorithm is responsible for adjusting the weights of the network based on the errors calculated during the backpropagation process.

A neural network model learns how to map a particular set of input variables to the output variable using examples. The goal is to ensure that this mapping works well not only on the training dataset but also on new, unseen examples. This ability to function effectively on specific as well as new examples is known as the model's ability to generalize. Essentially, the model must be able to apply what it has learned to new, unseen data while still producing accurate results.

- Loss function: This is a function used to assess the model's performance on the training dataset, based on a particular set of weights.
- Epochs: The number of times the model goes through the entire training dataset before the training process is stopped.

3.2.8.1 Test Loss:

Test loss is a metric that is commonly used in machine learning to evaluate the performance of a trained model on a dataset that was not used during the training process. It measures the difference between the predicted outputs of the model and the actual outputs for a set of input data in the test dataset.

The test loss is computed using a loss function that compares the predicted output of the model to the actual output for each input in the test dataset. The loss function used depends on the type of problem being solved.

3.2.8.2 Test Accuracy:

Test accuracy is a metric used to evaluate the performance of a machine learning model on a test dataset. It measures the percentage of correct predictions made by the model on the test dataset.

To compute test accuracy, the model is first trained on a training dataset, and its performance is evaluated on a validation dataset. Once the model is trained and tuned to perform well on the validation dataset, it is then tested on a separate test dataset to evaluate its performance on new, unseen data.

The test accuracy is calculated as the percentage of correctly predicted labels in the test dataset.

3.2.8.3 Validation Loss:

During the training process, a model is typically trained on a training dataset, and its performance is evaluated on a separate validation dataset. The validation dataset is used to monitor the model's performance and to prevent overfitting, which occurs when a model performs well on the training data but poorly on new data.

Validation loss is computed by evaluating the model on the validation dataset using a loss function, such as mean squared error or cross-entropy. The validation loss is then used to adjust the model's parameters to improve its performance.

3.2.8.4 Validation Accuracy:

During the training process, a model is typically trained on a training dataset, and its performance is evaluated on a separate validation dataset. The validation dataset is used to monitor the model's performance and to prevent overfitting, which occurs when a model performs well on the training data but poorly on new data.

Validation accuracy is computed by evaluating the model on the validation dataset and measuring the percentage of correct predictions.

3.2.9 Train Dataset:

The training data is a subset of data used to teach a model to identify patterns and relationships between input and output variables. The training data is used to adjust the model's parameters so that it can make accurate predictions on new, unseen data.

3.2.10 Testing:

When it comes to Machine Learning models, the term "testing" typically refers to evaluating the accuracy or precision of the model. This is different from the use of the term in traditional software development.

CHAPTER 4

Software Used

4.1 Python

Python is a high-level, interpreted programming language that was first released in 1991 by Guido van Rossum. It is designed to be easy to read and write, making it a popular choice for beginners and experienced programmers alike. Python's syntax is simple and elegant, with an emphasis on readability and simplicity.

One of the key features of Python is its extensive library of modules, which allow developers to perform a wide range of tasks without having to write code from scratch. Python is also known for its versatility, with applications ranging from web development and data analysis to scientific computing and machine learning.

Python's popularity has grown steadily over the years, due in part to its active community of developers who have contributed to its open-source codebase. Today, Python is widely used in academia, industry, and government, and it is considered one of the most popular programming languages in the world. Python is supported on a wide range of platforms, including Windows, Linux, and macOS, and it has become a popular choice for scripting and automation tasks, as well as web development using frameworks like Django and Flask. With its powerful and flexible syntax, rich library of modules, and active community, Python continues to be a popular choice for developers and businesses alike.

4.2 Jupyter Notebook

Jupyter Notebook is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It supports over 40 programming languages, including Python, R, and Julia.

Jupyter Notebook provides an interactive computing environment, where users can write and execute code in cells. Each cell can contain code, markdown text, or raw text. The output of a code cell is displayed directly below the cell, allowing users to see the results of their code immediately.

