ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS

FACULTY OF

INFORMATICS

THESIS

OE-NIK Student's name: **Ghedir, Yasmine**

2022 Student's registration number: **T/008796/FI12904/N**

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

NEUMANN JÁNOS

FACULTY OF
INFORMATICS

Driver Drowsiness Detection System

Óbuda University
John von Neumann Faculty of Informatics
Institute for Cyber-physical Systems

DIPLOMA THESIS
TOPIC DESCRIPTION

Student name: **Yasmine Ghedir**
Registration number: T/008796/FI12904/N
Neptun's code: [redacted]

Title of diploma thesis:

Driver drowsiness detection system
Járművezetői fáradtságot érzékelő rendszer

Internal supervisor: Péter Fésüs
External supervisor

Deadline: 15th of May, 2022.

Topic description:

Implement and design a drowsiness detection system for vehicle drivers which is a safety technology that can prevent accidents that are caused by drivers who fell asleep while driving.
Preventing drowsiness while driving, is a major trend and challenge in the domain of accidence avoidance systems. The goal of this research is to reduce the number of traffic accidents by developing an automatic real time system that can benefit from the dynamic behaviour of the human face features and detect the state of the driver based on computer vision and image processing analysis and machine learning model.

The diploma thesis should cover:

- the precise description and detailed analysis of the problem to be solved,
- possible implementation methods and solutions,
- the detailed system plan and its justification,
- presentation of the technologies used in the implementation,
- a detailed description of the implementation process, work phases and experiences,
- the test methods, test plans and test results,
- the concludes and further improvement possibilities of the implemented system,
- the source code of the developed system and the prepared documentation in online uploaded form.

Ph.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * 1. Budapest

Fleiner R.
.....
Dr. Rita Fleiner
head of institute

Term of limitation: **15th of May, 2024.**

The diploma thesis is adequate and can be submitted:

.....
external supervisor

.....
internal supervisor

STUDENT'S DECLARATION

I the undersigned student hereby declare that this thesis is the result of my own work; I have disclosed the references and tools used in an identifiable manner.

The results indicated in my thesis completed may be used by the university and the institution announcing the task for their own purposes free of charge.

Dated: Budapest, 28.04. 2022

student's signature

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Faculty of Informatics

CONSULTATION LOG

Student's name:	Neptun code:	Specialty:
Yasmine Ghedir	[redacted]	Computer Science Engineering

Phone number: Mailing address (e.g. domicile):

[redacted]

Degree project / thesis[1] title in Hungarian:

Járművezetői fáradtságot érzékelő rendszer

Degree project / thesis[2] title in English:
Driver Drowsiness Detection System

Institutional supervisor: External supervisor:

Fésüs Péter
Please write the data in printed capital letters!

Occ.	Date	Content	Signature
1.	17/02/2022	RECOMMENDATION OF SOME FORMAL SOURCE OF PAPERS .	Fésüs Péter
2.	24/02/2022	COLLECTING SOME PAPERS ABOUT RELATED WORK.	Fésüs Péter
3.	03/03/2022	FILTERING THE USEFUL PAPERS.	Fésüs Péter
4.	10/03/2022	DISCUSSING THE THESIS STRUCTURE.	Fésüs Péter

The Consultation log is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Degree project I / Degree project II (BSc) or Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4[3], and is allowed to proceed for reporting / defense[4].

Dated: Budapest, 10/03/2022

WRITING SECOND CHAPTER OF THE THESIS

Fésüs Péter
.....
institutional supervisor

[1] Underline as appropriate
[2] Underline as appropriate
[3] Underline as appropriate
[4] Underline as appropriate

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Faculty of Informatics

CONSULTATION LOG

Student's name: Yasmine Ghedir

Neptun code: [redacted]

Specialty: Computer Science Engineering

Phone number: [redacted]

Mailing address (e.g. domicile): [redacted]

Degree project / thesis[1] title in Hungarian:

Járművezetői fáradtságot érzékelő rendszer

Degree project / thesis[2] title in English:
Driver Drowsiness Detection System

Institutional supervisor: Fésüs Péter

External supervisor:

Please write the data in printed capital letters!

Occ.	Date	Content	Signature
1.	17/03/2022	WRITING THE LITERATURE OF THE THESIS.	
2.	24/03/2022	WRITING SECOND CHAPTER OF THE THESIS	
3.	31/03/2022	REVIEWING THE FIRST CHAPTERS	
4.	7/04/2022	DISCUSSING THE USED TECHNIQUES AND TECHNOLOGIES FOR PRACTICAL WORK.	

The Consultation log is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Degree project I / Degree project II (BSc) or Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4[3], and is allowed to proceed for reporting / defense[4].

Dated: Budapest, 7/04/2022

.....
institutional supervisor

[1] Underline as appropriate
[2] Underline as appropriate
[3] Underline as appropriate
[4] Underline as appropriate

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Faculty of Informatics

CONSULTATION LOG

Student's name: Yasmine Ghedir

Neptun code: [redacted]

Specialty: Computer Science Engineering

Phone number: [redacted]

Mailing address (e.g. domicile): [redacted]

Degree project / thesis[1] title in Hungarian:

Járművezetői fáradtságot érzékelő rendszer

Degree project / thesis[2] title in English:
Driver Drowsiness Detection System

Institutional supervisor: Fésüs Péter

External supervisor:

Please write the data in printed capital letters!

Occ.	Date	Content	Signature
1.	14/04/2022	DISCUSSING THE IMPLEMENTATION OF THE SOLUTION.	Fésüs Péter
2.	21/04/2022	STARTING THE IMPLEMENTATION.	Fésüs Péter
3.	28/04/2022	GETTING THE RESULTS AND ADDING THEM TO THE REPORT.	Fésüs Péter
4.	09/05/2022	DID SOME MODIFICATION ON THE FINAL VERSION OF THE REPORT	Fésüs Péter

The Consultation log is required to be countersigned by either supervisor on a total of 4 occasions of consultation.

The student has complied with the requirements of the subject titled Degree project I / Degree project II (BSc) or Thesis 1 / Thesis 2 / Thesis 3 / Thesis 4[3], and is allowed to proceed for reporting / defense[4].

Dated: Budapest, 09/05/2022

Fésüs Péter
.....
institutional supervisor

[1] Underline as appropriate
[2] Underline as appropriate
[3] Underline as appropriate
[4] Underline as appropriate

TABLE OF CONTENTS

Abstract

Drowsiness or sleepiness while operating a vehicle is a very critical issue. Drowsy drivers cause each yearnearly three thousand car accidents.Crashes due to drivers getting drowsy or sleepy account for around 20% of all accidents and on certain long journey roads it's up to 50%. It is a serious issue and this turns out to be a big problem not only for the driver but also for other people who use the road.

Most people that have driven for long hours at night can relate to the fact that fatigue and slight brief state of unconsciousness can happen to anyone and everyone. Thus drowsiness detection play a vital role in preventing traffic accidents.

Developing various technologies for monitoring and preventing drowsiness while driving is a major trend and challenge in the domain of accidence avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects. Driver inattention might be the result of a lack of alertness when driving due to driver drowsiness and distraction. Driver distraction occurs when an object or event draws a person's attention away from the driving task. Unlike driver distraction, driver drowsiness involves no triggering event but, instead, is characterized by a progressive withdrawal of attention from the road and traffic demands. Both driver drowsiness and distraction, however, might have the same effects, i.e., decreased driving performance, longer reaction time, and an increased risk of crash involvement. By developing an automatic solution for alerting drivers of drowsing, before an accident occurs, this could reduce the number of traffic accidents.

There are numerous technologies for detecting drowsiness, which can be classified into three categories:

The first is about vehicle behavior: This type monitors vehicle characteristics such as speed, exacerbation angle, and position. These techniques may be considered non-intrusive, but they have a number of limitations, such as the type of car, driving conditions, and so on. Furthermore, it necessitates specialized equipment and preparation, which can be costly and inconvenient (not practical).

The second type is the biological indicators: This type monitors biological factors like eye movement, brain waves, and heart rate. EEG sensors can detect eye blink strength, frequency, and brainwave activity while measuring a person's alertness and sleepiness.These techniques are generally characterized by the highest detection accuracy and intrusiveness and in these papers we will go through the theoretical aspects of this method.

The third method is computer vision analysis. This type benefits from the dynamic behavior of the human face and eye because they have a high degree of variability; face detection is a difficult problem in computer vision research, whereas the eyes can be considered a salient and relatively stable feature on the face in comparison to other facial features. These features will be fed into a machine learning classifier, which will determine whether they represent an open or closed eye.

This research proposes a real-time detection approach for driver drowsiness. The proposed approach has two phases: first understanding the physiology of driver drowsiness and the biological indicators using EEG sensors and the second phase is the image processing and machine learning analysis and implementation.

Chapter 1: Introduction to Driver Drowsiness & Fatigue Detection System

1.1 Motivation

1.1.1 Key drowsy driving statistics and Road Accidents:

Driver drowsiness represents an important risk on the roads, as it is one of the main factors leading to accidents or near-missed accidents. This has been proven by many studies that have established links between driver drowsiness and road accidents.

According to data from The Royal Society for the Prevention of Accidents, 20% of serious accidents in the UK only are due to driver extreme exhaustion or weariness resulting from physical or mental activity.

-Drowsy driving accounts for about 100,000 crashes annually on the roadway, 71,000 injuries, and 1,550 fatalities per year (NSC).

- According to the National Highway Traffic Safety Administration (NHTSA), drowsy driving was responsible for 91,000 traffic accidents in 2017, resulting in approximately 50,000 injuries and 800 deaths.

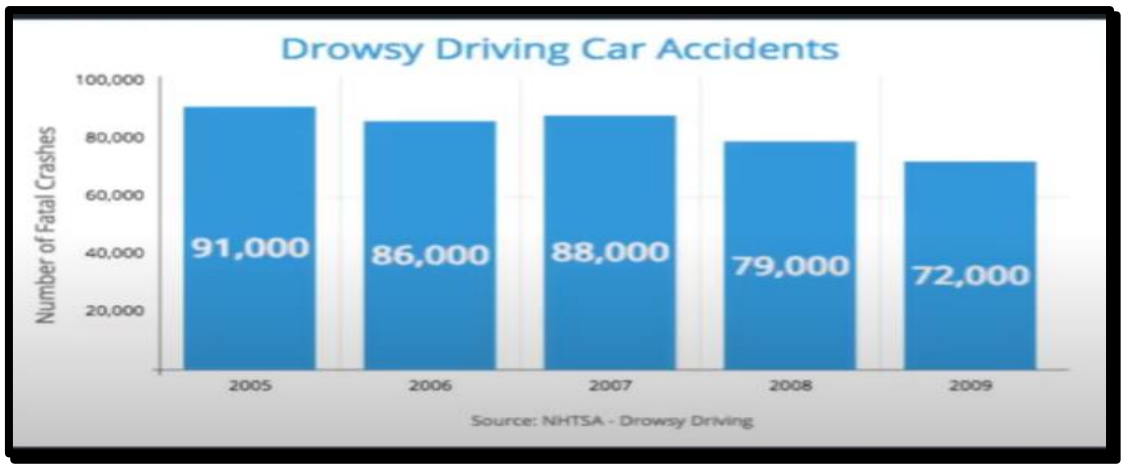


Figure 1:Drowsy Driver car crashes

1.1.2 The Mechanisms of Human Sleepiness

-A body of literature exists on the mechanisms of human sleep and sleepiness that affect driving risks.

The sleep-wake cycle is governed by 2 internal biological mechanisms: circadian rhythm and homeostasis that work together to regulate when you are awake and sleep.

- **Homeostasis** relates to the neurobiological need to sleep, It keeps track of your need for sleep. The homeostatic sleep drive reminds the body to sleep after a certain time and regulates sleep intensity. This sleep drive gets stronger every hour you are awake and causes you to sleep longer and more deeply after a period of sleep deprivation (the

longer the period of wakefulness, the more pressure builds for sleep and the more difficult it is to resist)

- **The circadian pacemaker** is an internal body clock that completes a cycle approximately every 24 hours.Homeostatic factors govern circadian factors to regulate the timing of sleepiness and wakefulness.

1.1.3 Biology of Human Sleepiness and fatigue

-« Fatigue » is generally used in everyday speech to describe a general set of feelings or sensations, including one or more of the following: tiredness, sleepiness, boredom, or physical weariness. However, the term is too imprecise to be useful in scientific research.Sleepiness and fatigue are two distinct conditions.

- **Sleepiness:** refers to the inability to stay awake even in situations in which wakefulness is required, such as at work or behind the wheel of a car.
- **Fatigue:** is a state of overwhelming sustained exhaustion and decreased capacity for physical and mental work that is not relieved by rest.

➔Both conditions can greatly affect your quality of life, productivity, performance and safety, and should be evaluated for underlying causes so that appropriate treatment may be prescribed.

1.2 Aims and Objectives of the Research:

-The development of technologies for detecting and preventing drowsiness in the vehicle is a major challenge in the field of accident avoidance systems. Because of the issues that drowsiness poses on the road, methods for counteracting its effects must be developed.

-The goal of this research is to create a prototype drowsiness detection system that can detect driver drowsiness and warn the driver before driving is impaired. The main focus will be on developing a system that will accurately detect the symptoms of driver drowsiness early enough to avoid a car accident.

1.3 Thesis Structure:

Chapter 1: Introduction to Driver Drowsiness & Fatigue Detection Systems

Chapter 2: Review of The Research on the physiology of driver drowsiness.

Chapter 3: Driver drowsiness detection System Description Based on EEG technology.

Chapter 4: System Description Based on computer vision technique and machine learning.

Chapter 5: Prerequisites.

Chapter 6: Modeling diagrams and system design.

Chapter 7: Results and Discussion.

Chapter 2: Review of The Research on the physiology of driver drowsiness

2.1 Defining Drowsiness:

The phenomena of drowsiness has been widely researched, although no universally accepted definition exists. Drowsiness is a condition marked by a reduced capacity for labor and decreased efficiency of accomplishment, often accompanied by a sense of fatigue and exhaustion. According to this description, the involvement of drowsiness in a road crash might range from falling asleep at the wheel to lack of attention.

-The four main factors of driver drowsiness according to the general consensus are:

- Lack of sleep
- Time of day
- Time spent performing a task
- Type of driver

-We should also note that drowsiness is influenced by a variety of factors, including age, physical fitness, and medical condition.

2.1.1 Lack of Sleep:

-Sleep is essential for everyone. The longer someone remains awake, the more difficult it is to resist falling asleep. Individual sleep requirements vary, however sleeping for 8 out of 24 hours is common, and 7 to 9 hours of sleep is essential to optimise performance.

-Sleeping less than four hours per night impairs performance. Sleepiness reduces reaction time, which is a critical element of safe driving. Lack of sleep lowers attention and focus needed for safe driving. Decision-making quality may also be harmed (affected).

2.1.2 The Time of Day:

-Humans possess a neurobiological based sleep-wake cycle called a circadian rhythm or body clock. According to research, there are two periods within the 24 hour circadian cycle when people are most sleepy. The first occurs at night and early in the morning, and the second occurs in the afternoon. Many functions (such as attentiveness, performance, and subjective mood) are harmed during these times of sleepiness.

-Sleep related accidents have peaks in the early hours of the morning, between 2 :00 am and 6 :00 am, and mid afternoon, between 3.00 pm and 4.00 pm. Drivers are 50 times more likely to fall asleep at the wheel at 2.00 am than at 10.00 am. This risk is three times as great between 3:00 and 4:00 pm as at 10:00 am. Studies identified that young drivers are more likely to sleep at the wheel in the early hours in the morning and older drivers are more likely to fall asleep at the wheel during the afternoon sleep period.

-The long sleep-wake cycles are the result of variable insensitivity to the sleep drive.

-Randomly selected volunteers in one study went 72 hours without sleep and scored their drowsiness level on a scale every three hours in relation to their typical drowsiness (=100%). The feeling of drowsiness was always at its peak in the early morning and at its lowest in the afternoon.

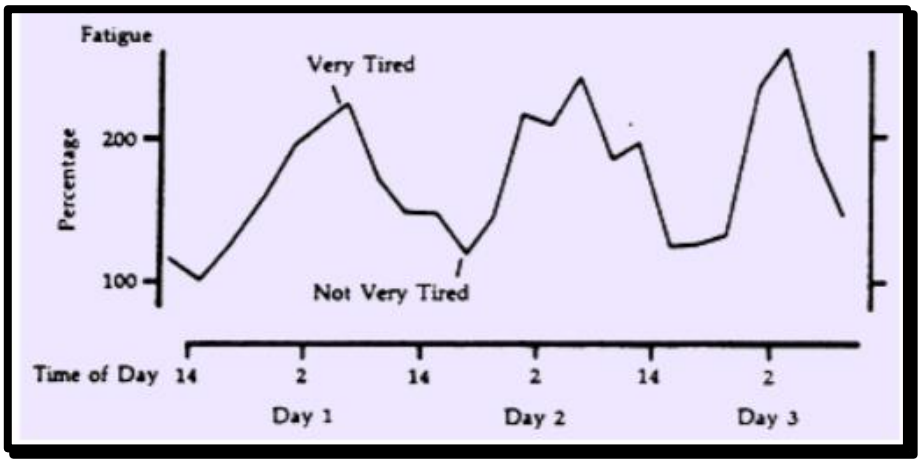


Figure 2: Measurement of fatigue during 72 hours of lack of sleep

-The following diagram (bellow) shows the time required to fall asleep during the day, after long sleep, typical sleep, and a sleepless night. Between 9:30 a.m. and 7:30 p.m., the participants lay down for two hours. Sleep propensity is determined by the amount of time it takes to fall asleep. The participants take longer to fall asleep after an extended sleep night but this time is drastically reduced after a night without sleep.

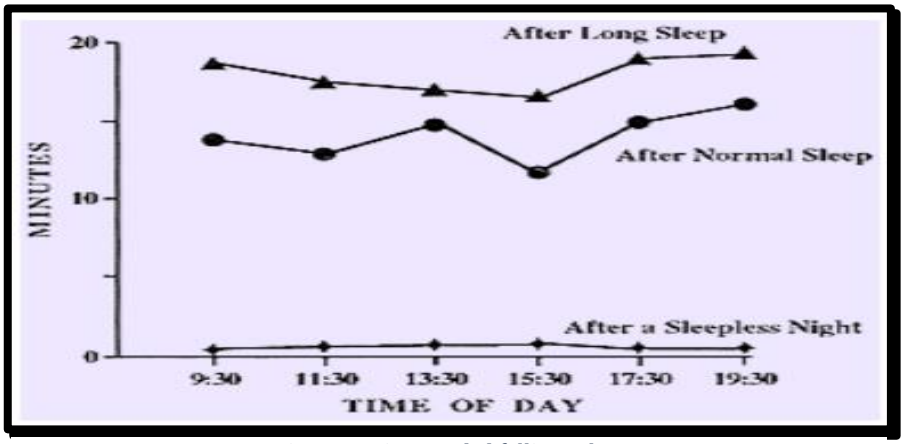


Figure 3: Time needed falling asleep

2.1.3 Time on Task:

Time-on-task can be defined as the amount of time you spend being actively. A continuous mental workload without a break will result in decreased alertness and a disinclination to continue the effort. Similarly, prolonged physical activity without rest leads to muscular drowsiness.

- The duration of time spent on a task influences performance, according to research based on driving activities. The level of drowsiness increases as time spent on a task increases, reaction time slows, vigilance and judgment are diminished, and the likelihood of falling asleep throughout the task increases.

2.1.4 Type of the Driver:

Young male drivers under the age of 30 have been identified as one of the categories most at risk of being involved in sleep-related road accidents in several studies. Furthermore, company vehicle drivers are more likely to fall asleep at the wheel because they frequently drive long distances on tight schedules. In addition, shift workers and persons who have sleep issues are at danger.

The close environment of the inside of a car, as well as the lack of air flow and low oxygen level, all contribute to a greater desire to sleep.

2.2 Physiological Measures Related to Driver Drowsiness Detection:

The main goal of this section is to discover the various measurements that can be used to identify driver drowsiness, as well as their operational definitions.

2.2.1 Eyelid Closure:

-Eyelid closure is a very reliable predictor of driver drowsiness. Plethysmography (a device for measuring and recording changes in the volume of the body or of a body part or organ), respiration rate, Electroencephalography (EEG), skin electrical characteristics, Electromyography (EMG), heart rate variability, and eyelid closure were all investigated to see if they were predictive of sleep onsets. Among the measures used, eyelid closure was found to be the most reliable predictor of the onset of sleep.

-Eyelid closure indicates the onset of sleep and is unquestionably the cause of poor performance in visual tasks, particularly tracking skills like driving. It seems self-evident that a driver's ability to operate a car would be affected if his or her eyelids were closed. Researchers looked at how well sleep-deprived drivers could complete a one-and-a-half-hour driving job. To simulate on-the-road situations, various disturbances were purposefully input into the driving simulator's steering system. Eyelid closures were shown to be highly linked with performance metrics like as lane departure and steering velocity.

2.2.2 Brain Wave Activity

-Researchers found that there is no reliable alteration in background brain activity prior to eyelid closure. Upon eyelid closure, the researchers found that a very rapid shift in brain wave patterns takes place. This shift is identifiable as the early stage of sleep.They discovered a sharp changes in the frequency content of brain wave activity are observed during the crossing from alertness to a stage of hypoalertness, then to drowsiness, and finally to sleep. A slowdown of the brain activity in general, an increase in the percentage of alpha waves and, in turn, a decrease in the percentage of beta waves, is observed at the same time that a decline in performance is seen.

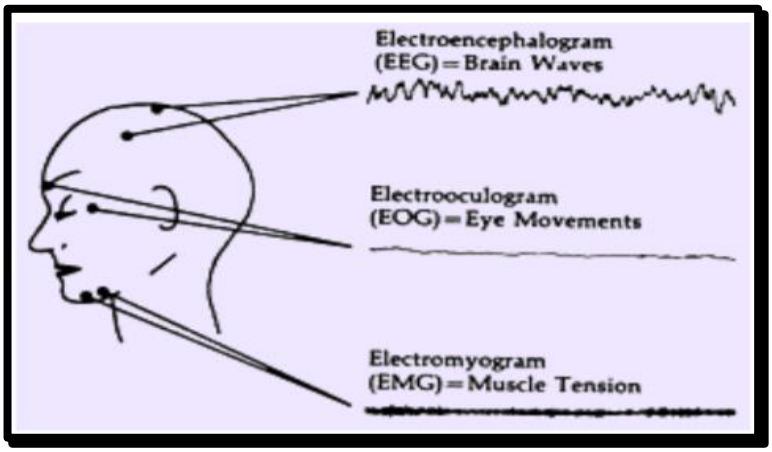


Figure 4: Positions of Physiological Measures

Chapter 3: System Description Based on EEG Technology

Introduction:

In this chapter we will explore the function of the brain, the relationship between neurons in the brain, brainwaves, and other brain activity patterns. Then we go through EEG technology and how to use an EEG for a Brain Computer Interface. In the next sections of this chapter, the sensor employed, its specifications, functioning technique, and data gathering operations employing this sensor will be discussed.