Jupyter Notebook also supports the creation of interactive visualizations, using libraries such as Matplotlib, Bokeh, and Plotly. Jupyter Notebook is widely used in data science and scientific computing, as it provides a powerful and flexible environment for data exploration, visualization, and analysis. Its interactive nature makes it ideal for rapid prototyping and experimentation, while its ability to combine code, data, and narrative text makes it an effective tool for communication and collaboration.

4.3 Libraries

4.3.1 Open Source-Computer Vision Library:

OpenCV is a vast open-source library for computer vision, image processing, and machine learning. Its applications are crucial in real-time operations for modern systems. It allows for the identification of human handwriting, faces, and objects in images and videos. In conjunction with other libraries, such as NumPy, Python can analyze the OpenCV array structure to process images. To identify image patterns and their distinct characteristics, vector space is used, and mathematical operations are carried out on these features.

The initial version of OpenCV was 1.0, and it is available under a BSD license, making it free for commercial and academic purposes. OpenCV has interfaces for Java, Python, C++, and C and is supported by various operating systems, including Mac OS, Windows, Linux, iOS, and Android. Its primary objective was to support real-time applications, which is why it is built with optimized C/C++ code to take advantage of multi-core processing.

OpenCV Functionality:

- Image and video input/output: OpenCV can read and write image and video files in various formats, such as JPEG, PNG, BMP, and MPEG.
- Image processing: OpenCV provides a wide range of image processing functions such as filtering, thresholding, edge detection, morphology, and many more.
- Feature detection and extraction: OpenCV includes algorithms for detecting and extracting various features from images, such as corners, blobs, and lines.
- Object detection and recognition: OpenCV provides several object detection and recognition algorithms, such as face detection, pedestrian detection, and object recognition.

Applications of OpenCV:

There are lots of applications which are solved using OpenCV, some of them are listed below:

- Object detection and recognition: OpenCV is often used for object detection and recognition in various fields such as security, surveillance, and robotics. For example, it can be used to detect faces, pedestrians, and vehicles in real-time video streams.
- Medical imaging: OpenCV is used in medical imaging applications for tasks such as image segmentation, classification, and analysis. It can be used for applications such as tumour detection, image registration, and image-guided surgery.
- Robotics: OpenCV is used in robotics applications for tasks such as object detection, navigation, and control. For example, it can be used to detect obstacles or track the position of a robot in a given environment.
- Gaming: OpenCV can be used for gaming applications such as motion capture, gesture recognition, and facial expression recognition.

4.3.2 Numerical Python :

NumPy (short for Numerical Python) is a popular Python library used for numerical computing and scientific computing. It provides a powerful array computing functionality and a wide range of mathematical functions for working with arrays, matrices, and other numerical data structures.

NumPy provides an array object that is like a list or a Python array, but with additional features such as fast and efficient indexing, slicing, and broadcasting. NumPy arrays are also homogenous, meaning that all elements in an array must have the same data type, which allows for more efficient memory allocation and computation. In addition to arrays, NumPy provides a wide range of mathematical functions for working with arrays, including basic arithmetic operations, linear algebra, Fourier transforms, random number generation, and more. It also integrates well with other scientific computing libraries such as SciPy, Matplotlib, and Pandas.

4.3.3 Tensor Flow:

TensorFlow is an open-source software library initially created by the Google Brain Team, comprising engineers and researchers working within Google's Machine Intelligence research organization, to facilitate machine learning and deep neural network research. Despite its origins, TensorFlow's applicability transcends its initial purpose and is widely employed in various domains. Google released TensorFlow as an open-source software in November 2015.

TensorFlow's success stems from several factors, including its computational graph concept, automatic differentiation, and its adaptable Python API structure. These features enable programmers to tackle real-world problems with TensorFlow more easily. The unique problem-solving approach employed by TensorFlow's engine contributes significantly to its popularity, allowing for efficient resolution of machine learning problems.

4.3.4 Keras:

Keras is an open-source neural network library written in Python that is designed to provide a simple and efficient way to build and train deep learning models. It was

developed by François Chollet and is now a part of the TensorFlow library. Keras is known for its user-friendliness, modular design, and flexibility. It is widely used by data scientists and machine learning practitioners.

Key features and concepts of Keras:

- **User-friendly API:** Keras provides a high-level, easy-to-use API that abstracts away much of the complexity of deep learning. It allows users to quickly build and train models without needing to have an in-depth understanding of the underlying mathematics.
- **Modularity:** Keras is designed with a modular architecture that makes it easy to build complex models. Models can be built using pre-built building blocks, called layers, that can be combined in various ways to create different architectures.
- **Compatibility with Python:** Keras is built in Python, which means that it is compatible with many other Python libraries and tools commonly used in data science and machine learning.
- **Pre-trained models:** Keras provides access to a number of pre-trained models that can be used for a variety of tasks, including image classification, object detection, and natural language processing.

4.3.5 Matplotlib:

Matplotlib is a popular Python library used for data visualization. It provides a variety of tools for creating high-quality plots, graphs, and charts, allowing users to visualize and analyse data in a clear and concise manner.

Matplotlib can be used to create a wide range of plots, including line plots, scatter plots, bar plots, histograms, heatmaps, and more. It provides a range of customization options for creating plots that meet specific requirements, including the ability to modify colours, fonts, labels, axes, and annotations. Matplotlib is an open-source library and is actively maintained and developed by a community of contributors. It is widely used in various fields, including scientific research, finance, engineering, and machine learning.

4.3.6 OS module in Python:

The OS module in Python provides a way to interact with the operating system on which the Python interpreter is running. It provides a way to access file and directory management functionality, system information, and process management. The module includes functions for creating, deleting, moving, and renaming files and directories, as well as executing shell commands, accessing environment variables, and creating child processes. The `os.path` submodule provides functions for working with file paths and manipulating file and directory names. The OS module is a powerful tool for interacting with the operating system from within a Python program. It allows developers to create scripts that can automate common tasks, such as file and directory management, system administration, and process control.

The `os` module in Python provides several submodules that offer additional functionality for interacting with the operating system. Here are some of the key submodules of the `os` module:

- `os.path`: This submodule provides functions for working with file paths and manipulating file and directory names. Some of the functions in this submodule include.
- `os.system`: This submodule provides functions for executing shell commands from within a Python script. The `os.system()` function can be used to execute a command in the shell and return the output.
- `os.environ`: This submodule provides access to the environment variables on the system. The `os.environ` dictionary contains key-value pairs for each environment variable.
- `os.fdopen`: This submodule provides a way to open file descriptors using the file object interface. The `os.fdopen()` function takes a file descriptor and returns a file object that can be used for reading and writing.

4.3.7 Dlib:

Dlib is a popular C++ library for developing machine learning and computer vision applications. It is known for its high performance and flexibility and is widely used by researchers and developers in the field. dlib includes several pre-trained models for various tasks, such as face detection, facial landmark detection, object detection, and image segmentation. These models can be easily integrated into applications and used to quickly achieve state-of-the-art performance. In addition to pre-trained models, dlib provides tools for training custom models. These include support for various types of machine learning algorithms, such as SVMs, decision trees, and neural networks, as well as efficient optimization algorithms for training these models.

dlib also includes several utility classes and functions for working with images, matrices, and other data structures commonly used in machine learning and computer vision. It is designed to be portable across platforms and can be used on a variety of operating systems, including Windows, macOS, Linux, and Android.