3.1 The function of the brain:

The brain is the control center of the human body and it is a complex organ that it manages everything we do .It controls every process that regulates our body.

Whether you're thinking, dreaming, playing sports, or even sleeping or breathing, the brain is involved in one way or another.

-from an anatomical perspective, the brain can be divided into three different parts: cerebrum, brain stem and cerebellum

*** The cerebrum**: contains the convoluted surface with left and right hemisphere; this is called cerebral cortex. It initiates and coordinates movement and regulates temperature. Other areas of the cerebrum enable speech, judgment, thinking and reasoning, problem-solving, emotions and learning. Other functions relate to vision, hearing, touch and other senses.

*** The cerebellum:** It plays an important role in the Pavlovian learning. Essential in the control of movements balance and in cognitive processes that require coordination.

*** The brain stem:** plays an essential role in cardiac regulation and respiration. It modulates the frequency and strength of the heart's contractions. It also modulates blood pressure by impacting the diameter of blood vessels and controls hormones.

- EEG primarily depends on cerebral cortex as highest influence comes from the surface position.

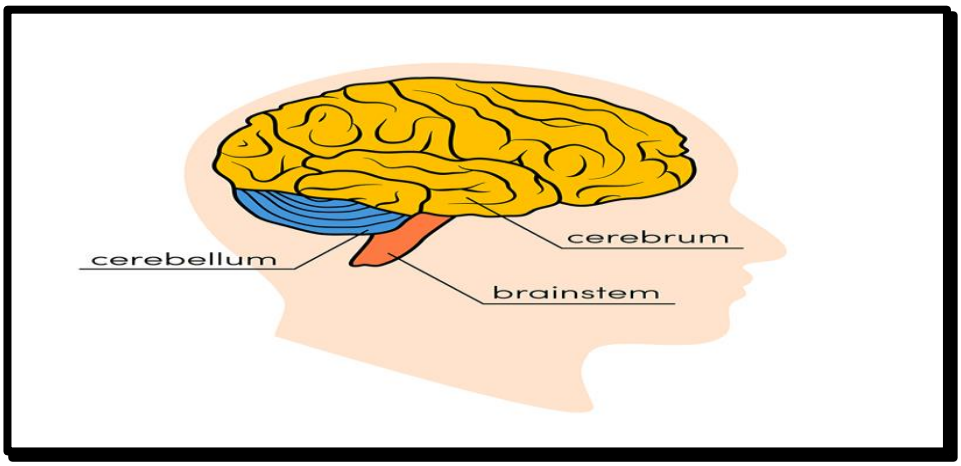


Figure 5: Major parts of human brain

3.2 Electroencephalogram (EEG):

3.2.1 Origin of the EEG signal:

-An electroencephalogram (EEG) is a method that detects electrical activity generated by the nerve cells of the brain, mainly the cortical activity, using small, metal discs (electrodes) attached to the scalp. The brain cells communicate via electrical impulses and are active all the time, even when you're asleep. This activity shows up as wavy lines on an EEG recording.

-The main origin of the EEG is the neuronal activity in the cerebral cortex, but some activity also originates from the thalamus and from subcortical parts of the brain. The EEG represents the summation of excitatory and inhibitory postsynaptic potentials in the nerve cells. The rhythmic activity is due to the synchronous activation of the nerve cells.

-The signal is classified on the basis of its amplitude and frequency range. The recorded pattern differs during the different sleep stages, but also when performing cognitive tasks, focusing attention, preparing manual tasks or by brain diseases, for example epilepsy or tumours. An EEG is one of the main diagnostic tests for epilepsy. An EEG can also play a role in diagnosing other brain disorders. It might also be helpful for diagnosing or treating the following disorders:

- ✓ Brain tumor
- ✓ Brain damage from head injury
- ✓ Brain dysfunction that can have a variety of causes (encephalopathy)
- ✓ Inflammation of the brain (encephalitis)
- ✓ Stroke
- ✓ Sleep disorders

3.2.2 Classification of EEG:

-Brain waves are oscillating electrical voltages in the brain measuring just a few millionths of a volt. The waves are then derived using the Fourier transform on the raw EEG signal. Though the 13spectrum is continuous, the brain state of certain individual may make certain frequencies more dominant. The amplitude of the EEG is ~ 100 μV when measured on the scalp, and about 1-2 mV when measured on the surface of the brain. The bandwidth of this signal is from under 0 Hz to about 50 Hz.

There are five widely recognized brain waves, and the following table shows the main frequency bands and their relation with one another.

Brainwave	**Frequency (Hz)**	**Characteristics**
Gamma	30-100 Hz	-Strong focusing and concentration -It is related with cognitive functioning, learning new material, memory and information processing. - Beta and gamma waves together have been linked with attention and perception
Beta	13 Hz –30 Hz.	-These waves are small, fast and connected with focused concentration. They are associated with logical thinking, attention, conscious thoughts and stimulating effect. - Various stimulants like caffeine raise the level of beta waves. Too much release of Beta waves relate with anxiety or stress
Alpha	8-13 Hz	-Relaxation. -Thoughts about something peacful. -In the occipital and frontal parts. - Fewer produce of alpha shows anxiety or high stress whereas too much alpha measured as inability to focus or being too relaxed.

Theta	4-8 Hz	-Subconcious waves. -Paradoxical sleep phase state, deep meditationn, during the peak moments -Creativity and spirtuality.
Delta	0.5-4 HZ	-Unconcious waves. -Deep sleep without dreams. -In combination with other waves is typical for intuitive concentraion (such as radar).

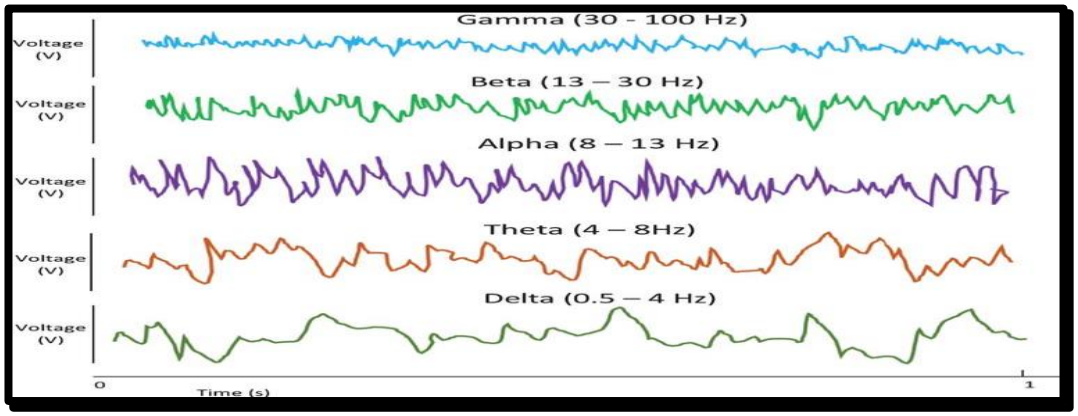


Figure 6: EEG waves

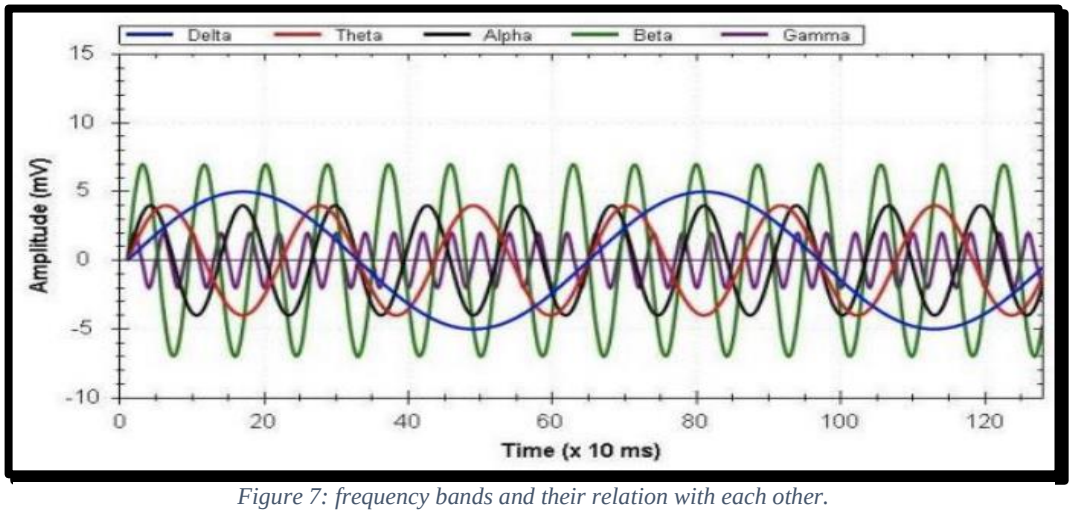


Figure 7: frequency bands and their relation with each other.

3.3 Brain computer interface (BCI) and EEG:

3.3.1 Definition of BCI:

-A BCI (brain-computer interface) is a type of technology that allows the brain to communicate with an external device by sending and receiving signals. Brain-computer interfaces, also known as brain-machine interfaces, are a type of brain-computer interaction. BCIs gather and interpret brain impulses before sending them to a linked machine, which responds with orders based on the information received. A simplified BCI definition might describe the technology as "a direct communication link between the brain and an external device." This connection is a two-way link (a bidirectional interface). One direction involves a BCI sending brain activity to a computer, and the computer translating brain activity into motor commands. Communication can also take a place in the opposite direction: where the computer sends information directly to the BCI user's brain. This is called active BCI where there is a direct brain connection, compared to passive BCI which is non-invasive.

-Recently, this technology is being ventured for not only clinical field but also towards healthy people measuring their cognitive state. So, according to the current definition of BCI, it is a system that receives a biological signal from a person's brain and produces an abstract representation of that person's cognitive state.

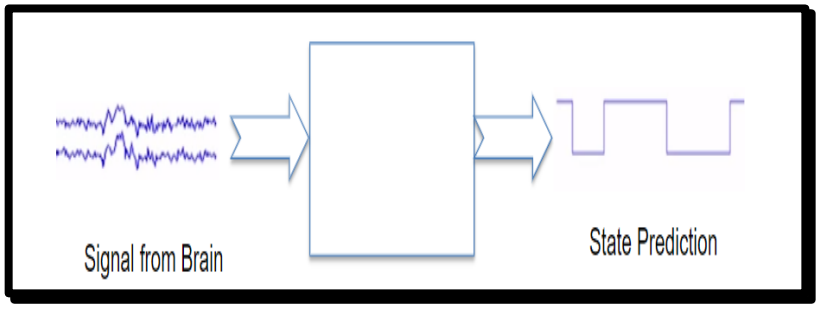


Figure 8: BCI or Brain Computer Interface

3.3.2 BCI types:

BCI is divided into three categories:

- ✓ **Active BCI:** derives its outputs from brain activity which is consciously controlled by the user, independently from the external events, for controlling an application.
- ✓ **Reactive BCI:** obtains its outputs from the brain activity that occurs in response to external stimulation and it is adjusted indirectly by the user for application control.
- ✓ **Passive BCI:** derives its outputs from arbitrary brain activity without the goal of voluntary control, in order to enrich a human-computer interaction with implicit data.

3.3.3 Measurement of EEG:

-EEG can be measured using multiple electrodes placed on the human scalp. For multichannel montages, electrode caps are preferred, with number of electrodes installed on its surface. The 10-20 international system is the standard naming and positioning scheme for EEG applications. The name indicates that the electrodes are placed at positions 10 % and 20 % of the distance between four anatomical landmarks. The landmarks are the nasion (bridge of nose), the inion (projection of bone at the back of the head) and the left and right preauricular points (depressions in front of the ears).

The points are labelled with a letter and a subscript index. The letters refer to the regions of the brain; F = frontal, O = occipital, C = central, P = parietal and T = temporal. The subscript indices are z which indicates the midline and numbers indicating the lateral placement and degree of displacement from the midline. An odd number refers to the left hemisphere, an even to the right. The number gets 13 higher the farther away it is from the midline.The 10-20 international system is based on an iterative subdivision of arcs on the scalp startingfrom craniometric reference points: Nasion (Ns), Inion (In), Left (PAL) and Right (PAR) pre-auricular points. The intersection of the longitudinal (Ns-In) and lateral (PAL-PAR) is named the Vertex. The original 10-20 system included only 19 electrodes. Later on, extensions were proposed so that now you can place over 70 electrodes in standard positions.

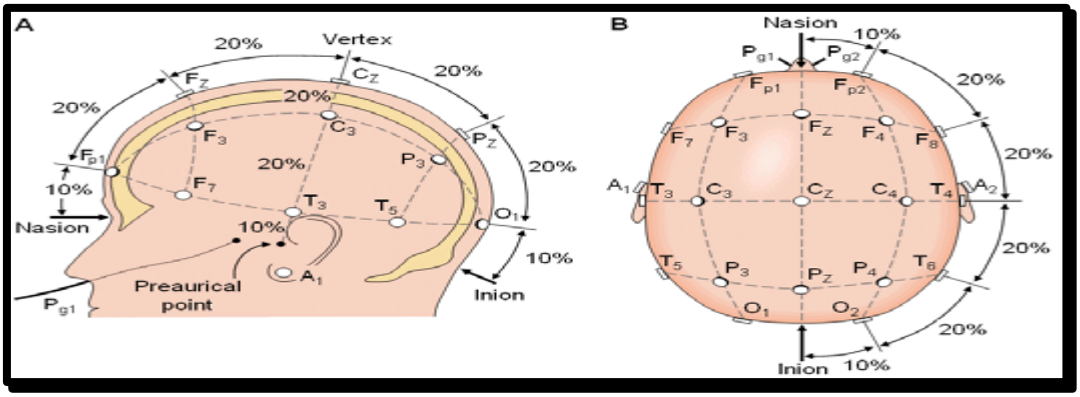


Figure 9: Labels for points according to 10-20 electrode placement system

3.3.4 Neurosky Mindwave Sensor

NeuroSky is one of the leaders in biosensor innovation with a goal of making biosensors affordable to the general public. Their goal is to power thousands of products across the health and wellness sector around the globe. EEG activity can be measured with dry biosensors from Neurosky. This is a single-channel portable research sensor that amplifies and analyses raw brain signals. It features a versatile API that allows us to retrieve raw EEG and brainwave band signals.

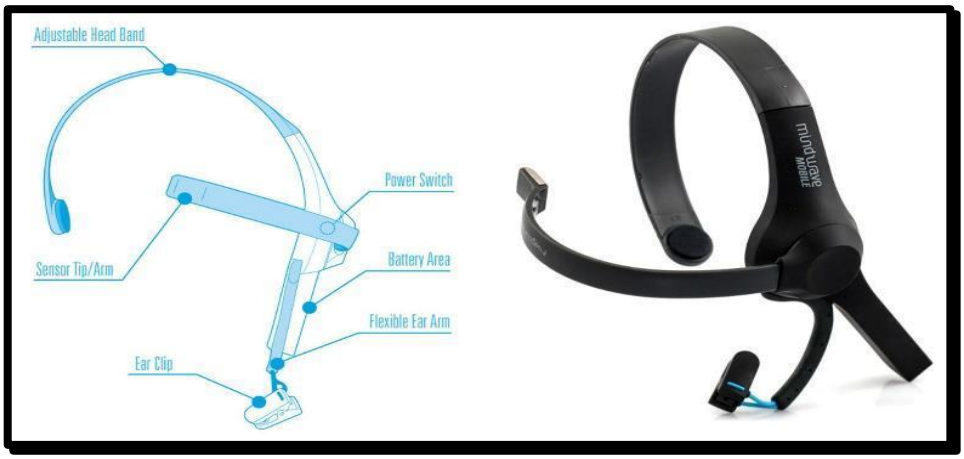


Figure 10: NeuroSky Biosensor

- The NeuroSky Mindwave module is extremely basic, consisting of only a headset, an ear clip, and a sensor arm. The headset and ground electrode references are located on the ear-clip. The EEG electrode is attached to the sensor arm and placed on the forehead above the eye. Neurosky sensor placement is aimed at FP1. This area of the forehead has very little hair because it produces EEG clarity, allowing for the near-accurate delivery of raw EEG. This is an excellent location for gathering the cognitive process described above, which includes attention and relaxation level. Another critical aspect of using the sensor on FP1 is its close proximity to the eye. It allows the sensor to detect eye blinks and then the strength of the blinks. This is one of the primary reasons for using Neurosky to carry out this experiment.

- EEG signals are extremely weak and easily contaminated by outside sources. An artifact is EEG that was not generated by the brain. Body movements, heartbeats, muscle contractions, tongue movements, and even sweating can all result in physiologic and non-physiologic artifacts. The sensor in Neurosky detects ambient noise or artifacts from the body or other devices. An ear-clip monitor is included with the sensor. It serves as a ground reference, allowing Neurosky's chip to filter out all of the electrical noise from the body and the surrounding environment and focus more on brainwaves.

-There have been several benchmark tests of the dry EEG by comparing EEG signals measured by the dry sensor system with signals from the medical grade EEG system.The NeuroSky and Biopac systems recorded EEG at the same time. The Biopac system is a well-known wet electrode EEG system that is extensively used in medical and research applications. The electrodes for the two systems were positioned in the same location, as close together as possible without interfering with one another. The NeuroSky system used dry electrodes, whereas the Biopac system used silver-silver-chloride disposable electrodes with gel. EEG was recorded for a variety of conditions, including the subject relaxing and meditating, alert and attentive, and during eye blink artifacts.

Conclusion:

A system that can detect drowsiness and deliver appropriate alerts is a viable countermeasure meant to lower the occurrence of driver drowsiness-related crashes. The efficiency and

dependability of this system in alerting the driver to the issue in a timely manner is the most critical factors.

- In sleep laboratories, a variety of technologies have been applied to track sleep and wakefulness in volunteer participants. Electrooculogram (EOG), electromyogram (EMG) and Monitoring the electroencephalogram (EEG) are examples of these methods. However, because electrode attachments are required, these approaches are not flexible to use in practical situation for routine driver monitoring. However, when EEG technology have been used for research in drivers, it has succeeded to reliably detect drowsiness.

Chapter 4: System Description based on Computer Vision technique and machine learning

Introduction:

Fatigue and drowsiness cause noticeable changes in the driver's facial features and expressions, as well as the position of the head and eyes. The majority of studies on the effects of fatigue and sleepiness have focused on the dynamic changes in the eyes and their movements during periods of fatigue and sleepiness. The level of drowsiness among drivers was detected in this study using an image-processing technique and machine learning training.

4.1 Image processing:

4.1.1 Definition:

Image processing is the process of converting an image to a digital format and performing various functions on it in order to obtain a better image or extract other useful information from it. It is a type of signal time when the input is an image, such as a video frame or image and output can be an image or features associated with that image. Typically, the Image Processing system includes the treatment of images as two equal symbols while employing the set methods.

4.1.2 Steps in Image Processing

- The steps of image processing are as the following:

- **Image Acquisition:** This is the first step in image processing. To create specific images, such as a real or real situation internal arrangement of an object, digital image detection is used. This term is commonly expected to accept processing, congestion, storage, printing, and display of such images. Image acquisition may humble as considering the pre-existing image digital form.
- **Image Enhancement:** is the process of converting digital images into more visually appealing (suitable) results for display or multiple image analysis. Because you can, for example, turn off sound, sharpen, or turn on image, making it easier to identify key features.
- **Image Restoration** is the process of taking an unethical / noisy image and measuring it against an unused, new image. Exploitation can take many forms, including action blurring, sound, and camera focus. The goal of image restoration techniques is to reduce noise and reclaim decision loss.
- **Coloring** : Image Processing: Coloring Image Processing necessitates an understanding of the physics of light as well as color vision physiology. The color of human use details of material classification, building materials, food, places, and time of day Color is used in the separation image processing process.

- **Wavelets processing and Multiple Solutions:** When you decorate a photo with atmosphere, clouds, trees, and flowers, you will use a different level brush based on the size of the topographies. Wavelets are similar to brushes. The wavelet transform is a useful tool for image representation. The wavelet transform enables the investigation of multiple image solutions.

- **Image compression:** Image compression is a type of data compression that is used in digital photography to reduce the costs of storage and distribution. Processes can reap (earn) visual benefits awareness and asset data image assets to complex effects related to normal pressure strategies.

- **Character recognition:** Optical character binding, accessed as OCR, is a machine-operated or electronic substitute for scanned photos, typed text, or any other computer-readable text. It's frequently used as a presence to access records once the new type of data source is introduced, whether it's a paper invoice, bank statement, income, business cards, quantity of printed records, or email. It's a standard approach to create digital printed manuscripts that can be electronically sorted, searched, and saved in machine processes like machine conversion and online display, text-to-speech, key data retrieval, and text mining. OCR is the result of a slew of studies on intelligence, pattern recognition, and computer concepts.

4.2 Computer Vision:

4.2.1 Definition:

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs and take actions or make recommendations based on that information. Itcomes from modelling image processing using the techniques of machine learning. Computer vision applies machine learning to recognise patterns for interpretation of images. Computer vision works much the same as human vision. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image. Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyze thousands of products or processes a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities. Computer vision, like image processing, takes images as input and gives output in the form of information on size, colour intensity etc.

4.2.2 Steps of Computer Vision:

- **Aquiring an image :** Images and large sets can be aquired in real time through photos, video or 3D technology for analysis

- **Processing the image :** Deep learning models automate much of this process, but the models are often trained by first being fed a thousand of pre-identified or labeled images

- **<u>Understanding the image:</u>** this is the final step and it's the interpretative step, where an object is identified or classified.

<u>4.2.3 How does computer vision work?</u>

- Computer vision technology tends to mimic the way the human brain works. However, how does our brain deal with visual object recognition? One popular theory holds that our brains use patterns to decode individual objects. This concept is used in the development of computer vision systems.

-Computer vision algorithms that we use today are based on pattern recognition. We train computers on a massive amount of visual data, computers process images, label objects on them, and find patterns in those objects.

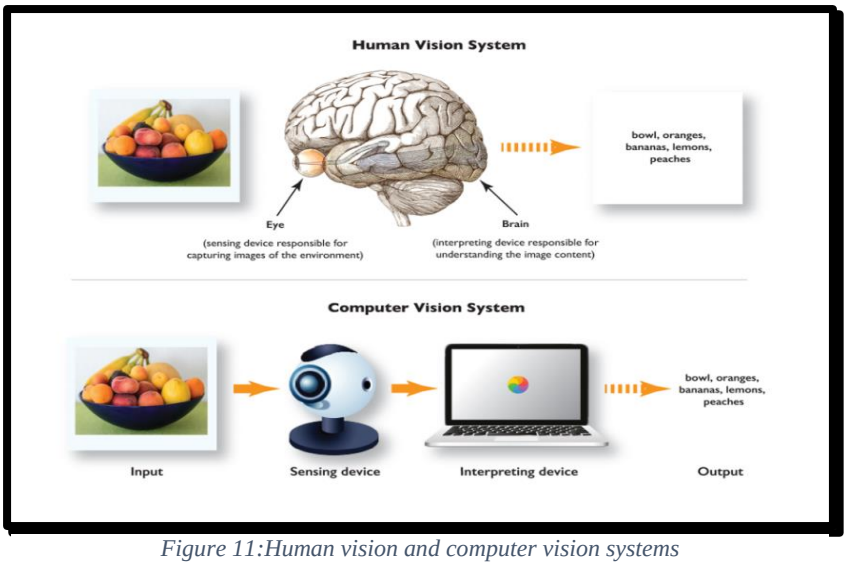


Figure 11:Human vision and computer vision systems

While the previous steps mapping out the fundamentals of computer vision appear simple, processing and analyzing an image using machine vision is rather challenging.