Some of the main features of DLIB include:

- Face detection and recognition: DLIB provides pre-trained models for detecting faces in images and videos, as well as models for recognizing individual faces.
- Object detection: DLIB includes tools for training and using object detectors, which can be used to identify and locate objects of interest in images and videos.
- Image segmentation: DLIB provides algorithms for segmenting images into different regions based on colour, texture, and other features.
- Machine learning: DLIB includes tools for training and using various types of machine learning models, such as support vector machines (SVMs), decision trees, and deep neural networks.

CHAPTER 5

Experimental Results

Input:

The input for system is a human face image. The system has MRL eye dataset consisting of 80,000 cropped images of eye region, and the following are driver input images which were given to detection system.



Figure 10 Driver Input images

Output:

The input is captured through camera, then face tracking and detection is done through MC-KCF. Once face detection and image resizing have been carried out, the resulting images are as follows:



Figure 11 Alert Images



Figure 12 Drowsy images

From the above image's features like eyes are extracted. The extracted features are passed through ResNet CNN and produce buzzer alert if the driver is in drowsy state.

5.1 Results through live face tracking:

5.1.1 Frames recognized as drowsy:

The system works by continuously analysing a stream of video input to detect signs of driver drowsiness. When the system detects drowsiness, it will highlight the driver's eyes using a red rectangle and emit a beep sound as an alert to make the driver aware of their drowsiness. The detection of drowsiness is achieved by utilizing a well-trained ResNet Convolutional Neural Network (CNN) model.

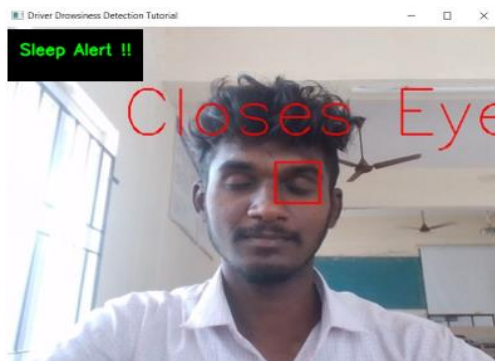


Figure 13 Person 1 in drowsy state

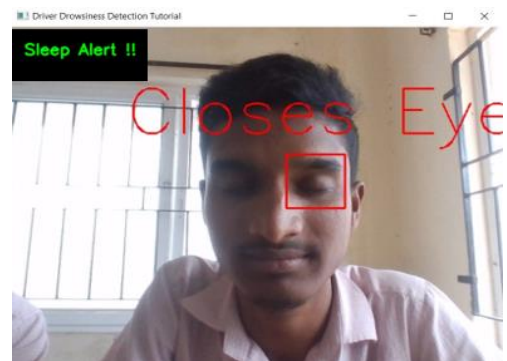


Figure 14 Person 2 in drowsy state

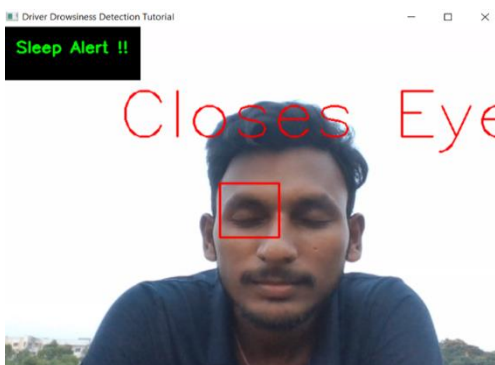


Figure 15 Person 3 in drowsy state



Figure 16 Person 4 in drowsy stat

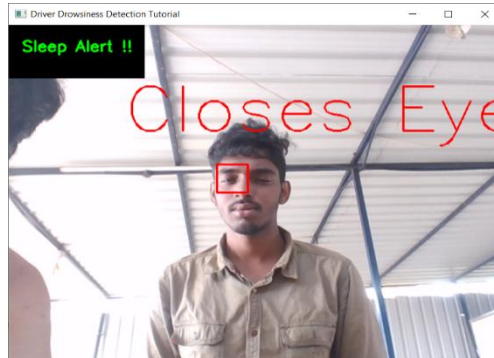


Figure 17 Person 5 in drowsy state

By observing above figures in this section, the system detects eye region in whole face expression and as the eyes are closed, the detecting system alerts driver through buzzer sound.

5.1.2 Frames recognised as not drowsy:

The system operates by continuously analysing a live video stream to determine if the driver is not drowsy. If the system detects that the driver is not drowsy, it will highlight their eyes using a green-coloured rectangle and will not generate an alert. The detection of driver drowsiness is accomplished using a well-trained ResNet Convolutional Neural Network (CNN) model.

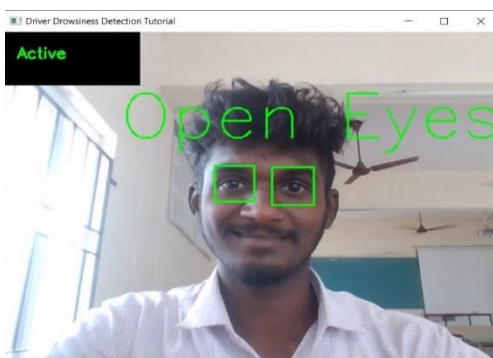


Figure 18 Person 1 in alert state

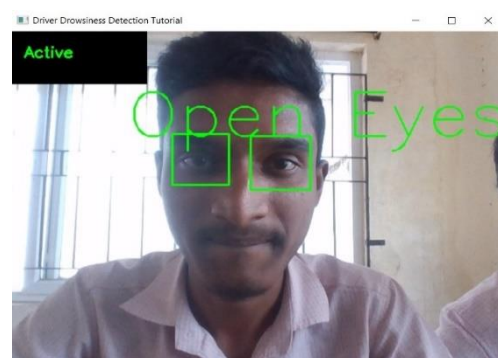


Figure 19 Person 2 in alert state



Figure 20 Person 3 in alert state

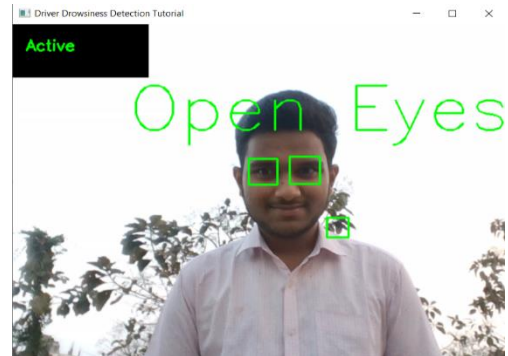


Figure 21 Person 4 in alert state

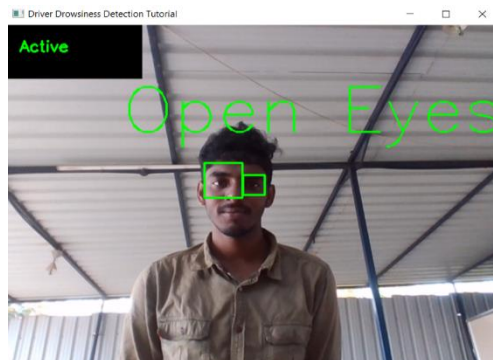


Figure 22 Person 5 in alert state

By observing above figures in this section, the system detects eye region in whole face expression and as the eyes are open, the detecting system states that the driver is in active mode i.e., eyes are not closed.

Results on test data:

Test loss: 0.0183

Test accuracy: 0.9817

Validation loss:0.0167

Validation accuracy:0.9833

In general, test loss is calculated by evaluating dataset which was not used during training. A lower test loss indicates that the model is more accurate and better at generalizing to new data. The test accuracy is calculated as the percentage of correctly

predicted labels in the test dataset. Validation loss is computed by evaluating the model on the validation dataset using a loss function, such as mean squared error or cross-entropy. The goal of training a model is to minimize the validation loss, which indicates that the model is becoming more accurate at predicting outputs for new, unseen data.