-A pixel is the smallest quanta in which an image may be divided, and a picture is made up of multiple pixels.Computers use an array of pixels to process images, with each pixel having a set of values that reflect the presence and intensity of the three primary colors: red, green, and blue. A digital image is formed when all pixels are combined.As a result, the digital image is transformed into a matrix, and Computer Vision is transformed into a study of matrices. While the most basic computer vision algorithms modify these matrices using linear algebra, more complicated applications use operations such as convolutions with learnable kernels and downsampling via pooling.

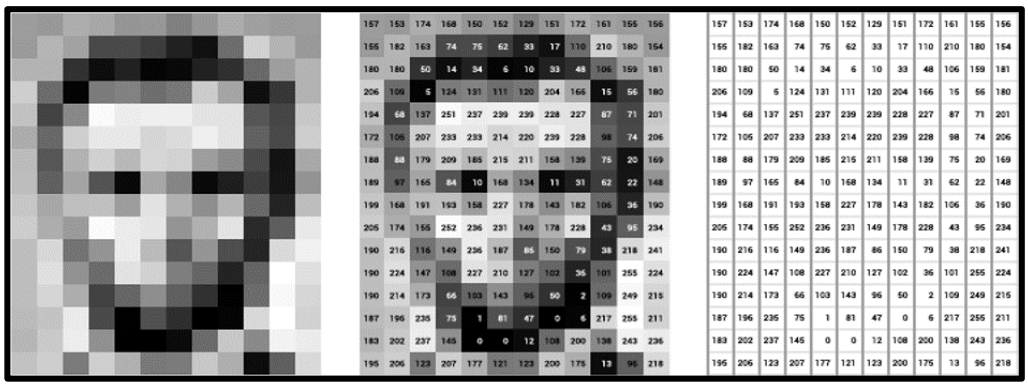


Figure 12: Color values of individual pixels are converted into a simple array of numbers used as input for a computer vision algorithm.

-Two essential technologies are used to accomplish this: machine learning and a convolutional neural network (CNN).

4.3 Machine Learning:

4.3.1 Definiftion:

Machine learning: is a subfield of artificial intelligence (AI) that enables systems to learn from massive volumes of data and solve specific problems. It uses computer algorithms that improve their efficiency automatically through experience. If enough data is supplied into the model, the computer will "look" at the data and learn to distinguish between images.

4.3.2 The Different *Types of Machine Learning:*

- Generally, tasks in machine learning are classified into broad categories. These categories are based on the manner in which learning is received or on the manner in which feedback on learning is provided to the system's development.

-Due to the complexity of machine learning, it has been divided into two primary areas: supervised and unsupervised learning. Each has a distinct purpose and action, generating results and utilizing a variety of data types. Around 70% of machine learning is supervised learning, while 10% to 20% is unsupervised learning.

- ❖ **supervised learning**

Overview:

-The most common type of machine learning is supervised learning, which is used by the majority of machine learning algorithms. This is also referred to as inductive learning.

Supervised learning involves providing the computer with example inputs labeled with their desired outputs. The purpose of this method is for the algorithm to "learn" by comparing its actual output to the "taught" outputs in order to identify and correct errors in the model. Once the model is trained based on the known data, you can use unknown data into the model and get a new response.supervised learning makes use of patterns to forecast the label values of additional unlabeled data.

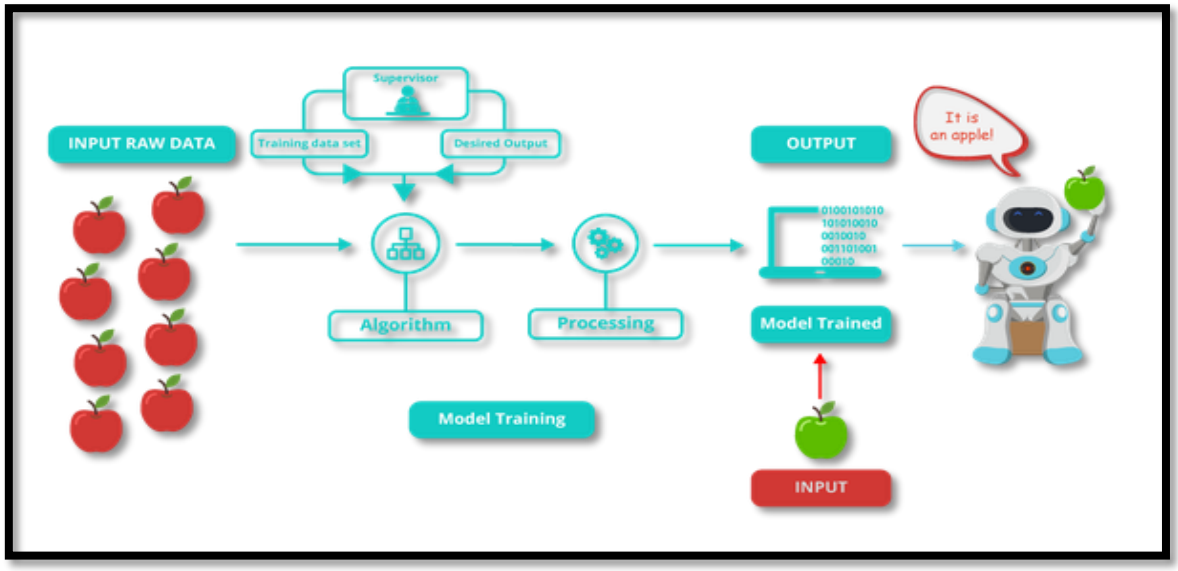


Figure 13: Supervised Learning model

Algorithms: The main types of supervised learning algorithms include:

- Classification algorithms: These algorithms construct predictive models from training data that includes features and labels for classes. These predictive models, in turn, predict class labels using the features learned from training data on new, previously unseen data. The output classes are discrete. Classification algorithms come in a variety of forms, including decision trees, random forests, and support vector machines.
- Regression algorithms: These algorithms are used to forecast output values based on data-derived features. To accomplish this, the algorithm creates a model from the training data's features and output values, and then uses this model to predict values for new data. In this case, the output values are continuous rather than discrete. Among the numerous types of regression algorithms are linear regression, multivariate regression, regression trees, and lasso regression.

Applications:

-Generally supervised learning algorithms are used to solve classification and regression problems.

supervised learning is used in a variety of applications, including speech recognition, credit scoring, medical imaging, weather forecasting, sales forecasting, and stock price analysis and search engines.

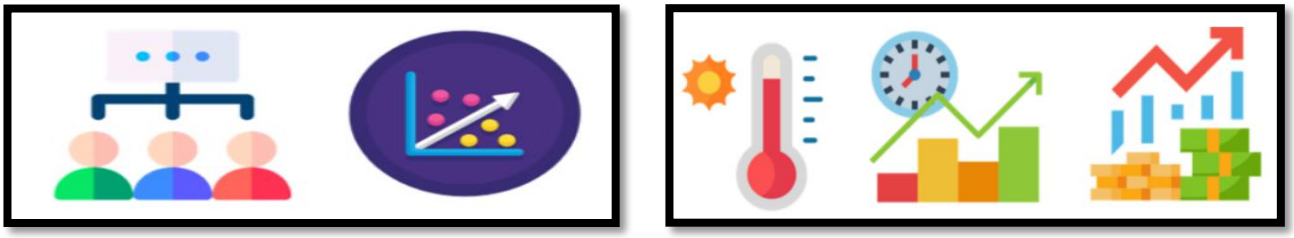


Figure 14: Supervised learning Applications

- **Unsupervised learning :**

Overview:

Unsupervised learning is used when the training data is unknown and unlabeled - that is, no one has previously examined the data. Without the aspect of known data, the input cannot be guided to the algorithm, resulting in the unsupervised term. This data is used to train the machine learning algorithm. The trained model makes an attempt to find a pattern and respond appropriately. In this case, it frequently appears as though the algorithm is attempting to crack code in the same way that the Enigma machine did, but without the involvement of a human mind, but rather a machine. Unsupervised learning may have the simple goal of discovering hidden patterns within a dataset, or it may have the goal of feature learning, which enables the computational machine to discover the representations required to classify raw data automatically.

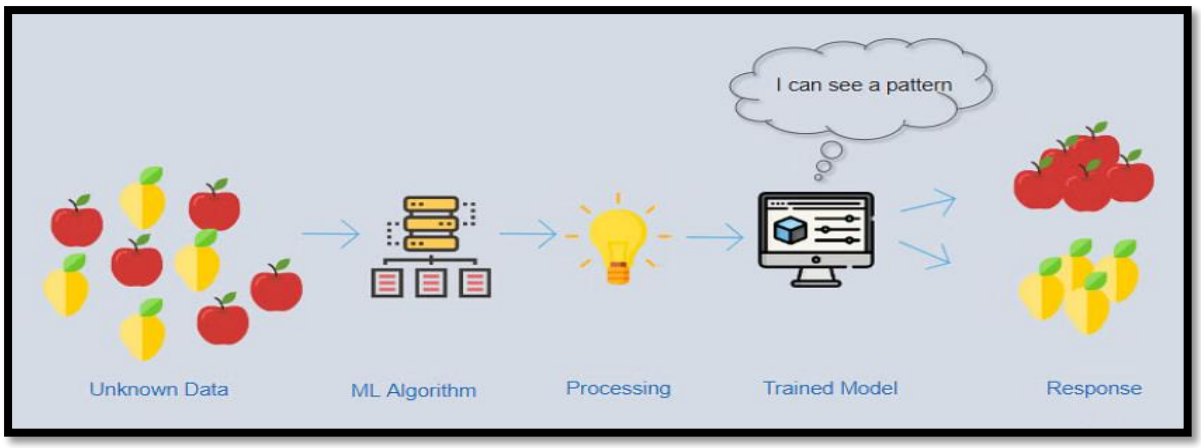


Figure 15: Unsupervised Learning Model

-In this instance, the unknown data consists of apples and pears that appear to be identical. The trained model attempts to group them all together in such a way that similar items appear in similar groups.

Algorithms: The top seven algorithms used for unsupervised learning at the moment are as follows:

- Partial least squares
- Fuzzy means

- Singular value decomposition
- K-means clustering
- Apriori
- Hierarchical clustering
- Principal component analysis

Applications:

- Unsupervised learning is used to solve problems involving clustering and association.
- Customer segmentation is one of the applications of unsupervised learning. You can segment and cluster similar customers based on their behavior, likes, dislikes, and interests. Churn rate analysis is another application example of unsupervised learning algorithms.

Figure 16: Unsupervised learning Application

4.3.3 Machine Learning Steps

Figure 17: Machine learning steps

The task of imparting intelligence to machines appears daunting and impossible. But it is actually really easy. It can be broken down into 7 major steps:

- Data Collection:

As we know machines initially learn from the data we provide. It is critical to collect reliable data so that machine learning model can identify the correct patterns. The accuracy of your model is determined by the quality of the data feeded to the machine. If the data is incorrect or

out of date, we will get incorrect results or predictions that are irrelevant. So ascertain from the data source and using data from a reputable source is extremly important, as this will have a direct impact on the outcome of the model. Good data is relevant, contains a small number of duplicated and missing values, and accurately represents the various subcategories/classes present.

❖ Data Preparation:

After getting the data we have to prepare it. This can be accomplished by:

- Combining all of the data and randomizing it. This ensures that data is distributed evenly and that the ordering has no effect on the learning process.
- Cleaning data to eliminate duplicate values, unwanted data, missing values, rows, and columns, and duplicate values, as well as data type conversion. After that restructure the dataset, modifying the rows and columns or their indexes.
- Visualize the data in order to comprehend its structure and the relationships between the various variables and classes.
- Creating two sets from the cleaned data: a training set and a testing set. The training set is the data set from which the model learns. A testing set is used to determine the model's accuracy after the training.

❖ Choosing a Model:

This step involves selecting the appropriate machine learning model to solve the problems, as well as the features that will be included in the models. A machine learning model is used to predict the output of a machine learning algorithm when it is applied to a collected data. It is critical to select a model that is appropriate for the task at hand. Over time, scientists and engineers developed a variety of models for a variety of tasks, including speech recognition, image recognition, and prediction. Apart from that, we must determine whether your model is suitable for numerical or categorical data and make the appropriate choice.

❖ Model Training:

The most critical step in machine learning is model training. During training, we feed prepared data to the machine learning model, which analyzes it for patterns and predictions. As a result, the model learns from the data in order to complete the task set. With time and training, the model improves at predicting.

❖ Evaluating the Model:

It’s quite important to evaluate the model after training it. This is accomplished by evaluating the model's performance on previously unseen data. The unseen data is the testing set into which previously divided our data. If testing is conducted on the same data as training, you will not obtain an accurate measure, as the model is already familiar with the data and will detect the same patterns as it did previously. This will result in disproportionately high accuracy. When

applied to testing data, we can obtain an accurate estimate of the model's performance and speed.

- ❖ Parameter Tuning:

After creating and evaluating thr model, determine whether its accuracy can be improved in any way. This is accomplished by tuning the parameters present in the model. Parameters are the variables in the model that are generally determined by the programmer. The accuracy will be maximum for a particular value of the parameter. The term "parameter tuning" refers to the process of determining these values.

- ❖ Making Predictions

Finally, the model can be used on unseen data to make more precise predictions a better approximation of how the model will perform in the real world

4.4 Convolutional Neural Networks (CNN).

4.4.1 Artificial Neural Networks overview:

An artificial neural network (ANN) is a type of computing system that is used to simulate how the human brain analyzes and processes data. It is the bedrock of artificial intelligence (AI) and is capable of resolving problems that would be considered impossible or difficult to solve by human or statistical standards.Artificial Neural Networks are primarily designed to mimic and simulate the human brain's function. ANNs are constructed to replicate biological neurons using a mathematical structure.

Because neural networks can adapt to changing input, they can generate the best possible result without requiring the output criteria to be redesigned. The neural network concept, which originated in the field of artificial intelligence, is rapidly gaining popularity in the development of trading systems.

The human brain has a decision-making process: it sees or gets exposed to information via the five sense organs; this information is stored, correlated with previous learnings, and appropriate decisions are made.

The ANN concept is similar to that of a natural neural network. The goal of ANN is to teach machines or systems to understand and mimic how the human brain makes decisions and then acts. Inspired by the human brain, neural networks' fundamentals are connected through neurons or nodes and they are depicted as follows:

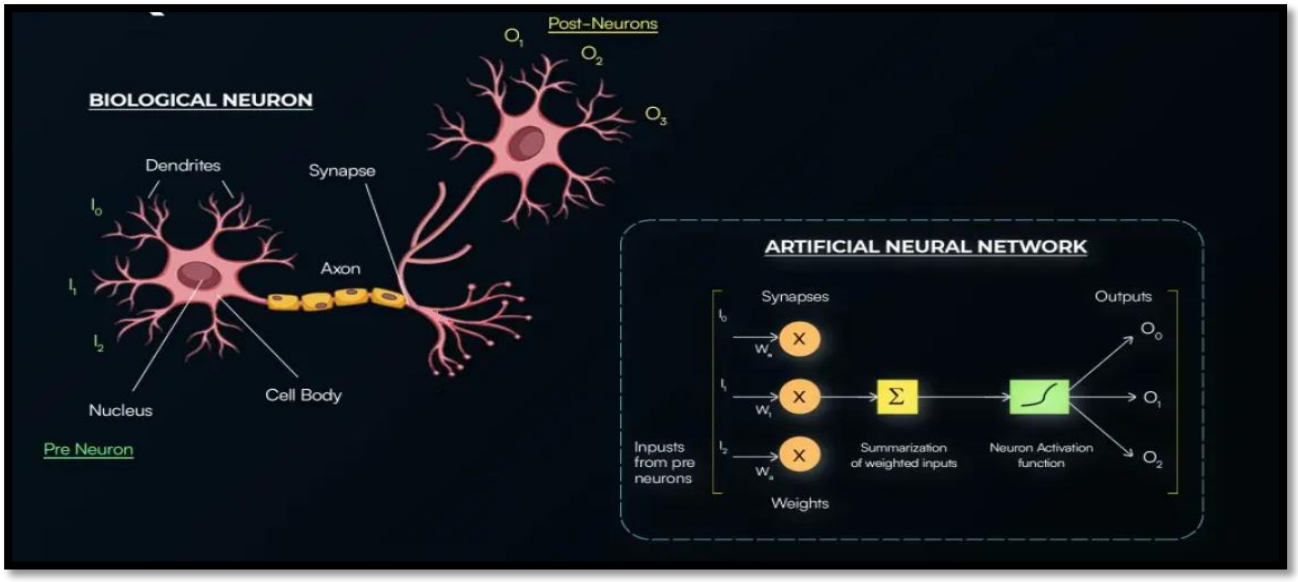


Figure 18: BNN and ANN

The structure of the neural network is determined by the specification of the problem and is configured for the application. Though, on the whole, a neural network has the following structure, and the components of artificial neural networks that comprise the neural network's fundamentals are as follows:

- Layers: Input, Hidden, and Output layers
- Neurons
- Activation Function
- Weights and Bias

Layers

Three layers comprise a neural network: the Input Layer, the Hidden Layers, and the Output Layer.

- ❖ The input layer: is composed of the predictors, which are the inputs or the independent X variables. External sources such as text data, images, audio, or video files are used to collect these inputs. These Xs represent the information perceived by the sense organs in a natural network.
- ❖ The output layer: contains the neural network's output; it may be a numerical value in the case of regression or a binary or multi-layer class in the case of classification. Additionally, the output can be the recognition of handwriting or audio voice or classified text or image in categories.
- ❖ Apart from the Input and Output layers, Neural Networks have a third layer called the Hidden Layer, which is responsible for deriving the model's features. The following image contains 3 hidden layers.

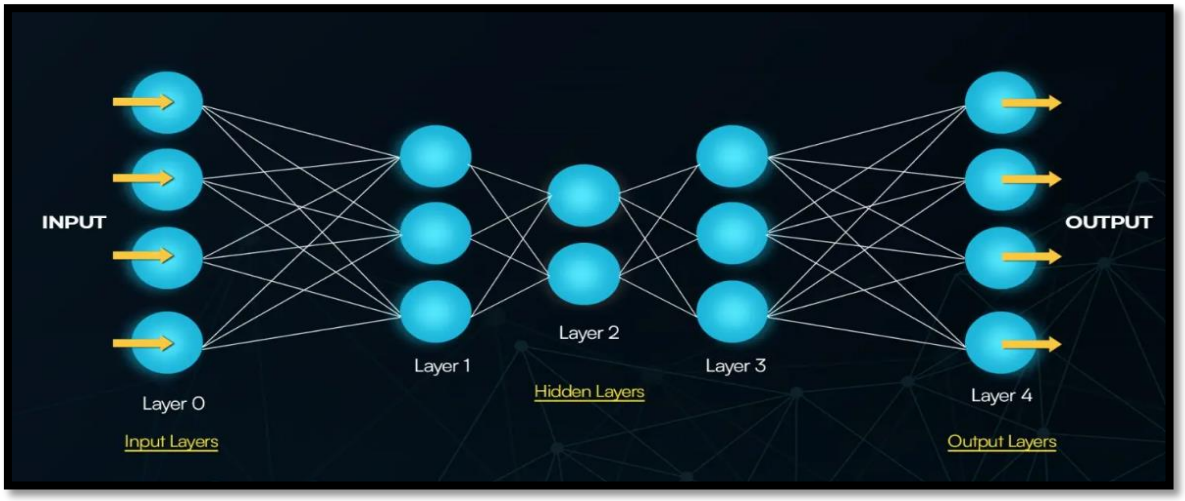


Figure 19: The Neural Network Layers

Neurons:

As previously stated, neurons are the primary and most fundamental processing unit in a neural network. It receives data or information, performs simple calculations, and then forwards it on. Neurons are located in the hidden and output layers. No neurons are present in the input layer; the circles in the input layer represent the independent variables, or Xs.

The output layer's node number is determined by the nature of the business problem. The architecture of the neural network is determined by the user and is configured according to the problem at hand. The input layer is always predefined, and the output layer is also predefined. The number of hidden layers and neurons constitute the hyperparameters. This is because these precise parameters define the features, and a small adjustment to them can have a significant effect on the output.

Weights and Bias:

Each neuron in the first hidden layer is connected to an input from the input layer. Likewise, the neurons of this hidden layer are connected to the neurons of the subsequent layers. Each neuron can have numerous relationships due to its multiple input and output connections. Weights and bias are applied to each of the connections during the nodes' transmission between the layers.

The synapses in a biological neural network indicate the strength of the neurons. Similarly, weights control the strength of the connections between neurons in neural networks artificial intelligence. These weights indicate the neural network's relative importance, indicating how much precedence the input X or the derived neurons will have on the output. This naturally results in the neurons with a higher weight having more influence in the subsequent layer, and the neurons with insignificant weights eventually being dropped out. As a result, the neural network is directed by the weights or connections.

Bias is a separate input to each layer, beginning with the input layer. The preceding layer has no effect on or influence on the bias. In simple terms, bias is the constant intercept term. This means that even if no inputs or independent variables are present, the model will be activated with a default bias value.

➔The model's learnable parameters are the weights and biases. The first iteration, or initialization process, involves randomly setting or assigning weights and optimizing them to minimize loss or error.

4.4.2 Basic Architecture of the work: The Convolutional Neural Networks (CNN):

The model that was constructed on this work is based on Convolutional Neural Networks (CNN).

A convolutional neural network is a special type of deep neural network which performs extremely well for image classification purposes. CNN uses the term 'Convolution' to refer to the mathematical function of convolution, which is a special type of linear operation in which two functions are multiplied to generate a third function that expresses how the shape of one function is modified by the shape of the other. In simple terms, two images that can be represented as matrices are multiplied to produce an output from which features can be extracted.

A CNN is composed basically of three layers: an input layer, an output layer, and a hidden layer that may contain many layers.