Validation accuracy is computed by evaluating the model on the validation dataset and measuring the percentage of correct predictions. The goal of training a model is to maximize the validation accuracy, minimize the validation loss, which indicates that the model is becoming more accurate at predicting outputs for new, unseen data.

From the above test results, model contains minimum test loss and validation loss and maximum validation accuracy, which states that model is well trained and produces better results than previous models in detecting drowsiness.

CHAPTER 6

Conclusion and Future Scope

6.1 Conclusion

This project proposes a new method for driver drowsiness detection using a combination of MC-KCF and ResNet CNN. The proposed method utilizes the advantages of both methods to track eye movements using MC-KCF and classify the driver's face to determine the drowsiness through ResNet CNN in real-time.

Experimental results demonstrated that the proposed method produced high accuracy rates in detecting driver drowsiness based on testing results and through Resnet CNN model, when driver is in drowsy state upto 1sec, the alarm sound is produced and alerts the driver stating that he/she is in drowsy state. The proposed method has the potential to be used in various settings, including in-vehicle systems, to provide real-time feedback to drivers and prevent accidents caused by driver fatigue.

This project also contributes to the field of driver drowsiness detection by providing an overview of existing methods. However, there are few limitations to the proposed algorithm, such as camera placement and light conditions, variability in eye movements, computational complexity but the proposed algorithm work can be extended to other related fields, such as human-computer interaction, facial expression recognition and emotion detection.

Thus, in this project, algorithm for fatigue detection using MC-KCF and ResNet CNN is successfully designed and executed. Overall, this project paper provides a promising approach to driver drowsiness detection that can help improve road safety and prevent accidents caused by driver fatigue.

6.2 Future Scope

The future scope for driver drowsiness detection using MC-KCF and Residual Neural Networks is vast and promising. Here are a few potential areas of development and improvement for this project:

Real-time application: Currently, the implementation of this project requires processing video frames offline, which may not be practical for real-time application. Future work could focus on optimizing the algorithms and hardware to enable real-time processing of video streams, which would make the system more practical and effective for real-world scenarios.

Multi-modal input: While the current implementation of the project relies solely on visual cues to detect drowsiness, incorporating other modalities, such as audio or physiological signals, could improve the accuracy and robustness of the system.

Personalization: People exhibit different signs of drowsiness, and some individuals may exhibit unique cues that are not captured by the current algorithm. Future work could explore how to personalize the drowsiness detection algorithm to individual drivers to improve its accuracy and effectiveness.

REFERENCES

- [1] Snehal S.Bharambe and P.M.Mahajan, "Implementation of Real Time Driver Drowsiness Detection System" IJSR, vol 4 issue1, Jan 2015.
- [2] Wanghua Deng and Ruoxue wu, "Real-Time Driver-Drowsiness Detection System Using Facial Features" IEEE access, vol 7,2019.
- [3] Pratyush Agarwala and Rizul Sharma, "Driver Drowsiness Detection Techniques: Review" EasyChair, Dec 21,2019.
- [4] Kusuma Kumari B.M, "Review on Drowsy Driving: Becoming Dangerous Problem" IJSR, vol 3 issue 1, Jan 2014.
- [5] Tejal Jadhav, Siddhi Gajare, Shubhman Salve and Prof. Hani Patil, "Advanced Driver Assistance System for drivers using Machine Learning and Artificial Intelligence Techniques" IJRASET, vol 10 issue V, May 2022.
- [6] V B Navya Kiran, Raksha R, Anisoor Rahman, Varsha K N and Dr. Nagamani N P, "Driver Drowsiness Detection" IJERT, vol 8 issue 15,2019.
- [7] Jagbeer Singh, Ritika Kanokia, Rishika Singh, Rishita Bansal and Sakshi Bansal, "Driver Drowsiness Detection System-An approach by Machine Learning Application", Journal of Pharmaceutical Negative Results, vol 13, special issue 10, 2022.
- [8] Whui Kim, Kyong Hee Lee, Hyun Kyun Choi and Byung Tae Jan, "A Study on Feature Extraction Methods Used to Estimate a Driver's Level of Drowsiness", IEEE, February 2019[2019 21st International Conference on Advanced Communication Technology (ICACT)].
- [9] Tianyi Hong, Huabiao Qin, "Drivers Drowsiness Detection in Embedded System.", IEEE, December 2007[2007 IEEE International Conference on Vehicular Electronics and Safety].
- [10] Dwipjoy Sarkar and Atanu C, "Real Time Embedded System Application for Driver Drowsiness and Alcoholic Intoxication Detection", IJETT, Volume 10 Number 9, April 2014.
- [11] Fouzia, Roopalakshmi R, Jayantkumar A Rathod, Ashwitha S, Supriya K, "Driver Drowsiness Detection System Based on Visual Features.", IEEE, April 2018[2018