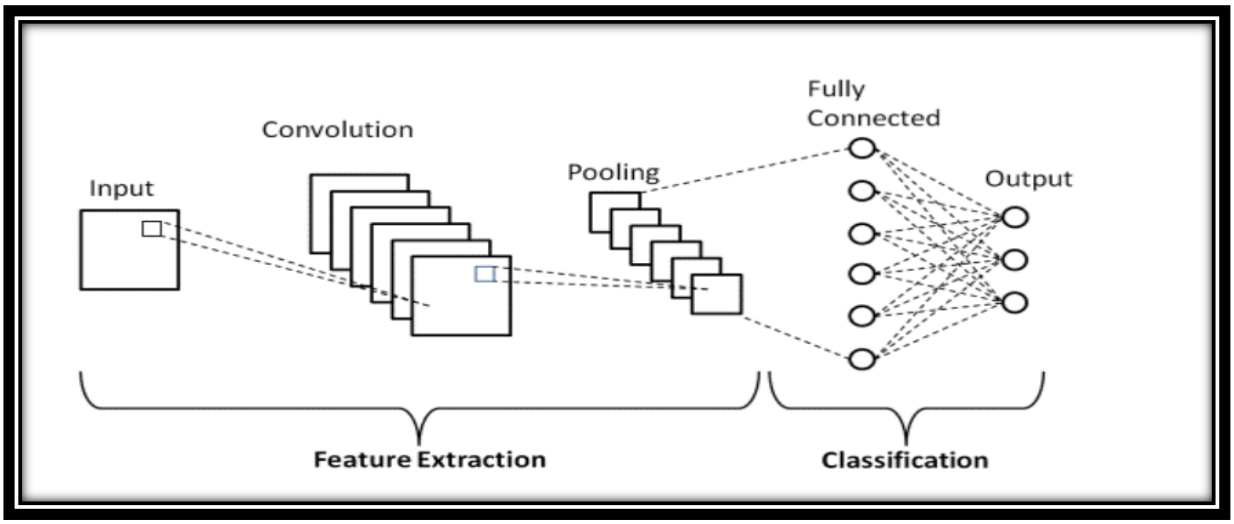


Figure 20: A CNN architecture

- Convolution Layer :

This is the first layer used to extract various features from the input images. This layer performs the mathematical operation of convolution between the input image and a filter of size MxM. By swiping (moving) the filter over the input image, the dot product is calculated between the filter and the parts of the input image that are proportional to the filter's size (MxM).

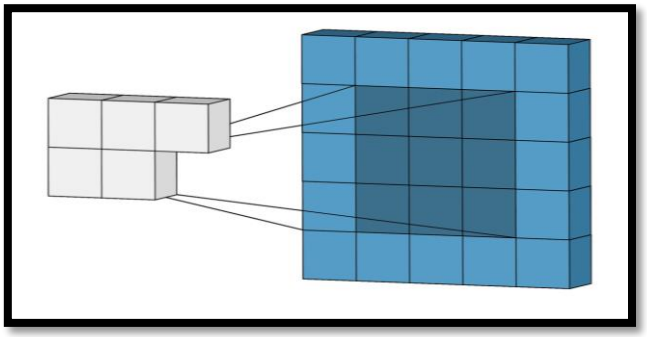


Figure 21: Convolution Operation

« During the forward pass, the kernel slides across the height and width of the image-producing the image representation of that receptive region ». This results in a two-dimensional representation of the image called an activation map, which contains the kernel's reaction at each spatial location in the image. A stride is the name given to the kernel's sliding size.

If we have an input volume of dimension W x W x D and a Dout number of kernels with a spatial size of F, stride S, and padding P, we can determine the size of the output volume using the following formula:

-The kernel is the number of pixels processed together. It is typically expressed as the kernel's dimensions (eg: 2x2, or 3x3).

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

Figure 22: Formula for Convolution Layer

-The stride parameter specifies the number of pixels moved by the analysis window during each iteration. A stride of two indicates that each kernel is two pixels farther from its predecessor.

-Padding is the insertion of (usually) zero-valued pixels to an image's borders. This is done to ensure that border pixels are not discounted (omitted) from the output due to their participation in a single receptive field instance in practice. Padding is normally one less than the kernel dimension. For instance, a convolutional layer composed of three 3x3 kernels would receive a two-pixel padding on all four sides of the image.

-The result is known as the Feature map, and it contains information about the image such as its corners and edges. This feature map is then fed to other layers, which learn several other features from the input image.

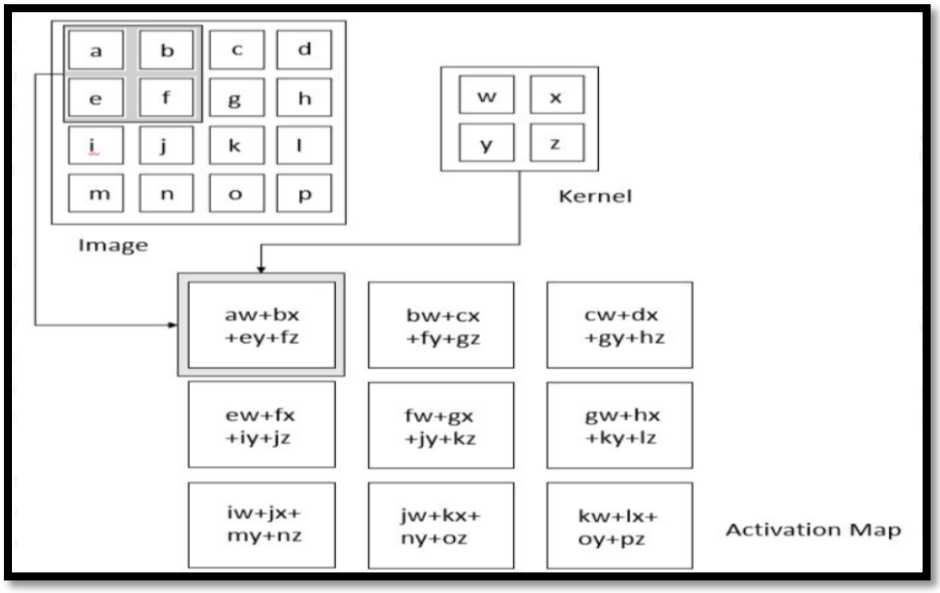


Figure 23: The Feature Map

❖ Pooling Layer

A Convolutional Layer is usually followed by a Pooling Layer. This layer's primary goal is to reduce the size of the convolved feature map in order to reduce computational costs. This is accomplished by reducing the connections between layers and operating independently on each feature map. Pooling operations are classified into several types based on the method used. There are several non-linear functions to implement pooling, where max pooling is the most common. It partitions the input image into a set of rectangles and, for each such sub-region, outputs the maximum.

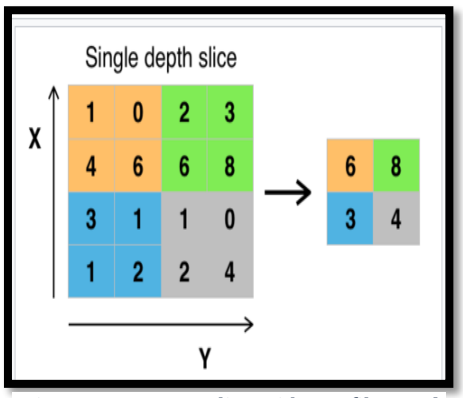


*Figure 24: Max Pooling with 2*2 filter and stride=2*

The largest element from the feature map is used in Max Pooling. Average Pooling computes the average of the elements in a predefined image section size. Sum Pooling computes the total sum of the elements in the predefined section. The Pooling Layer is commonly used as a link between the Convolutional Layer and the FC Layer.

❖ Fully Connected Layer

After series of convolutional and maximum pooling layers, the final classification is performed using fully connected layers. The FC layer (Fully Connected) consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular (non-convolutional) artificial neural networks. Thus, their activations may be

determined using an affine translation followed by matrix multiplication and a bias offset (vector addition of a learned or fixed bias term). These layers are typically placed prior to the output layer and constitute the final few layers of a CNN Architecture.

The flattened vector is then passed through a few more FC layers, where the mathematical function operations are typically performed. At this point, the classification procedure is initiated.

- ❖ Dropout layer:

Typically, when all features are connected to the FC layer, the training dataset becomes overfit. Overfitting happens when a model performs so well on training data that it has a negative affect on the model's performance when applied to new data.

To address this issue, a dropout layer is used, in which a few neurons are removed from the neural network during training, resulting in a smaller model. After passing a dropout of 0.3, 30% of the nodes in the neural network are randomly removed.

- ❖ Activation Functions

Finally, the activation function is a key property of the CNN model. They are used to discover and approximate any continuous and complex relationship between network variables. In simple terms, it determines which information from the model is to be fired forward and which is to be held back.

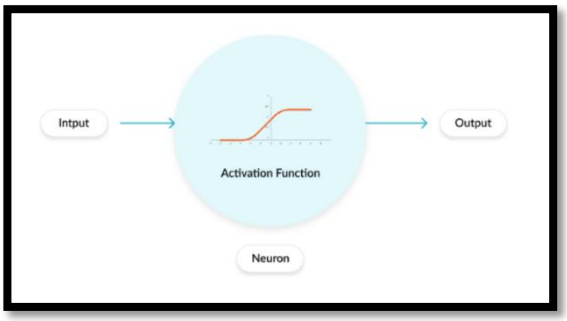


Figure 25: Activation Function

It essentially determines whether or not the neuron should be activated. The activation function of a node specifies its output in response to an input or set of inputs. As a result, the network becomes more complex.

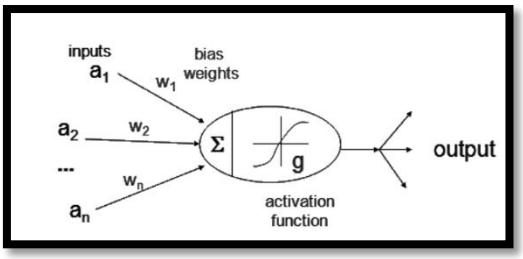


Figure 26: Input, Weights and output

-The Activation Functions are basically divided into 2 types :

1- Linear Activation Function

2-Non-linear Activation Functions

Linear Activation Functions:

A linear activation function is denoted by the equation

A = cv.

It takes the inputs and multiplies them by the neuron weights to generate an output signal proportional to the input. In some ways, a linear function is superior to a step function because it allows for more than just yes and no outputs (multiple outputs)

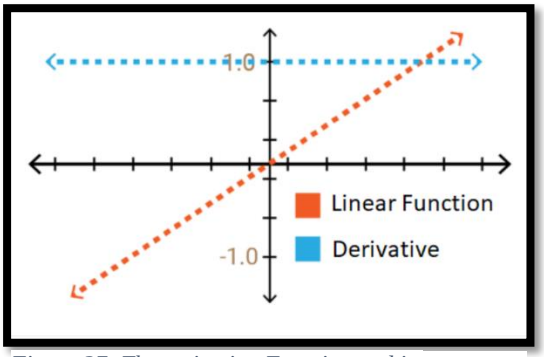


Figure 27: The activation Function and its Derivative

Pros :

- It generates a range of activations, indicating that it is not a binary activation.
- We can certainly connect a few neurons together, and if more than one fires, we can take the maximum (or softmax) value and decide according to that .

Cons :A linear activation function has two significant disadvantages:

- Backpropagation (gradient descent) cannot be used to train the model: because the derivative of the function is constant and has no relationship to the input, X. As a result, it is impossible to return and determine which weights in the input neurons can provide a more accurate prediction.
- All layers of the neural network collapse into one : with linear activation functions, regardless of the number of layers, the final layer is a linear function of the first layer (because a linear combination of linear functions is still a linear function). As a result, a linear activation function reduces a neural network to a single layer.

⇨ A neural network with a linear activation function is simply a regression model with a linear activation function. It has limited processing power and the ability to deal with complex input data with varying parameters. That is why, in deep learning, the linear activation function is rarely used.

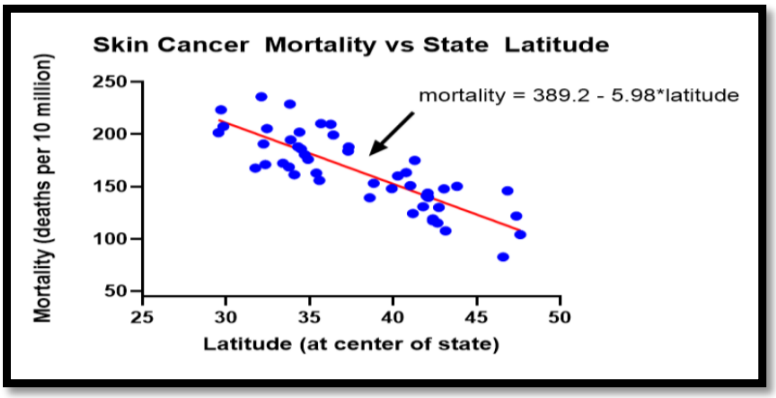


Figure 28: An illustration of a linear regression model

Non-Linear Activation Functions:

By and large, neural networks utilize non-linear activation functions, which enable the network to learn complex data, compute and learn virtually any function representing a question, and make accurate predictions. They support back-propagation due to the fact that they have a derivative function related to the inputs. Additionally, they enable the "stacking" of multiple layers of neurons in order to construct a deep neural network. Multiple hidden layers of neurons are required to accurately learn complex data sets.

- ReLU (Rectified Linear Unit Function) :

It is the most frequently used activation function. It is widely used in almost all convolutional neural networks and deep learning applications. The formula is deceptively straightforward:

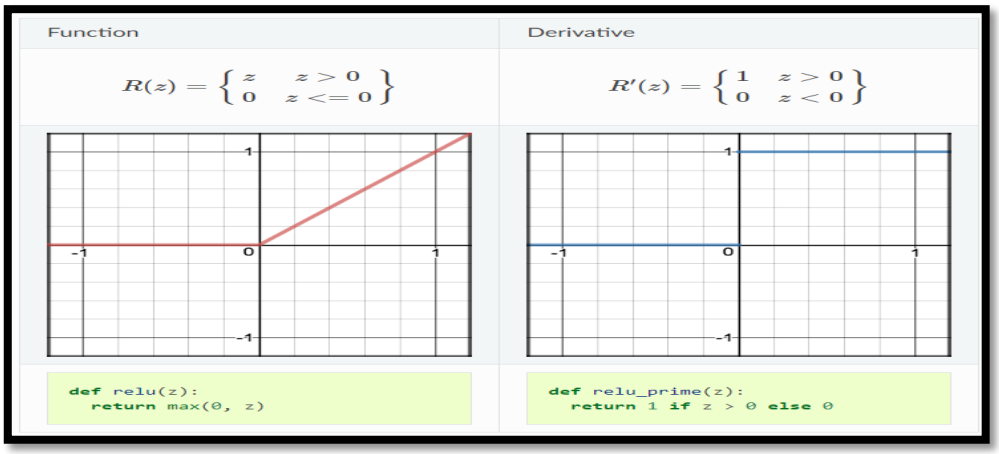


Figure 29: ReLU function and derivative

Pros:

- It circumvents and resolves the vanishing gradient problem.
- Due to the fact that it involves simpler mathematical operations, ReLu is less computationally expensive than tanh and sigmoid.

Cons:

- One of its limitations is that it should only be used within a neural network model's hidden layers.
- Certain gradients can be fragile and die during training. It may result in a weight update, preventing it from activating on any subsequent data point. In other words, ReLu has the potential to result in the death of neurons,
- In other words, for activations in the region (x<0) of ReLu, the gradient will be 0, which means that the weights will remain unchanged during descent. That is, neurons in that state will cease to respond to variations in error/input (simply because gradient is 0, nothing changes). This is referred to « the Dying ReLU problem ». So we should be very carefully to choose activation function, and activation function should be as per business requirement.

- ReLu has a range of [0, ∞).This means that it has the potential to obliterate the activation.

-Here are two simple neural networks constructed using randomly generated data from sklearn's datasets. The first used the ReLu activation function, while the second used the sigmoid activation function.

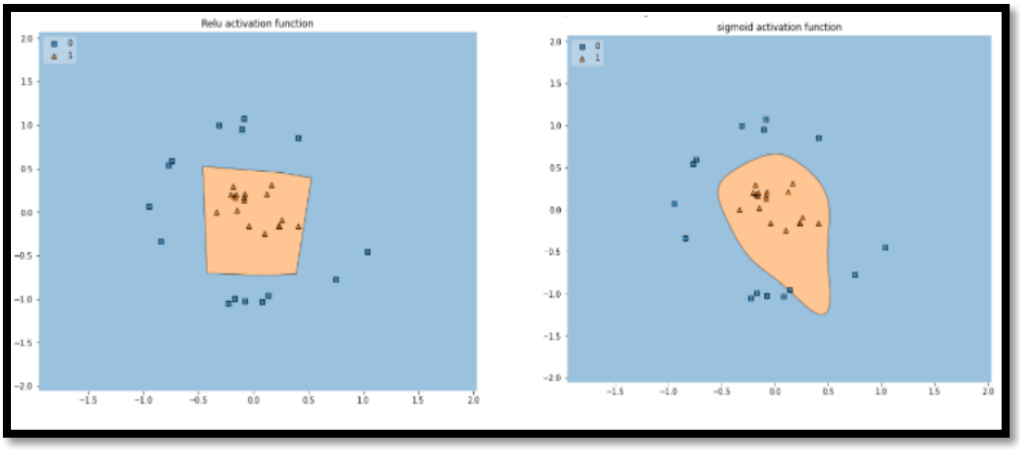


Figure 30: ReLU vs sigmoid activatin functions representations

➔ We can notice how ReLu produces a square-shaped decision boundary, whereas the sigmoid produces a smoother edge. ReLu is capable of altering the angle of a linear function. ReLu works well only when there is sufficient of it to generate a generalizable decision boundary.

- ➢ <u>ELU (Exponential Linear Unit) :</u>

The Exponential Linear Unit, or ELU for short, is a function that converges to zero more quickly and produces more accurate results. In comparison to other activation functions, ELU has an additional alpha constant that must be a positive integer.

ELU is very similar to RELU with the exception that it accepts negative inputs. For non-negative inputs, they are both in identity function form. On the other hand, ELU smooths gradually until its output equals -α, whereas RELU smooths abruptly.

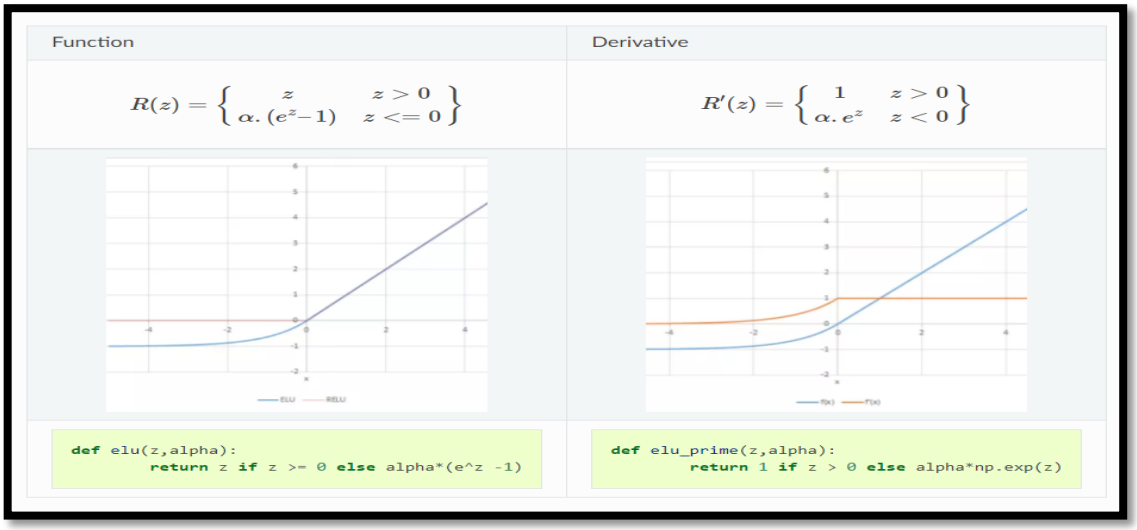


Figure 31: ELU function and derivative

Pros:

- ELU smooths gradually until its output equals -, whereas RELU smooths suddenly.
- ELU is a viable substitute for ReLU.
- In contrast to ReLU, ELU can generate negative outputs.

Cons:

- When x > 0, it can significantly blow up the activation with an output range of [0, INF].

➢ Leaky ReLU :

Leaky ReLU has the following form.

$$f(x) = \begin{cases} 0.01x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

Figure 32: Leaky ReLU form

-LeakyRelu is a ReLU variant. Rather than being

Zero at z<0, a leaky ReLU accepts a small, non-zero, constant gradient α (Normally, α=0.01) .However, the consistency's benefit across tasks is presently unclear

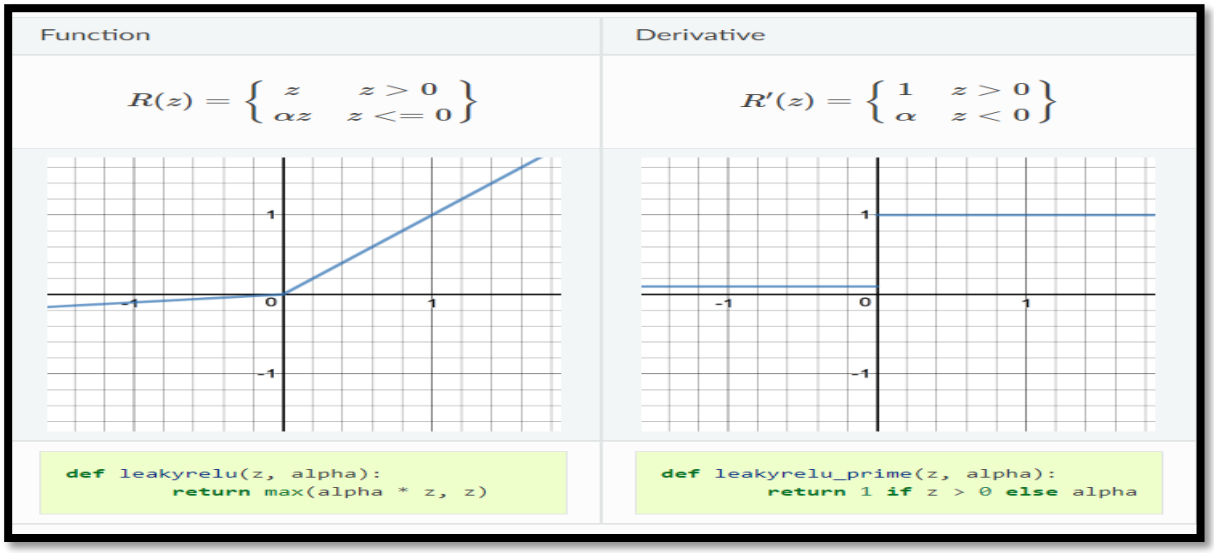


Figure 33: Leaky ReLU function and derivative

- Sigmoid Activation Function:

The Sigmoid Activation function is a straightforward function that accepts a real value as input and returns a probability that is always between 0 and 1. It is simple to use and possesses all of the desirable properties of activation functions: it is non-linear, continuously differentiable, monotonic, and has a fixed output range. The Sigmoid Function curve looks like an S-shape.

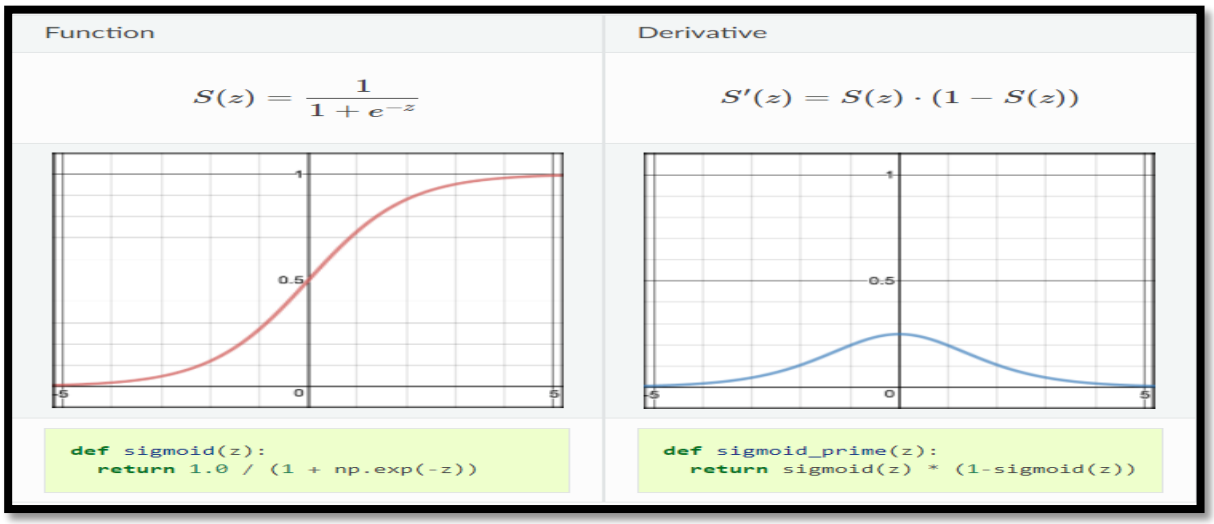


Figure 34: Sigmoid curve function and derivative

:

Pros:

- It is nonlinear by definition. Additionally, the combinations of this function are nonlinear.
- In contrast to the step function, it will provide an analog activation.
- Additionally, it has a smooth gradient.

- It is advantageous for a classifier.
- In comparison to the linear function's output range of (-inf, inf), the activation function's output will always be in the range (0,1). As a result, we've defined a range for our activations. That's convenient; it won't blow up the activations.

Cons:

- Y values tend to respond less to changes in X at either end of the sigmoid function.
- It creates a problem known as "vanishing gradients."
- Its output is not centered on zero. This causes the gradient updates to diverge excessively in different directions. 0 < outputs < 1, which complicates optimization.
- Sigmoids completely saturate and eliminate gradients.
- The network either stops learning or becomes significantly slower (depending on the use case and until the gradient/computation is limited by floating point values).

➢ Tanh (Hyperbolic Tangent Function):

Tanh is used to solve the sigmoid function's non-zero centered problem. Tanh reduces the range of a real-valued number to [-1, 1]. Additionally, its derivative is steeper, implying that it can extract more value than the sigmoid. This means it will be more efficient, as it will cover a larger range, allowing for faster learning and grading.

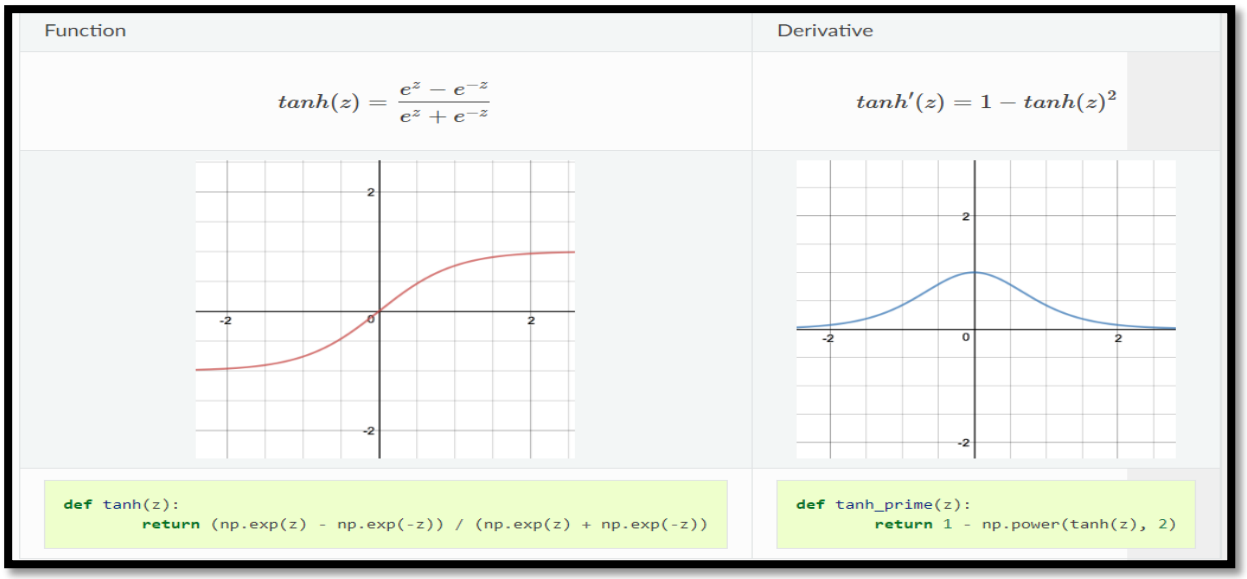


Figure 35: Tanh function and its derivative curve

Pros:

- For tanh, the gradient is stronger than for sigmoid (derivatives are steeper).

Cons:

- Though it overcomes the sigmoid's drawback, it cannot completely eliminate the vanishing gradient problem. This illustration should give us a good idea.

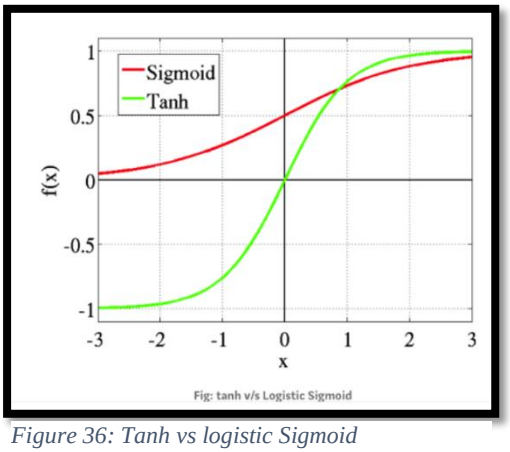


Figure 36: Tanh vs logistic Sigmoid

➔Activation functions like ReLU, Softmax, tanH, and Sigmoid are frequently utilized. In CNN models for binary classification, sigmoid and softmax functions are recommended, while softmax is typically utilized in CNN models for multi-class classification.

Chapter 5: Prerequisites

5.1 Software Requirements :

5.1.1 Programming Language:

Python : Python is a general-purpose, high-level, interpreted programming language. Its language constructs and object-oriented approach are aimed to aid programmers in writing concise, logical code for both small and large-scale projects.

Figure 37: Python Image

Python is a dynamically typed and garbage-collected programming language. It supports a variety of programming paradigms, including structured (especially procedural) programming,

object-oriented programming, and functional programming. Due to its comprehensive standard library,

it is frequently referred to as a "batteries included" language.It enables the development of a wide variety of applications. It is a popular choice among developers for projects involving Artificial Intelligence (AI), Machine Learning, and Deep Learning.

- Simplicity : Python code is concise and readable even by inexperienced developers, which is advantageous for machine learning and deep learning projects. Python development is faster than many other programming languages due to its simple syntax. Additionally, it enables the developer to test algorithms without actually implementing them.

- Visualization tools:Python includes a large number of libraries for visualization. Several of these frameworks include effective visualization tools. It is critical to present data in a human-readable format when working with AI, machine learning, and deep learning. As a result, Python is an ideal language for implementing this feature.Certain libraries, such as Matplotlib, enable data scientists to create charts, histograms, and plots that improve the representation of data and visualization. Additionally, the various APIs that Python supports facilitate the visualization process.

- Flexible integrations: Python projects can be integrated with systems written in a variety of different programming languages. This makes it much easier to integrate it with other artificial intelligence projects written in other languages. Additionally, due to Python's extensibility and portability, it can be used to perform cross-language tasks. Python's adaptability enables data scientists and developers to easily train machine learning models.

- Huge number of libraries :It comes with a variety of libraries and frameworks: Python includes a large number of libraries and frameworks that simplify programming. Additionally, this saves considerable time.

5.1.2 Some of the used libraries:

OpenCV:

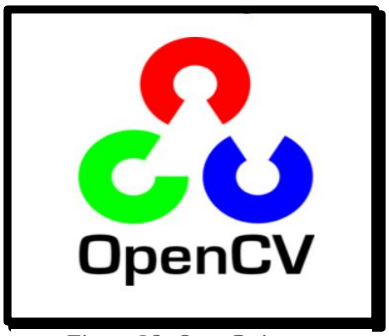


Figure 38: OpenCv image

It stands for Open Source Computer Vision. It's an Open Source BSDlicensed library that includes hundreds of advanced Computer Vision algorithms that areoptimized to use hardware acceleration. Originally written in C/C++, it now provides bindings for Python.

-OpenCV is commonly used for machine learning, image processing, performing computer vision tasks, image manipulation, and much more. It is a great library that can be

Used to perform tasks like face detection, objection tracking, landmark detection, and much more.

It supports multiple languages including python, java C++.

Although, for this article, we will be limiting to python only.

- OpenCV has a modular structure.There are shared and static libraries and a CV Namespace.

- In short, OpenCV is used in our application to easily load bitmap files that contain landscaping pictures and perform a blend operation between two pictures so that one picture can be seen in the background of another picture. This image manipulation is easily performed in a few lines of code using OpenCV versus other methods.

Tensorflow:

Figure 39: TensorFlow

It is a software library and free open-source deep learning platform created and developed by the Google Brain team for internal Google use in research and production.It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production. It offers a multitude of data and tools to configure and improve the neural network of software or an automated object.

Keras:

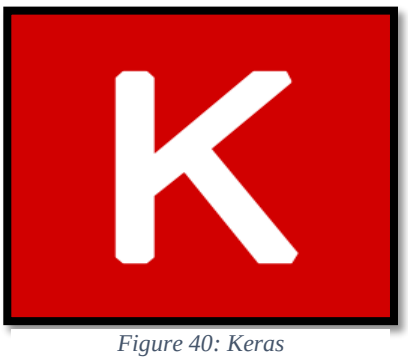


Figure 40: Keras

Keras: is an open-source software library that implements an artificial neural network interface in Python. Keras serves as a front-end to the TensorFlow library. Keras includes numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, and optimizers, as well as a variety of tools for working with image and text data and simplifying the coding required to write deep neural network code. It supports convolutional and recurrent neural networks in addition to standard neural networks. Other common utility layers such as dropout, batch normalization, and pooling are supported by

Keras as well.

Pygame:

Pygame library is an open-source module for the Python programming language that was created with the express purpose of assisting you in creating games and other multimedia applications. Pygame is a cross-platform development environment built on top of the highly portable SDL (Simple DirectMedia Layer) development library. It runs on a wide variety of platforms and operating systems.In my work i used to pla alarm sound in order to alert the driver in case of he is drowsy.

Figure 41: Pygame

5.2 Hardware requirements :

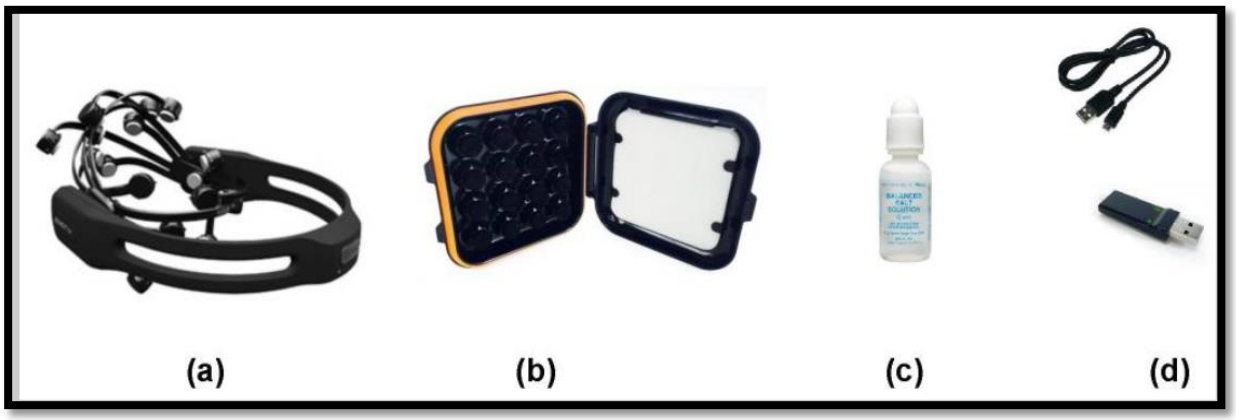


Figure 42:The different components of the Emotiv EPOC+ headset: (a) helmet, (b) fourteen-sensors box, (c) saline solution and (d) USB Key with cable.

5.3 DataSets:

5.3.1 Dataset for eye detection model:

The dataset used for this model is took it from that link: http://mrl.cs.vsb.cz/eyedataset

-The data comprises around 80000 images of people's eyes under different lighting conditions.

-In this work we annotated the dataset with the following properties (in the following order):

- subject ID : we collected data on 37 distinct individuals for this dataset (33 men and 4 women)
- Image ID: the dataset contains 84,898 images.
- gender [0 identifies a man, 1 identifies a woman]; the dataset contains information about each image's gender (man, woman)
- glasses [0 - no, 1 - yes]; each image includes information about whether the eye image contains glasses (with and without the glasses)
- eye state [0 indicates that the eye is closed, 1 indicates that it is open]; this property contains information about two different eye states (open, close)

- ❖ reflections [0 - no reflections, 1 - small reflections, 2 - large reflections]; we annotated three reflection states according to the size of the reflections (none, small, and big reflections)
- ❖ lighting conditions [0 - poor, 1 - excellent] : based on the amount of light during video capture, each image has two states (poor, excellent)
- ❖ sensor IDs [01 - RealSense, 02 - IDS, 03 - Aptina]; the dataset currently contains images captured by three distinct sensors (Intel RealSense RS 300 sensor with 640 x 480 resolution, IDS Imaging sensor with 1280 x 1024 resolution, and Aptina sensor with 752 x 480 resolution)

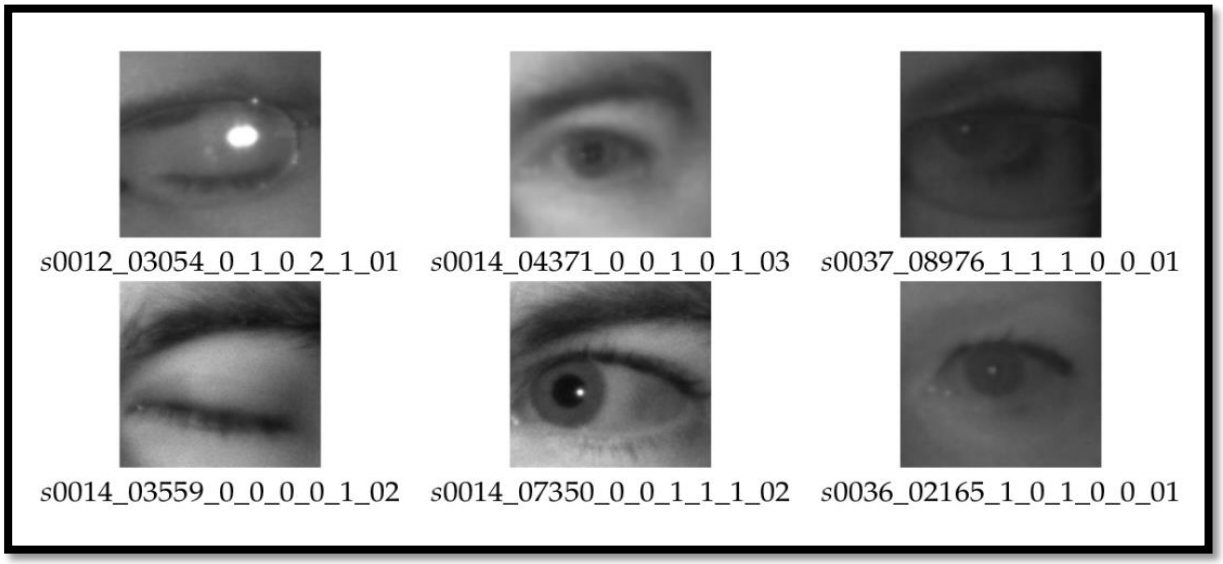


Figure 43: examples of image annotations of the proposed dataset

5.3.2 Dataset for the EEG detection:

-The dataset used for this model is took it from that link:

https://figshare.com/articles/dataset/EEG_driver_drowsiness_dataset/14273687

- In my sudy I used an open EEG dataset released in 2019 by Cao et al. The collection consists of 62 EEG data sets taken from 27 participants (aged 22 to 28) between 2005 and 2012. The participants were staff or students from the National Chiao Tung University.

A driving task requiring sustained attention was created in a virtual reality driving simulator. The participants were instructed to maintain the center of the lane while driving. To replicate slight variations in road curvature or stones on the road that cause cars to drift, random lane-departure events were incorporated that caused cars to drift to the left or right of the center lane.

Chapter 6: Modeling diagrams and system design

6.1 System Description and implementation based on EEG signals:

Electroencephalography (EEG) is one of the most effective physiological signals for detecting drowsiness in drivers because it directly measures neurophysiological brain activity. The oscillation patterns of EEG signals have been found to be strongly correlated with drowsiness. In addition, band power related features, time domain features, such as typical entropy features and multi-scale entropy features have been found to be beneficial for drowsiness detection from EEG signals. However, it is still difficult to construct a calibration-free drowsiness monitoring system with EEG due to the wide variation of EEG signals between subjects.

In recent years, researchers have paid considerable attention to deep learning. Since its initial success in numerous difficult image classification problems, it has been rapidly developed and has achieved success in numerous domains, including speech recognition, sensor readings, motion capture, spectrographs, electrocardiogram, electric devices, and simulated data. In the field of EEG signal processing, deep learning has also become a hot topic. As one of the major deep learning models, Convolutional Neural Network (CNN) is the most popular structure used for EEG signal classification.

CNN enables incremental end-to-end learning of essential characteristics from multi-channel high-dimensional EEG data without the need for a priori feature engineering, making it a potentially powerful tool for discovering drowsiness-related high-level features from a variety of EEG signals across multiple subjects.

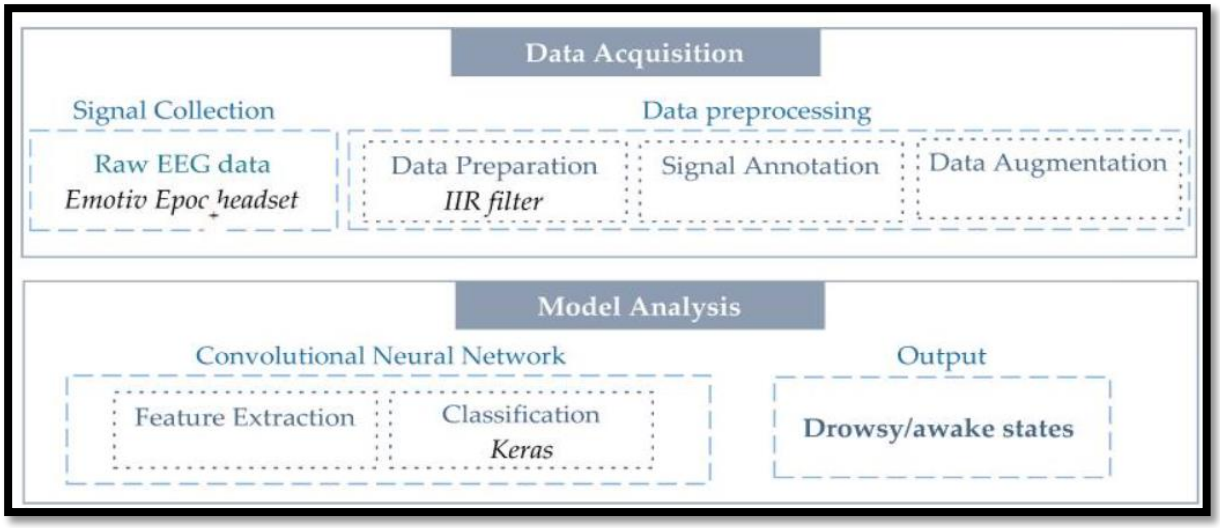


Figure 44: Pipeline of the proposed drowsiness detection system.

- **Data Acquisition**

The EEG data gathering approach includes two primary steps: signal collection utilizing the Emotiv EPOC+ headset and data preprocessing.

1) Signal Collection

Two stages, hardware and software, contribute to the development of the signal gathering step. The Emotiv EPOC+ hardware is a non-invasive brain-computer interface (BCI) used for human brain development and contextual studies. The various Emotiv EPOC+ helmet components utilized in the experimental phase included a headset, a box containing fourteen sensors, a USB key with cable for battery recharging that ensures the connection between the headset and the Emotiv Pro software, and a saline solution that ensures impedance and contact with the cortex. The saline solution is more user-friendly than medical gel and maintains effective touch with the scalp of both men and women.

The Emotiv EPOC+ headset delivers professional-grade access to brain data. The helmet is equipped with fourteen active electrodes and two reference electrodes, namely Driven Right Leg (DRL) and Common Mode Sense (CMS) .The electrodes are positioned around the participant's scalp in the following structures: frontal and anterior parietal (AF3, AF4, F3, F4, F7, F8, FC5, FC6), temporal (T7, T8), and occipital-parietal (O1, O2, P7, P8)

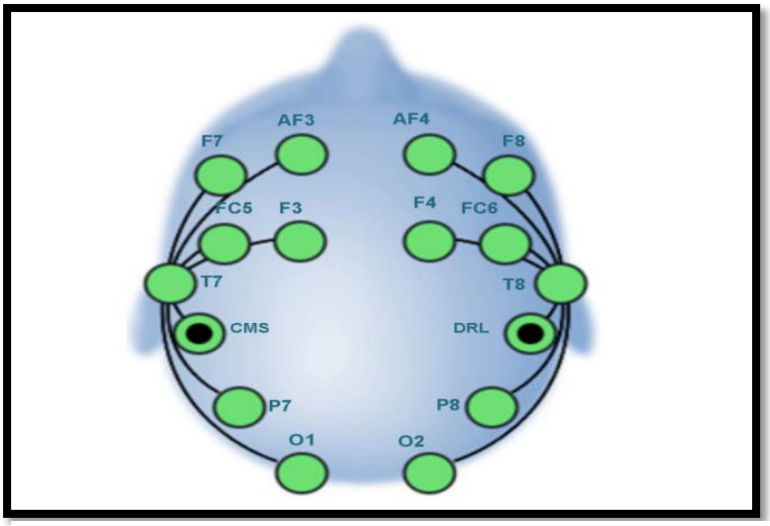


Figure 45: Location of the International System Emotiv EPOC+ helmet (10–20)

2) Data Preprocessing

Data preparation, signals annotation, and data augmentation are the unique preprocessing procedures of EEG data.

- *Data Preparation :*

Data preparation is the first stage in the denoising of EEG data: In this area, several filters based on EEG denoising algorithms have been proposed such as infinite impulse response (IIR) and finite impulse response (FIR) filters. Other sophisticated denoising techniques may be considered, although at the penalty of increased computing complexity.