Second International Conference on Inventive Communication and Computational Technologies (ICICCT)].

- [12] SaeidFazli, Parisa Esfehiani, “Tracking Eye State for Fatigue Detection”, ICACEE, November 2012.
- [13] Shruti Wasudeo Deolikar, “ Drowsy Driving Safety Detection”, IRJMETS, vol 05, issue 01, Jan 2023.
- [14] Gao Zhenhai, Le DinhDat, Hu Hongyu, Yu Ziwen and Wu Xinyu, “Driver Drowsiness Detection Based on Time Series Analysis of Steering Wheel Angular Velocity”, IEEE, January 2017[2017 9th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)].
- [15] Seth Weidman, “Deep Learning from Scratch”, O’Reilly ,2019.
- [16] Eli Stevens, Luca Antiga, and Thomas Viehmann, “Deep Learning with PyTorch”, Manning,Jul-2020.
- [17] Aleksandar Čolić , Oge Marques and Borko Furht, “ Driver Drowsiness Detection-Systems and Solutions”, Spring,2014.
- [18] Shehzad Saleem, “Risk assessment of road traffic accidents related to sleepiness during driving: a systematic review”, Eastern Mediterranean Health Journal,vol 28, issue 9, 2022.
- [19] Z Tian and H Qin, “Real-time driver's eye state detection. Proceedings of the IEEE International Conference on Vehicular Electronics and Safety, October 2005.
- [20] Marco Javier Flores, José María Armingol and Arturo de la Escalera, “ Real-Time Drowsiness Detection System for an Intelligent Vehicle”, IEEE, June 2008[2008 IEEE Intelligent Vehicles Symposium].
- [21] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng and Rong Qu, “ A Survey of Deep Learning-based Object Detection”, IEEE access, vol 7,2019.
- [22] Kaiming He, Xiangyu Zhang, “Deep Residual Learning for Image Recognition”,ILSVRC,2015.
- [23] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 37, no. 3, pp. 583–596, Mar. 2015.

- [24] Y. Zhang and C. Hua, “Driver fatigue recognition based on facial expression analysis using local binary patterns,” *Optik*, vol. 126, no. 23, pp. 4501–4505, Dec. 2015.
- [25] R. O. Mbouna, S. G. Kong, and M.-G. Chun, “Visual analysis of eye state and head pose for driver alertness monitoring,” *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1462–1469, Sep. 2013.
- [26] International Organization of Motor Vehicle Manufacturers. Provisional Registrations or Sales of New Vehicles. Available: <http://www.oica.net/wp-content/uploads/>
- [27] Wards Intelligence. World Vehicles in Operation by Country, 2013– 2017. Available: <http://subscribers.wardsintelligence.com/databrowse-world>.
- [28] National Highway Traffic Safety Administration. Traffic Safety Facts 2016. Available: <https://crashstats.nhtsa.dot.gov>.