- *Signals Annotation*

The central nervous system (CNS) comprises of the brain, spinal cord, and cerebellum. The latter is separated into the right and left hemispheres. Each hemisphere contains four lobes: frontal, parietal, occipital, and temporal. The EEG signal is typically divided into vast spectral frequency bands associated with EEG processors and rhythms of various frequency waves. Typically, brainwaves are categorized into five frequency and amplitude bands, including Gamma, Beta, Alpha, Theta, and Delta (as i explained them in the previous chapter) where each band wave corresponds to a participant's identifying state.

Moreover, sleepiness is a transitional state between wakefulness and sleep. During wakefulness, the human brain analyzes beta waves. The drowsy state is known as stage 1 of sleep, and its association with alpha and theta bands is confirmed. The drop in alpha band frequency and the increase in theta band frequency indicate drowsiness. The state of drowsiness is a transitory period between awake and sleep, characterized by theta brain waves. This stage is characterized by a decrease in the frequency and a rise in the amplitude of the EEG waves.

The third and fourth steps are associated with deep sleep, which is characterized by fluctuating delta waves with a low frequency and high amplitude. This investigation suggests that alpha-theta waves are the most effective bands for detecting drowsiness. Our annotation is based on

the study of Alpha-Theta waves for drowsiness/awakeness detection from, respectively, the occipital and temporal regions.

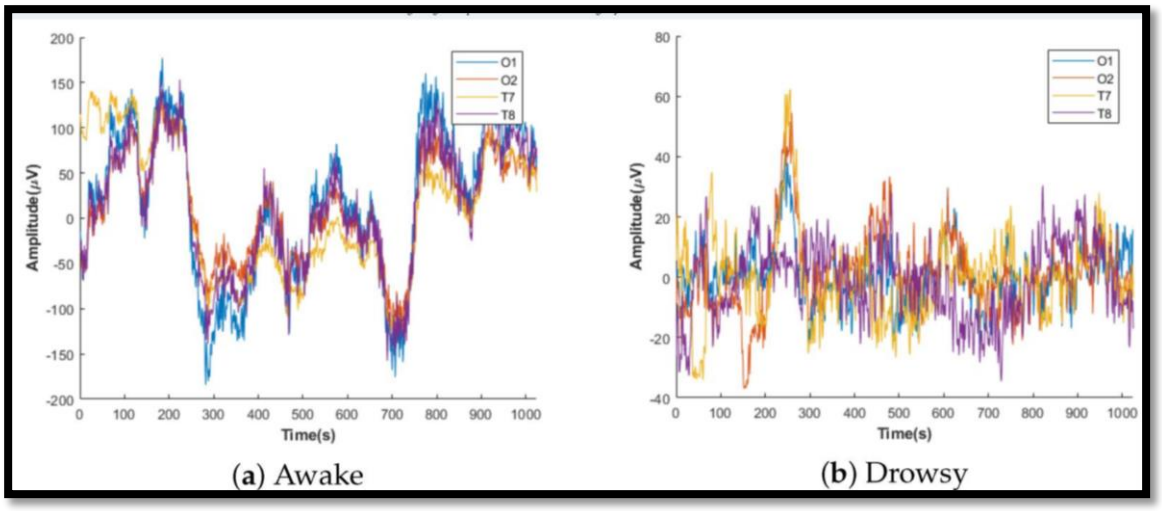


Figure 46: Annotation of drowsy (a) and awake (b) of (EEG) signal collection.

- *Data Augmentation :*

This strategy offers a number of benefits as it boosts the model's robustness against input unpredictability without reducing its effective capacity. To prevent over-fitting, we applied DA (Data Augmentation) steps solely to the training set in our work. The main idea of this procedure is to generate new samples by labeling retraining data transformations. The suggested DA approach is viewed as the reverse of dropout, in which a small amount of training data are randomly duplicated and appended to the training set. Each EEG segment of the training set, for instance, added a type of opposite operation to the dropout by extending the segments by duplicating the vectors at random time points to a specified length in the time dimension.

- **Model Analysis :**

Proposed CNN model:

We employ CNN as our model due to its extensive applicability in the processing of EEG signals and other types of time-series data. A common CNN structure is comprised of alternating convolutional and pooling layers, followed by fully linked layers. Some extension layers, such as batch normalization and dropout, have been discovered to be effective for accelerating convergence and preventing overfitting. Our goal is to construct a CNN model for identifying and visualizing cross-subject EEG features for sleepiness detection from single-channel signals. For ease of interpretation, we adopt a network layout with only important components. The model's structure is detailed below.

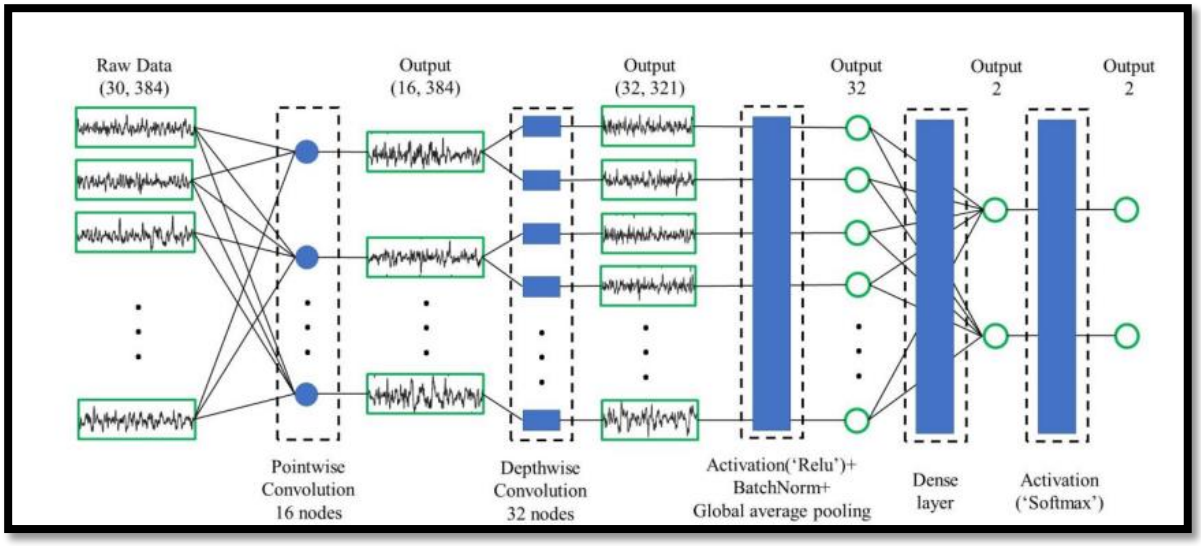


Figure 47: The used CNN model

Our Convolution Neural Network consists of the following types of layers that are majorly used:

- Convolution layer: transforms the raw EEG data into feature maps by applying kernel weights to filters.
- Batch normalization: Normalizes the given data by fixing the mean and variance of a given input layer's data.
- Dropout: Drops nodes at random way to minimize the overfitting issue.
- Max poling: reduces the size of feature maps by considering their maximum values.

The output :

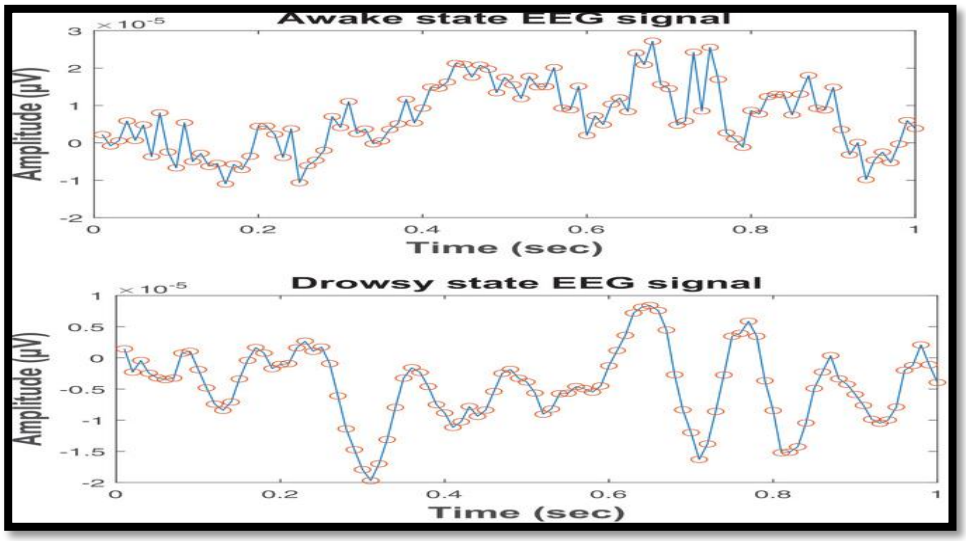


Figure 48: The output signal in both cases

6.2 System description and modeling based on eyes detection:

6.2.1 System Modeling:

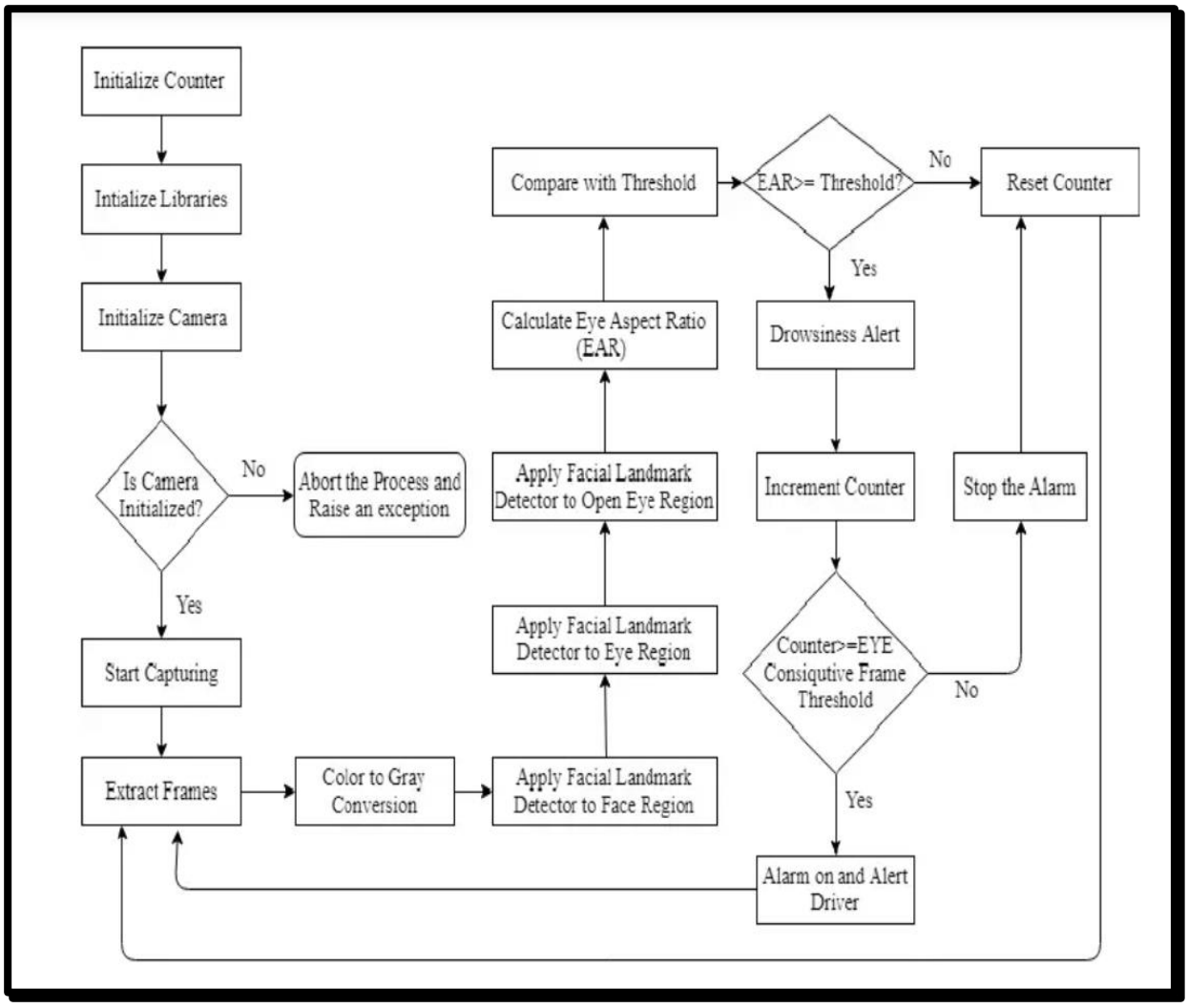


Figure 49:Flow chart of System

Figure 50: Level 0 of the system

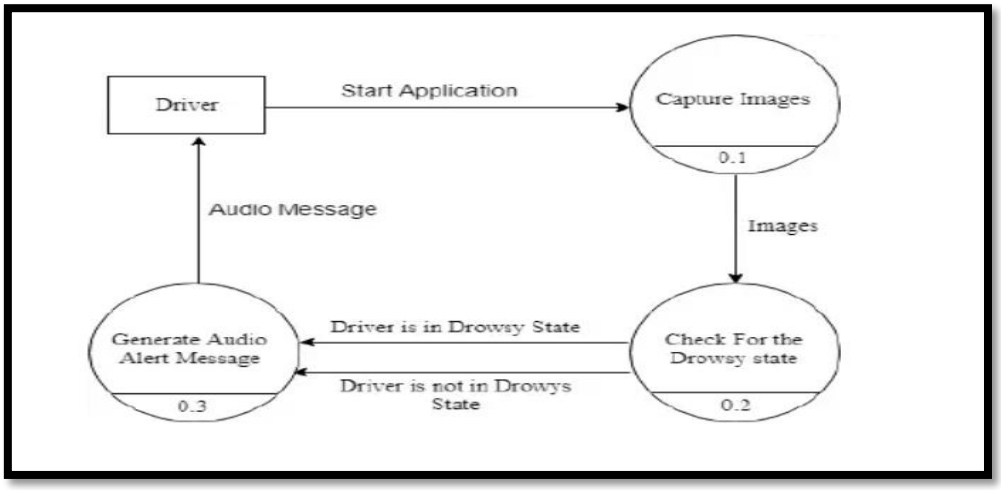


Figure 51: Level 1 of the system

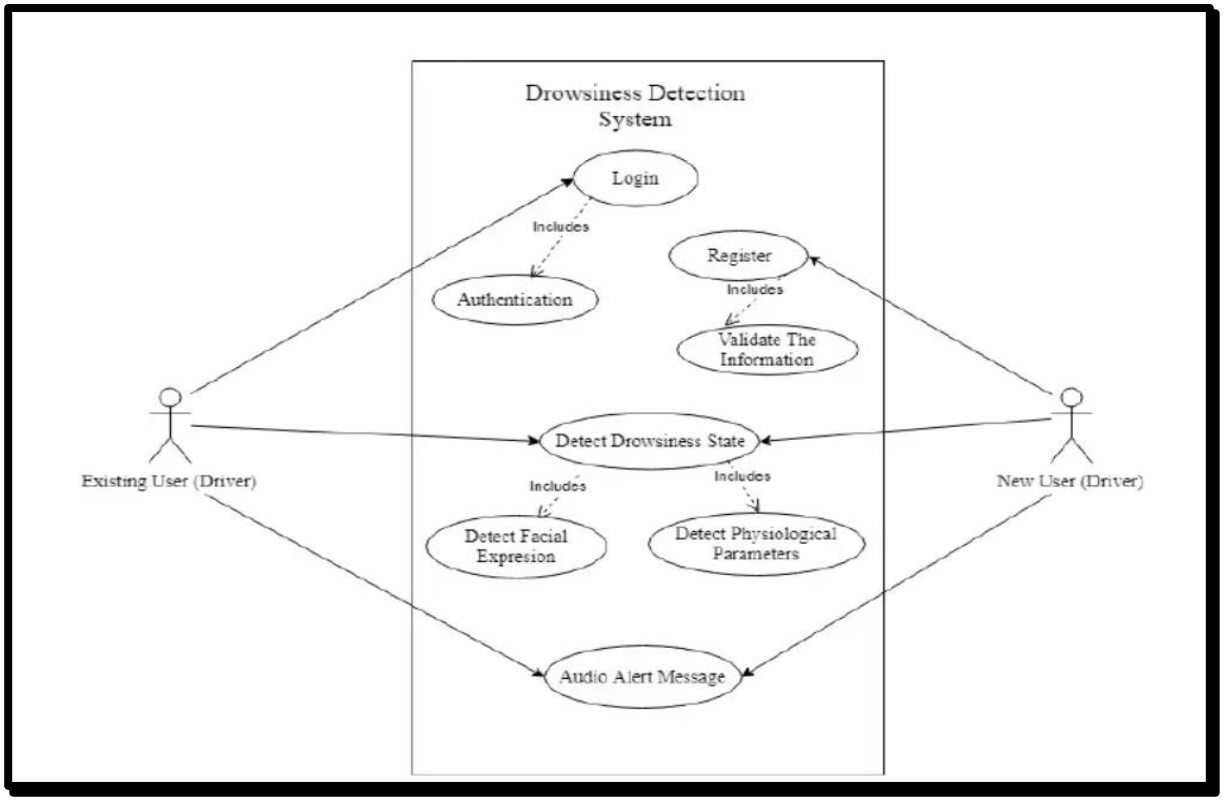


Figure 52: Use case diagram

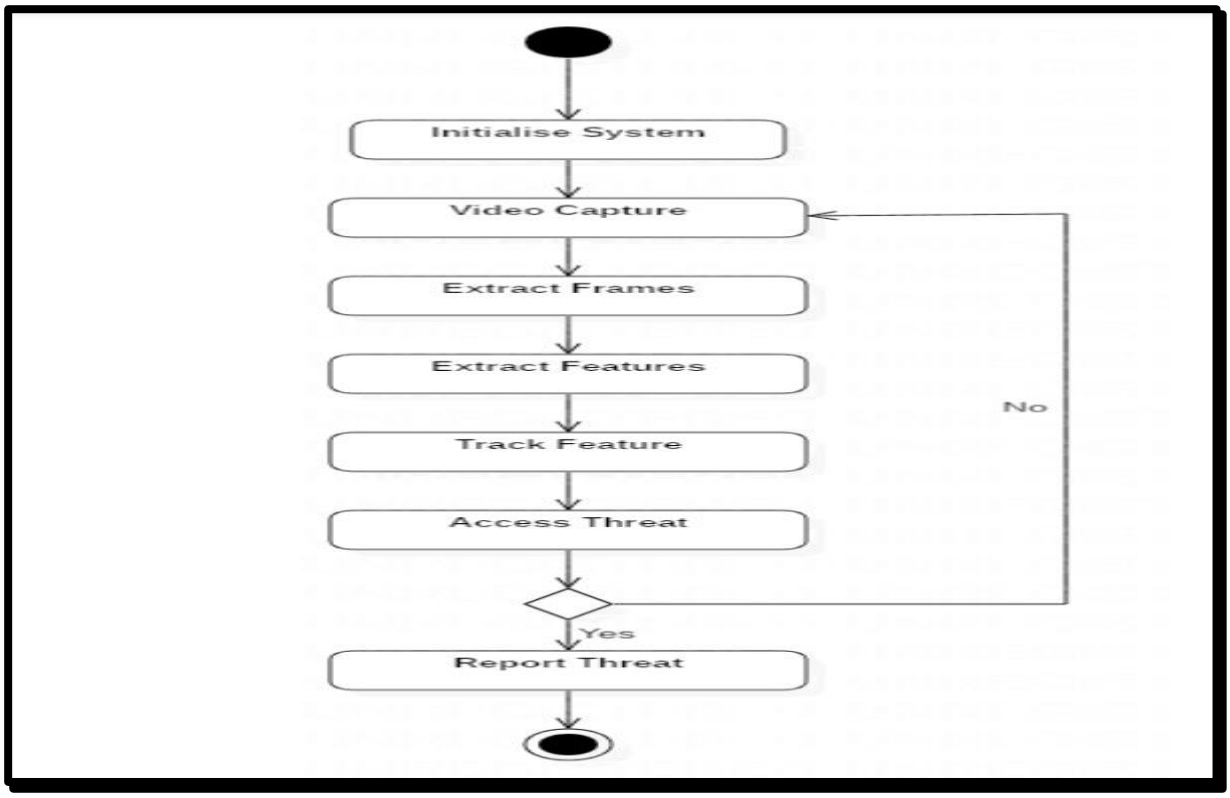


Figure 53: Activity diagram

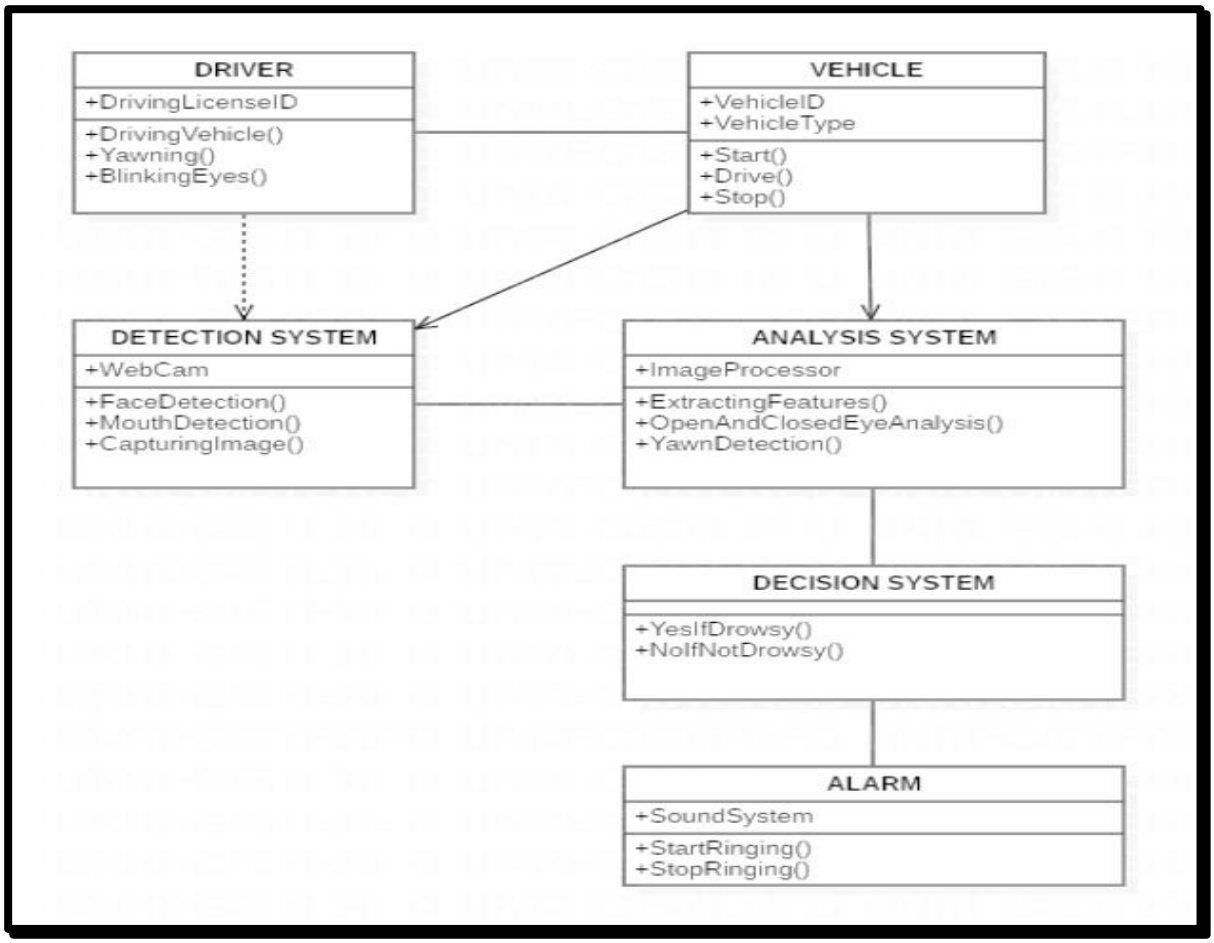


Figure 54: Class diagram

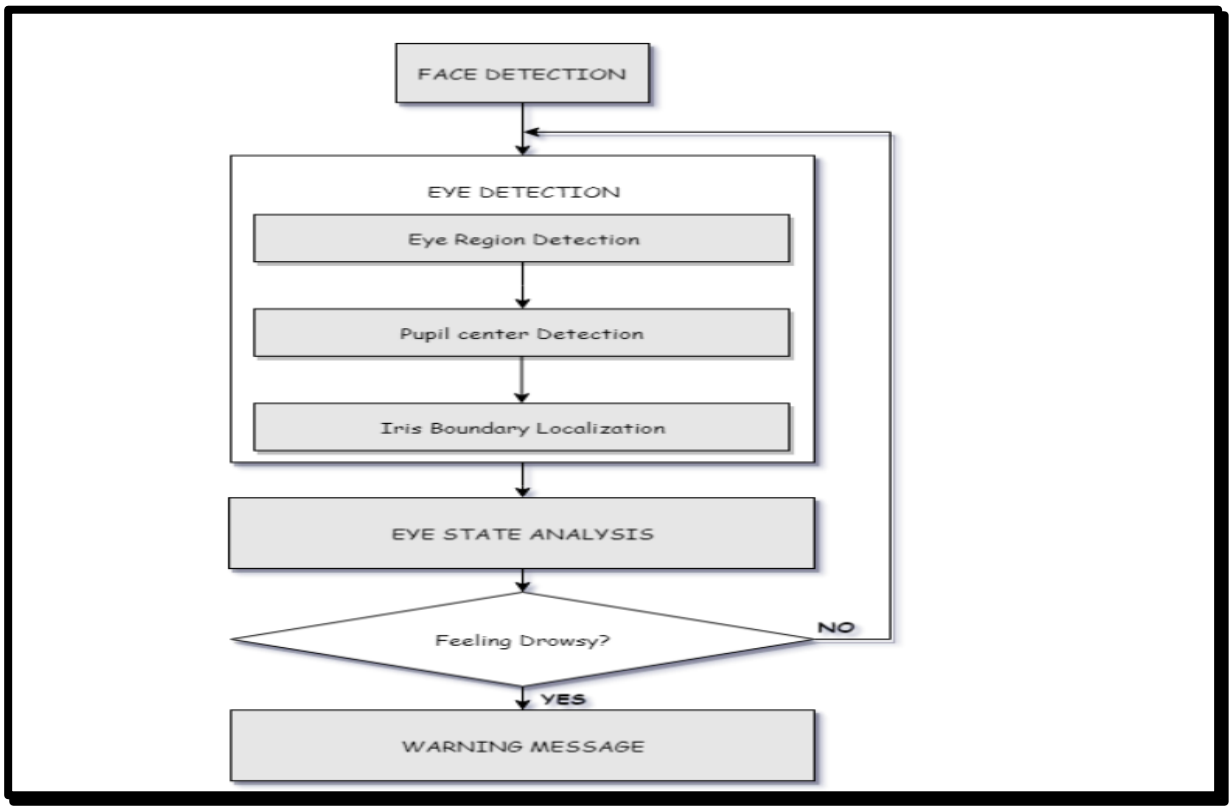


Figure 55: Block diagram

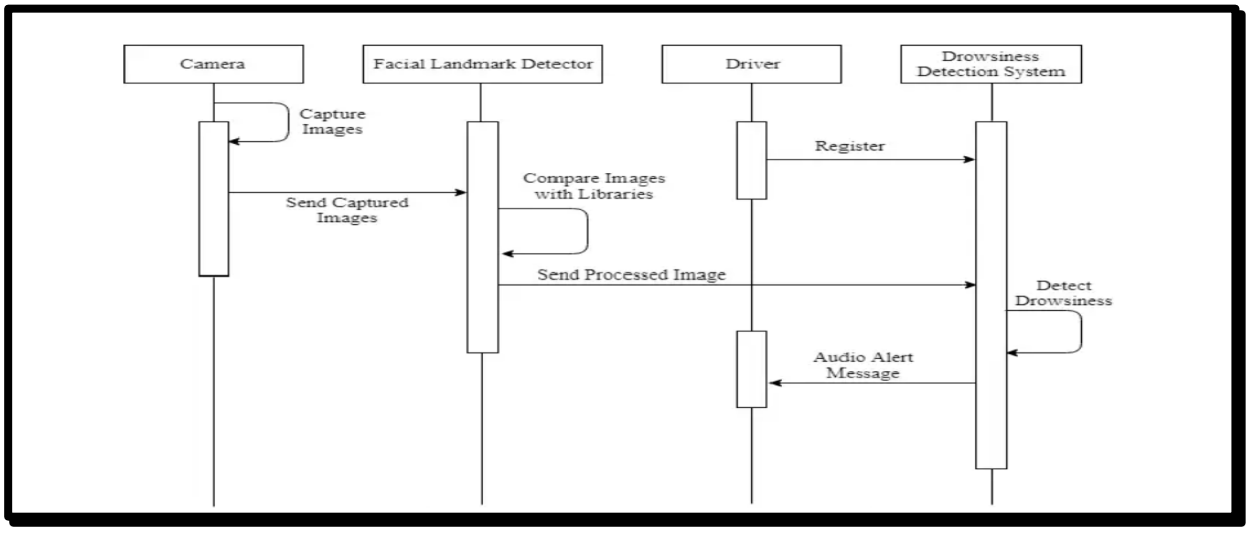


Figure 56: Sequence diagram

6.2.2 System description and implementation:

Login/ Sign up:

The initial stage in the system is the easy and flexible credential interface, which requires the user/driver to enter the login credentials or sign up to become a member of the system, as illustrated in the following Figure .

Figure 57: UI of the system

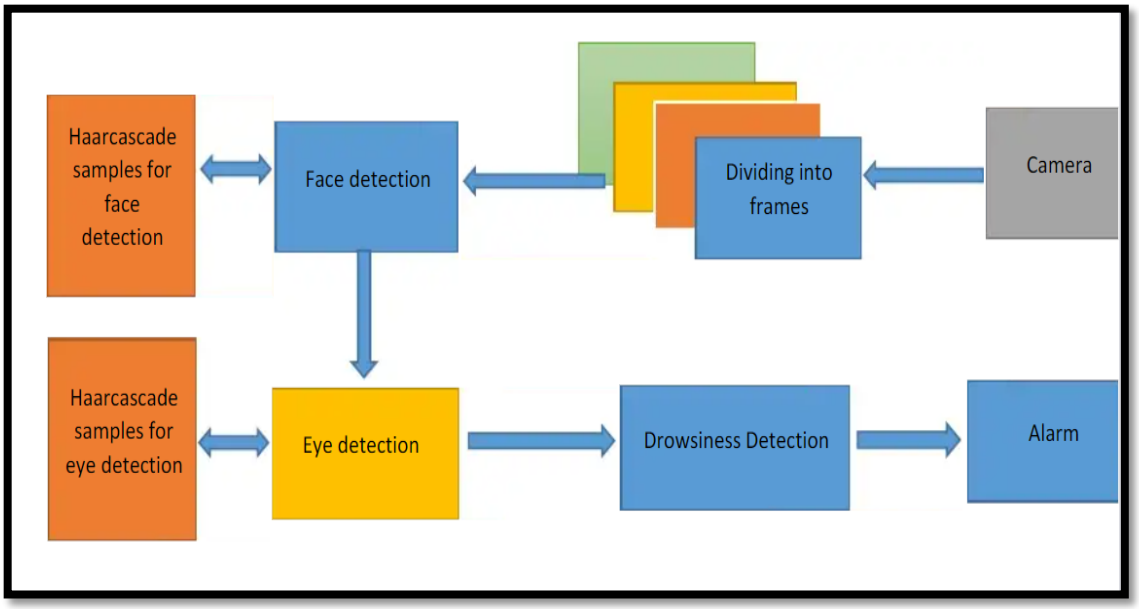


Figure 58: Proposed System Architecture

Flowchart and algorithm:

The various detection stages are discussed as:

Dividing into frames: This module takes live video as input and transforms it into a series of images / frames that are then processed.

Face Detection: It employs Haar feature-based cascade classifiers for face recognition. This is an excellent approach for object detection developed by Paul Viola and Michael Jones in their 2001 publication, "Rapid Object Detection Using a Boosted Cascade of Simple Features." It is a machine-learning strategy that involves training a cascade function on a large number of positive and negative photos. It is then used to determine the presence of objects in other photos.

- The face detection function takes one frame at a time from the frames provided by the frame grabber, and in each and every frame it tries to detect the face of the automobile driver.

- Here, we'll experiment with facial detection. To start, the algorithm requires a large number of positive photos (images with faces) and negative images (images without faces) for training. Following that, we'll need to extract the most important features from it.

- This is accomplished through the use of the Haar features depicted in the following graphic. They behave identically to our convolutional kernel. Each feature is a single value calculated by subtracting the total of the pixels beneath the white rectangle from the sum of the pixels beneath the black rectangle.

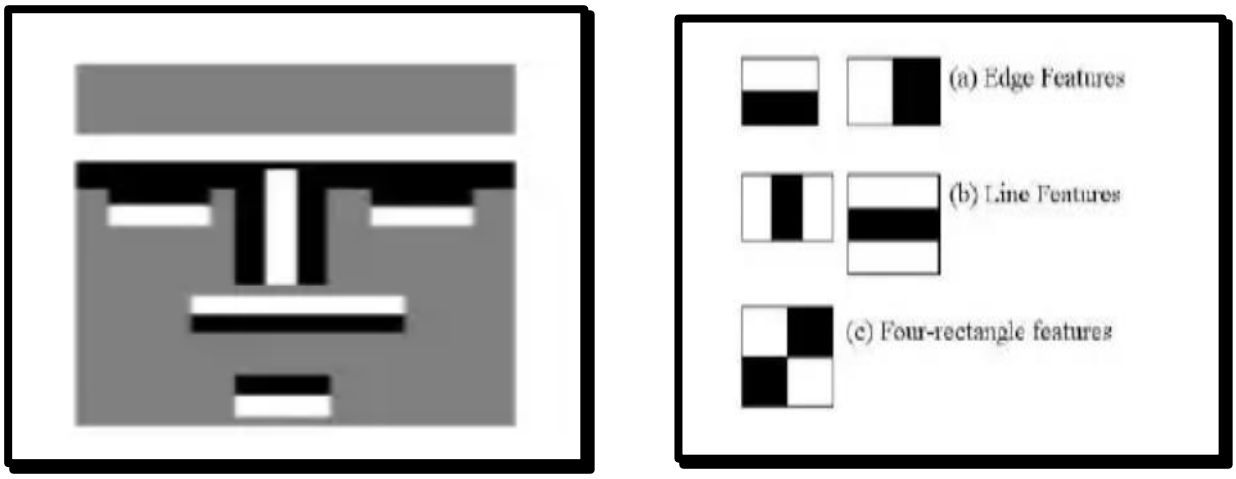


Figure 59: 5 haar like features & example

-To determine the face region, we use a cascaded classifier with Haar-like features. To begin, the compensated image is segmented into a number of rectangular sections that can be located and scaled anywhere within the original image. Due to the distinction between facial features, Haar-like features are effective for real-time face detection.These can be obtained by calculating the sum of the pixel values within rectangular areas by two. The differences in composition between the black and white regions illustrate the features. Cascaded Adaboost classifier is a type of strong classifier composed of many weak classifiers. Adaboost algorithm is used to constrain each weak classifier. The face region can be identified by passing a candidate sample through a cascaded Adaboost classifier. Almost all face samples are acceptable, but non-facial samples can be denied.

Eyes detection: Once the face detection function has detected the face of the automobile driver, the eyes detection function tries to detect the automobile driver's eyes. This is achieved by making use of a set of pre-defined Haarcascade samples.

-To detect eyes in the system, we employed facial landmark prediction. Facial landmarks are used to identify and symbolize prominent facial regions, such as the following:

- Eyes
- Eyebrows
- Nose
- Mouth
- Jawline

-Face alignment, head pose estimation, faces wapping, and blink detection have all been effectively applied using facial landmarks. In the context of facial landmarks, our objective is to use shape prediction methods to detect significant facial structures on the face. Thus, detecting face landmarks is a two-step process:

- Localize the face in the image.
- Detect the key facial structures on the face ROI.

- Localize the face in the image: The face image is localized using Haar feature-based cascadeclassifiers, as stated in the preceding stage of our approach, namely face detection. Recognize the critical facial structures on the face ROI: While there are numerous facial landmark detectors, all of them attempt to localize and label the following facial regions:

- Nose
- Mouth
- Left eyebrow
- Right eyebrow
- Right eye
- Left eye

- We initialize the face landmark detector by using the pretrained model. The model is based on ensemble regression trees because the model will predict continuous numbers.

This approach begins by utilizing the following:

- An image containing a training set of labeled face landmarks. These pictures have been manually labeled, with the (x, y) coordinates of regions surrounding each facial structure.
- Priors, or more precisely, the probability associated with the distance between two input pixels.

- The dlib library's pre-trained facial landmark detector is used to estimate the locations of 68 (x, y)-coordinates that correspond to facial structures on the face.

The graphic below illustrates the indices for the 68 coordinates.

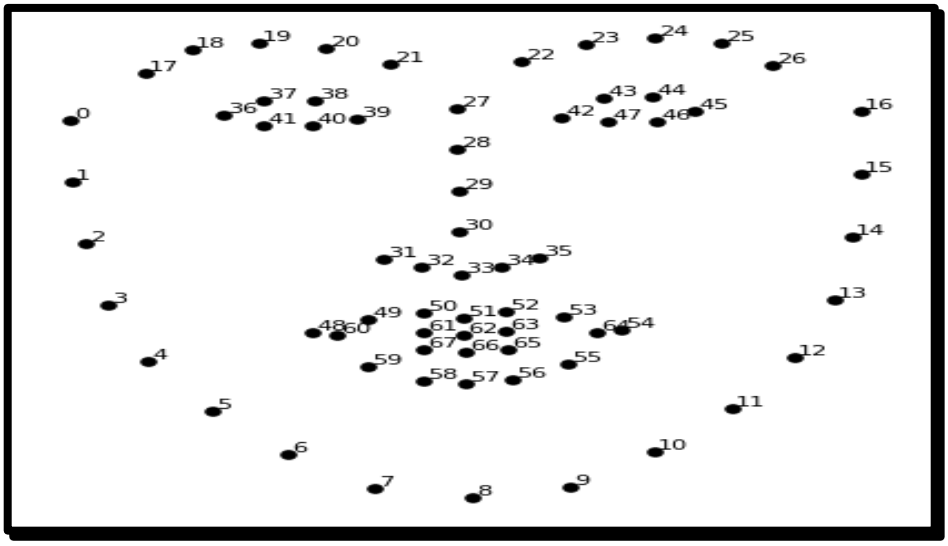


Figure 60: Visualizing the 68 facial landmark coordinate

Figure 61: Detection of the eyes

- By using the facial landmark index shown below, we can identify and access both eye regions.

- The right eye using [36, 42].
- The left eye with [42, 48].

Recognize the state of the eye

-The eye area can be determined from optical flow, by sparse tracking or by frame-to-frame intensity differencing and adaptive thresholding. After that, a decision is made whether the eyes

are or are not covered by eyelids. A different approach is to infer the state of the eye opening from a single image, for example, through correlation matching with open and closed eye templates, heuristic horizontal or vertical image intensity projections over the eye region, parametric model fitting to determine the eyelids, or active shape models.

- A significant disadvantage of the preceding systems is that they frequently implicitly place excessive requirements on the setup, such as relative face-camera posture (head orientation), picture resolution, lighting, and motion dynamics. Especially heuristic systems based on raw image intensity are likely to be extremely sensitive, despite their real-time performance.

As a result, i present a basic but effective approach for detecting eye blinks via a recent facial landmark detector. From the landmarks, a single scalar quantity corresponding to the level of the eye opening is produced. Finally, using a per-frame sequence of eye-opening estimations, an SVM classifier trained on examples of blinking and non-blinking patterns is used to detect eye blinks.

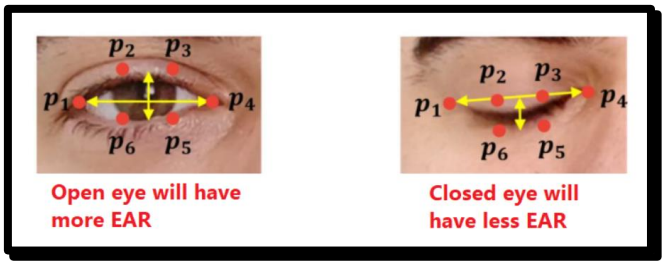
Calculation of the Eye Aspected Ratio:

The eye landmarks are determined for each video frame. The Eye Aspect Ratio (EAR) is calculated as the relationship between the eye's height and width.

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

- ||p2-p6|| :is the distance between points p2 and p6

Figure 62:The EAR formula



- The more the EAR, the more widely eye is open. We would decide a minmum EAR value and used this to decide if the eye is closed or not

Figure 63:The Eye Aspected Ratio

Determination of the Eye State:

Finally, the eye state is determined based on the EAR calculated in the preceding phase. If the distance is 0 or close to zero, the eye state is categorized as "closed," whereas if the distance is greater than zero, the eye state is classified as "open."

Drowsiness Detection:

The final stage of the algorithm is to determine the individual's condition based on a pre-defined sleepiness condition. A person's average blink duration is between 100 and 400 milliseconds (i.e. 0.1-0.4 of a second). As a result, if a person is drowsy, his or her eye closure must be greater than this interval. We established a time limit of 5 seconds. If the eyes are closed for five seconds or longer, drowsiness is recognized and an alert pop is triggered.

Figure 64: Drowsiness State Detection

Chapter 7: Results and Discussions

7.1 Results:

Implementation of drowsiness detection with by following these steps :Capturing video with a camera during its runtime is a success. The captured video was separated into frames and studied individually. Face detection was successful, followed by eye detection. If successive frames of eye closure are observed, it is classified as drowsy condition; otherwise, it is considered a normal blink, and the cycle of recording images and analyzing the driver's state is repeated.

Figure 65: Login Page of the Drowsiness system

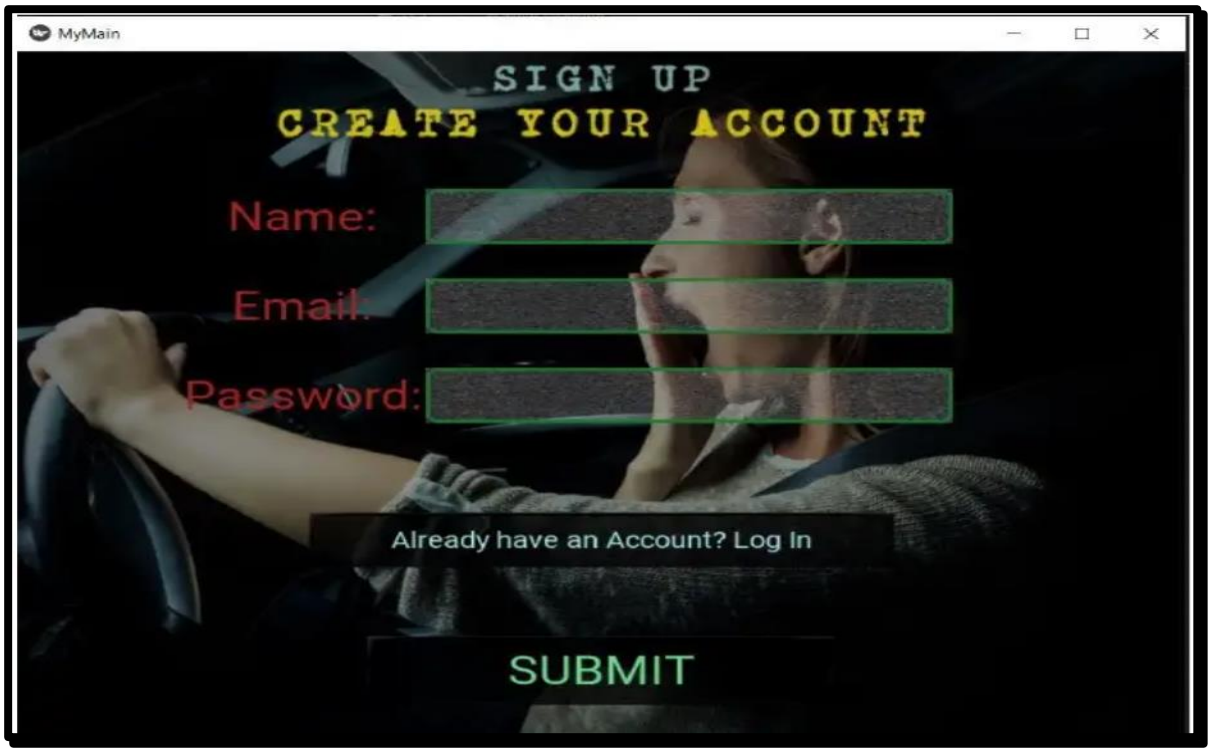


Figure 66: Sign Up page of the system

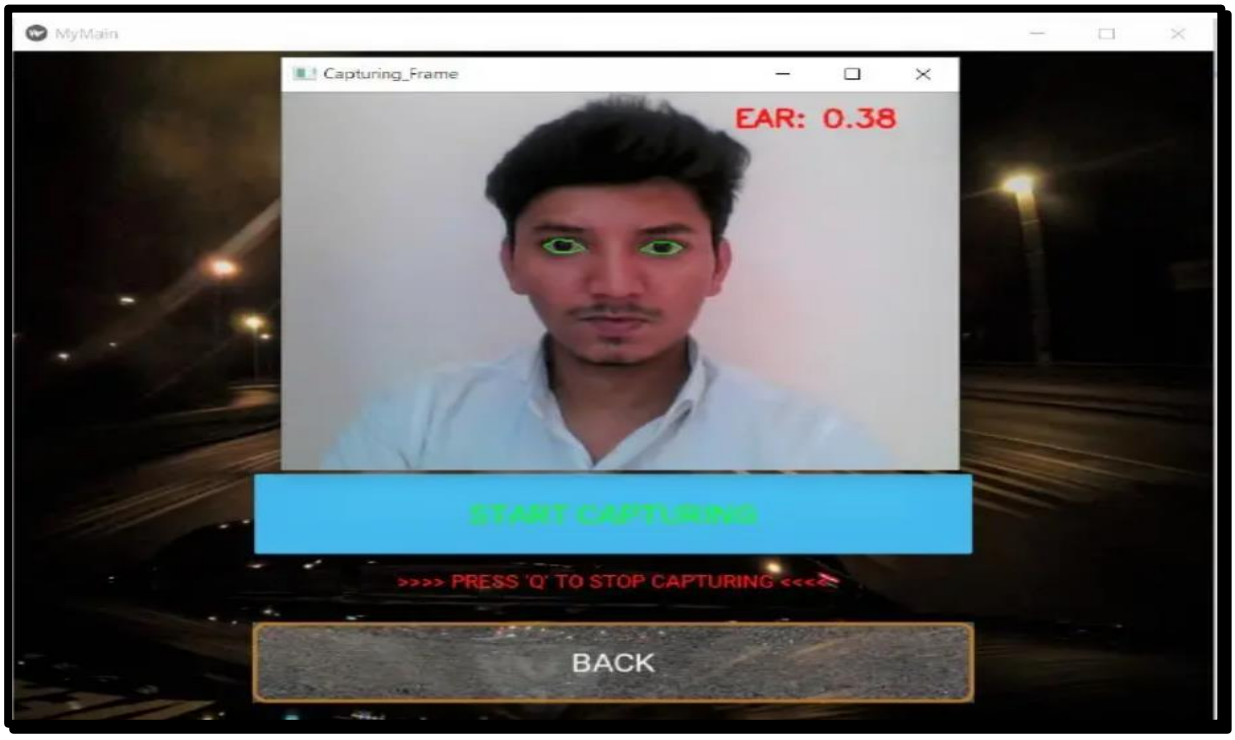


Figure 67: Eyes System detection

7.2 Future Work:

Our model is intended to detect drowsy eye states and to sound an alert or warning in the form of an audio sound. However, the driver's response after being warned may not be sufficient to avoid an accident, which means that if the driver is slow to reply to the warning signals, an

accident may occur. To avoid this, we can build and install a motor-driven system that is synchronized with the warning signal, causing the vehicle to automatically slow down upon receiving the warning signal.

Additionally, we can provide the user with an Android application that will provide information about his or her level of drowsiness during any journey. The user will be able to determine the Normal state, Drowsy state, and the number of times the eyes were blinking based on the number of frames captured. Which is illustrated in following Figure.

Conclusion

For years, driver drowsiness has been one of the major causes of road accidents and can lead to severe physical injuries, deaths and significant economic losses. Statistics indicate the need of a reliable driver drowsiness detection system which could alert the driver before a mishap happens.

It can be concluded that by combining these methods that the results are improved and can be highly decisive to provide precisive feedback but that's not the case in every scenario. A benefit of the method to detect drowsiness is intelligent sleepiness detection and use of various parameters in a long time. This advantage leads to detecting drowsiness in early stages and activate the alarm before a car accident occur.

Obviously the goal is to minimize false positives, while missing very few true positives and this would require much more testing beacuse sleepiness is very complicated phenomenon to be completely detected.

List of Figures

Refrences

Online Websites :

[1]: "Automatic Driver Drowsiness Detection Using Haar Algorithm and Support Vector Machine Techniques ", https://scialert.net/fulltext/?doi=ajaps.2015.149.157 , Sep.05. 2021

[2]: "What Is Computer Vision & How Does it Work? An Introduction"

https://xd.adobe.com/ideas/principles/emerging-technology/what-is-computer-vision-how-does-it-work/ , Oct.12. 2021

[3]: "Drowsiness detection with OpenCV", https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/, Jan.2.2022

[4]: "Driver Drowsiness Detection System with OpenCV & Keras"https://data-flair.training/blogs/python-project-driver-drowsiness-detection-system/ , Feb.8.2022

[5]: "The Introductory Guide to BCI (Brain-Computer Interface)", https://www.emotiv.com/bci-guide/, Feb.10.2022

[6]: "Detecting Driver Drowsiness Based on Sensors: A Review", https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3571819/, Feb.14.2022

[7]: "EEG (electroencephalogram)",https://www.mayoclinic.org/tests-procedures/eeg/about/pac-20393875, Mar.2.2022

[8]: "Technological Basics of EEG Recording and Operation of Apparatus", https://www.sciencedirect.com/topics/agricultural-and-biological-sciences/brain-waves, Apr.5.2022

[9] : "What Is A Brain Computer Interface?" ,https://www.scienceabc.com/innovation/what-is-a-brain-computer interface.html#:~:text=There%20are%20two%20kinds%20of,and%20Invasive%20Brain%2D Computer%20Interface , May.6.2022

[10]: "Convolutional Neural Networks, Explained", https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939, May.15.2022

papers:

[1] : "Driver drowsiness detection using ANN image processing" , https://iopscience.iop.org/article/10.1088/1757-899X/252/1/012097/pdf , May.17.2022

[2] : "Blink behaviour based drowsiness detection – method development and validation" , https://www.diva-portal.org/smash/get/diva2:19913/FULLTEXT01.pdf, Apr.3.2022

[3]: "A Project Report On INTELLIGENT CAR USING IMAGE PROCESSING", https://www.academia.edu/6861884/A_Project_Report_On_INTELLIGENT_CAR_USING_IMAGE_PROCESSING, Apr.10.2022

[4] : “Neurosky EEG Biosensor Using in Education” , https://www.researchgate.net/publication/311850592_Neurosky_EEG_Biosensor_Using_in_Education, May.10.2022

[5]: “BRAIN –COMPUTER INTERFACE” https://nexusacademicpublishers.com/uploads/portals/Brain-Computer_Interface.pdf, May.11.2022

Appendix

CompactCNN.py

```
import torch

class CompactCNN(torch.nn.Module):

    def __init__(self, classes=2 ,channels=32, kernelLength=64,sampleLength=384):

        super(CompactCNN, self).__init__()

        self.kernelLength=kernelLength


        self.conv = torch.nn.Conv2d(1,channels,(1,kernelLength))

        self.batch = Batchlayer(channels)

        self.GAP = torch.nn.AvgPool2d((1,sampleLength-kernelLength+1))

        self.fc = torch.nn.Linear(channels, classes)

        self.softmax=torch.nn.LogSoftmax(dim=1)


    def forward(self, inputdata):

        intermediate = self.conv(inputdata)

        intermediate = self.batch(intermediate)

        intermediate = torch.nn.ELU()(intermediate)

        intermediate = self.GAP(intermediate)

        intermediate = intermediate.view(intermediate.size()[0], -1)

        intermediate = self.fc(intermediate)

        output = self.softmax(intermediate)

        return output
```

```
def normalizelayer(data):
    eps=1e-05
    a_mean=data-torch.mean(data, [0,2,3],True).expand(int(data.size(0)), int(data.size(1)), int(data.size(2)),int(data.size(3)))
b=torch.div(a_mean,torch.sqrt(torch.mean((a_mean)**2,[0,2,3],True)+eps).expand(int(data.size(0)), int(data.size(1)), int(data.size(2)),int(data.size(3))))
    return b
class Batchlayer(torch.nn.Module):
    def __init__(self, dim):
        super(Batchlayer, self).__init__()
        self.gamma=torch.nn.Parameter(torch.Tensor(1,dim,1,1))
        self.beta=torch.nn.Parameter(torch.Tensor(1,dim,1,1))
        self.gamma.data.uniform_(-0.1, 0.1)
        self.beta.data.uniform_(-0.1, 0.1)
    def forward(self, input):
        data=normalizelayer(input)
        gammamatrix=self.gamma.expand(int(data.size(0)), int(data.size(1)), int(data.size(2)),int(data.size(3)))
        betamatrix = self.beta.expand(int(data.size(0)), int(data.size(1)), int(data.size(2)),int(data.size(3)))
        return data*gammamatrix+betamatrix
```

LeaveOneOutput_acc.py :

```
import scipy.io as sio
import numpy as np
from sklearn.metrics import accuracy_score
import torch.optim as optim
```

```
from CompactCNN import CompactCNN
torch.cuda.empty_cache()
torch.manual_seed(0)
def run():
#   load data from the file
    filename = r'dataset.mat'
    tmp = sio.loadmat(filename)
    xdata=np.array(tmp['EEGsample'])
    label=np.array(tmp['substate'])
    subIdx=np.array(tmp['subindex'])
    label.astype(int)
    subIdx.astype(int)
    samplenum=label.shape[0]
    channelnum=30
    subjnum=11
    samplelength=3
    sf=128
#   define the learning rate, batch size and epoches
    lr=1e-2
    batch_size = 50
    n_epoch =6
#   data contains the label of samples
    ydata=np.zeros(samplenum,dtype=np.longlong)
    for i in range(samplenum):
```

```
        ydata[i]=label[i]
# only channel 28 is used, which corresponds to the Oz channel
    selectedchan=[28]
# update the xdata and channel number
    xdata=xdata[:,selectedchan,:]
    channelnum=len(selectedchan)
# the result stores accuracies of every subject
    results=np.zeros(subjnum)
# it performs leave-one-subject-out training and classfication
# for each iteration, the subject i is the testing subject while all the other subjects are the training subjects.
    for i in range(1,subjnum+1):
# form the training data
        trainindx=np.where(subIdx != i)[0]
        xtrain=xdata[trainindx]
        x_train = xtrain.reshape(xtrain.shape[0],1,channelnum, samplelength*sf)
        y_train=ydata[trainindx]
# form the testing data
        testindx=np.where(subIdx == i)[0]
        xtest=xdata[testindx]
        x_test = xtest.reshape(xtest.shape[0], 1,channelnum, samplelength*sf)
        y_test=ydata[testindx]
        train = torch.utils.data.TensorDataset(torch.from_numpy(x_train),
torch.from_numpy(y_train))
        train_loader = torch.utils.data.DataLoader(train, batch_size=batch_size, shuffle=True)
```

```
# load the CNN model to deal with 1D EEG signals
    my_net = CompactCNN().double().cuda()
    optimizer = optim.Adam(my_net.parameters(), lr=lr)
    loss_class = torch.nn.NLLLoss().cuda()
    for p in my_net.parameters():
        p.requires_grad = True
# train the classifier
    for epoch in range(n_epoch):
        for j, data in enumerate(train_loader, 0):
            inputs, labels = data
            input_data = inputs.cuda()
            class_label = labels.cuda()
            my_net.zero_grad()
            my_net.train()
            class_output= my_net(input_data)
            err_s_label = loss_class(class_output, class_label)
            err = err_s_label
            err.backward()
            optimizer.step()
# test the results
    my_net.train(False)
    with torch.no_grad():
        x_test =  torch.DoubleTensor(x_test).cuda()
        answer = my_net(x_test)
```

```
            probs=answer.cpu().numpy()
            preds = probs.argmax(axis = -1)
            acc=accuracy_score(y_test, preds)
            print(acc)
            results[i-1]=acc
    print('mean accuracy:',np.mean(results))
if __name__ == '__main__':
    run()
```

 # File VisualizationTech.py

```
import torch
import scipy.io as sio
import numpy as np
import torch.optim as optim
import matplotlib.pyplot as plt
from matplotlib.collections import LineCollection
from scipy.integrate import simps
from mne.time_frequency import psd_array_multitaper
import matplotlib.gridspec as gridspec
from CompactCNN import CompactCNN
plt.rcParams.update({'font.size': 12})
torch.cuda.empty_cache()
torch.manual_seed(0)
class FeatureVis():
    def __init__(self, model):
```

```
        self.model = model
        self.model.eval()
    def generate_heatmap(self,
allsignals,sampleidx,subid,samplelabel,multichannelsignal,likelihood):
  if likelihood[0]>likelihood[1]:
            state=0
        else:
            state=1
        if samplelabel==0:
            labelstr='alert'
        else:
            labelstr='drowsy'
        fig = plt.figure(figsize=(14,6))
        fig.suptitle('Subject:'+str(int(subid))+'   '+'Label:'+labelstr+'
'+'$P_{alert}=$'+str(round(likelihood[0],2))+'
$P_{drowsy}=$'+str(round(likelihood[1],2)))#, fontsize=12)
```

 # devide the figure layout

```
        gridlayout = gridspec.GridSpec(ncols=2, nrows=3, figure=fig,wspace=0.2, hspace=0.5)
        axis0 = fig.add_subplot(gridlayout[0:2,0])
        axis1 = fig.add_subplot(gridlayout[2,0])
        axis2 = fig.add_subplot(gridlayout[0:3,1])
```

 # do some preparations

```
        rawsignal=allsignals[sampleidx].cpu().detach().numpy().squeeze()
        channelnum=multichannelsignal.shape[0]
        samplelength=multichannelsignal.shape[1]
```

```
maxvalue=np.max(np.abs(rawsignal))
convkernelLength=self.model.kernelLength
```

 # calculate the heatmap for the sample

```
source = self.model.conv(allsignals)
source = self.model.batch(source)
source = torch.nn.ELU()(source)
activations=source[sampleidx].cpu().detach().numpy().squeeze()
weights=self.model.fc.weight[state].cpu().detach().numpy().squeeze()
cam=np.matmul(weights,activations)
heatmap=np.zeros(samplelength)
halfkerlength=int(convkernelLength/2)
heatmap[halfkerlength:(samplelength-halfkerlength+1)]=cam
for i in range(halfkerlength-1):
   heatmap[i]=heatmap[halfkerlength]*i/(halfkerlength-1)
for i in range((samplelength-halfkerlength),samplelength):
   heatmap[i]=heatmap[halfkerlength]*(samplelength-halfkerlength+1-i)/(halfkerlength)
heatmap= (heatmap-np.mean(heatmap)) / np.sqrt(np.sum(heatmap**2)/(samplelength))
```

 # calculate the band power components

```
psd, freqs = psd_array_multitaper(rawsignal, 128, adaptive=True,normalization='full', verbose=0)
freq_res = freqs[1] - freqs[0]
bandpowers=np.zeros(4)
idx_band = np.logical_and(freqs >= 1, freqs <= 4)
bandpowers[0] = simps(psd[idx_band], dx=freq_res)
```

```
idx_band = np.logical_and(freqs >= 4, freqs <= 8)
bandpowers[1] = simps(psd[idx_band], dx=freq_res)
idx_band = np.logical_and(freqs >= 8, freqs <= 12)
bandpowers[2] = simps(psd[idx_band], dx=freq_res)
idx_band = np.logical_and(freqs >= 12, freqs <= 30)
bandpowers[3] = simps(psd[idx_band], dx=freq_res)
totalpower=simps(psd, dx=freq_res)
if totalpower<0.00000001:
   bandpowers=np.zeros(4)
else:
   bandpowers /= totalpower
barx=np.arange(1, 5)
axis1.bar(barx,bandpowers)
axis1.set_xlim([0,5])
axis1.set_ylim([0,0.8])
axis1.set_ylabel("Relative power")
axis1.set_xticks([1,2,3,4])
axis1.set_xticklabels(['Delta','Theta','Alpha','Beta'])
```

 # draw the heatmap

```
xx= np.arange(1, (samplelength+1))
axis0.set_xticks([])
axis0.set_ylim([-maxvalue-10,maxvalue+10])
axis0.set_xlim([0,(samplelength+1)])
```

```
    axis0.set_ylabel("mV")
    points = np.array([xx, rawsignal]).T.reshape(-1, 1, 2)
    segments = np.concatenate([points[:-1], points[1:]], axis=1)
    norm = plt.Normalize(-1, 1)
    lc = LineCollection(segments, cmap='viridis', norm=norm)
    lc.set_array(heatmap)
    lc.set_linewidth(2)
    axis0.add_collection(lc)
    fig.colorbar(lc,ax=axis0,orientation="horizontal", ticks=[-1, -0.5, 0, 0.5, 1])
```

 # draw all the signals

```
    thespan=np.percentile(multichannelsignal,98)
    yttics=np.zeros(channelnum)
    for i in range(channelnum):
        yttics[i]=i*thespan
    axis2.set_ylim([-thespan,thespan*channelnum])
    axis2.set_xlim([0,samplelength+1])
    labels=['Fp1', 'Fp2', 'F7', 'F3', 'Fz', 'F4', 'F8', 'FT7', 'FC3', 'FCZ', 'FC4', 'FT8', 'T3', 'C3',
'Cz', 'C4', 'T4', 'TP7', 'CP3', 'CPz', 'CP4', 'TP8','T5', 'P3', 'PZ', 'P4', 'T6', 'O1', 'Oz','O2']
    plt.sca(axis2)
    plt.yticks(yttics, labels)
    heatmap1=np.zeros((channelnum,samplelength))-1
    heatmap1[-2,:]=heatmap
    xx=np.arange(1,samplelength+1)
    for i in range(0,channelnum):
        y=multichannelsignal[i,:]+thespan*(i)
```

```
            dydx=heatmap1[i,:]
            points = np.array([xx, y]).T.reshape(-1, 1, 2)
            segments = np.concatenate([points[:-1], points[1:]], axis=1)
            norm = plt.Normalize(-1, 1)
            lc = LineCollection(segments, cmap='viridis', norm=norm)
            lc.set_array(dydx)
            lc.set_linewidth(2)
            axis2.add_collection(lc)
def run():
    filename = r'dataset.mat'


    tmp = sio.loadmat(filename)
    xdata=np.array(tmp['EEGsample'])
    label=np.array(tmp['substate'])
    subIdx=np.array(tmp['subindex'])
    label.astype(int)
    subIdx.astype(int)
    samplenum=label.shape[0]
    channelnum=30
    classes=2
    subjnum=11
    samplelength=3
    lr=1e-2# for smalle net
    sf=128
```

```
batch_size = 50
n_epoch =6
ydata=np.zeros(samplenum,dtype=np.longlong)
for i in range(samplenum):
    ydata[i]=label[i]
selectedchan=[28]
rawx=xdata
xdata=xdata[:,selectedchan,:]
channelnum=len(selectedchan
```

 # We can set the subject id here

```
for i in range(2,3):
    trainindx=np.where(subIdx!= i)[0]
    xtrain=xdata[trainindx]
    x_train = xtrain.reshape(xtrain.shape[0],1,channelnum, samplelength*sf)
    y_train=ydata[trainindx]
    testindx=np.where(subIdx == i)[0]
    xtest=xdata[testindx]
    rawxdata=rawx[testindx]
    x_test = xtest.reshape(xtest.shape[0], 1,channelnum, samplelength*sf)
    y_test=ydata[testindx]
    train = torch.utils.data.TensorDataset(torch.from_numpy(x_train),
torch.from_numpy(y_train))
    train_loader = torch.utils.data.DataLoader(train, batch_size=batch_size, shuffle=True)
    my_net = CompactCNN().double().cuda()
    optimizer = optim.Adam(my_net.parameters(), lr=lr)
```

```
    loss_class = torch.nn.NLLLoss().cuda()
    for p in my_net.parameters():
        p.requires_grad = True
    for epoch in range(n_epoch):
        for j, data in enumerate(train_loader, 0):
            inputs, labels = data
            input_data = inputs.cuda()
            class_label = labels.cuda()
            my_net.zero_grad()
            my_net.train()
            class_output= my_net(input_data)
            err_s_label = loss_class(class_output, class_label)
            err = err_s_label
            err.backward()
            optimizer.step()
    my_net.train(False)
    with torch.no_grad():
        x_test =  torch.DoubleTensor(x_test).cuda()
        answer = my_net(x_test)
        probs=np.exp(answer.cpu().numpy())
        sampleVis =FeatureVis(my_net)
```

 # We can set the sample index here

```
        sampleidx=1
        sampleVis.generate_heatmap(allsignals=x_test,sampleidx=sampleidx,subid=i,
```

```
samplelabel=y_test[sampleidx],
multichannelsignal=rawxdata[sampleidx],
likelihood=probs[sampleidx])
if __name__ == '__main__':
    run()
```

#The training of the model and the detection of the eyes :

 # import the libraries

```
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
from tensorflow.keras.layers import Dropout,Input,Flatten,Dense,MaxPooling2D
from glob import glob
```

 # re-size all the images of the dataset to this

```
IMAGE_SIZE = [224, 224]
test_path = '/content/drive/MyDrive/dataset_prepared/test'
train_path = '/content/drive/MyDrive/dataset_prepared/train'
```

 # Import the inceptionV3 library and add preprocessing layer to the front of InceptionV3

 # using imagenet weights

```
inception = InceptionV3(input_shape=IMAGE_SIZE + [3], weights='imagenet',
include_top=False)

# never train existing weights

for layer in inception.layers:

        layer.trainable = False

# used for getting number of output classes

folders = glob('/content/drive/MyDrive/dataset_prepared/train/*')

# The layers

x = Flatten()(inception.output)

prediction = Dense(len(folders), activation='softmax')(x)

# create a model object

model = Model(inputs=inception.input, outputs=prediction)

# view the structure of the model

model.summary()

# Telling the model what cost and optimization method to use

model.compile(

  loss='categorical_crossentropy',

  optimizer='adam',

  metrics=['accuracy'])

# Using the Image Data Generator to import the images from the dataset

from tensorflow.keras.preprocessing.image import ImageDataGenerator


train_datagen = ImageDataGenerator(rescale = 1./255,shear_range = 0.2, zoom_range = 0.2,

horizontal_flip = True)
```

test_datagen = ImageDataGenerator(rescale = 1./255)

 # The same target size as initialied for the image size

training_set = train_datagen.flow_from_directory('/content/drive/MyDrive/dataset_prepared/train',target_size = (224, 224), batch_size = 32, class_mode = 'categorical')

test_set = test_datagen.flow_from_directory('/content/drive/MyDrive/dataset_prepared/test',

target_size = (224, 224), batch_size = 32, class_mode = 'categorical')

 # fit the model

 # Run the cell.

r = model.fit_generator(training_set, validation_data=test_set,epochs=10,steps_per_epoch=len(training_set), validation_steps=len(test_set))

 # save it as a h5 file

from tensorflow.keras.models import load_model

model.save('model_inception.h5')

#eye_aspect_ratio function

def eye_aspect_ratio(eye):

 # Calculate the euclidean distances between the two sets of the vertical eye landmarks (x, y)-coordinates :

A = dist.euclidean(eye[1], eye[5])

B = dist.euclidean(eye[2], eye[4])

 # compute the euclidean distance between the horizontal eye landmark (x, y)-coordinates

C = dist.euclidean(eye[0], eye[3])

 # compute the eye aspect ratio

ear = (A + B) / (2.0 * C)# return the eye aspect ratioreturn ear

Parsing command Line Argument

 # construct the argument parse and parse the arguments

```
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,
help="path to facial landmark predictor")
ap.add_argument("-a", "--alarm", type=str, default="",
help="path alarm .WAV file")
ap.add_argument("-w", "--webcam", type=int, default=0,
help="index of webcam on system")
  args = vars(ap.parse_args())
```

#Defining EYE_AR_THRESH

```
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 48
COUNTER = 0
ALARM_ON = False
```

#Facial landmark predictor

 # initialize dlib's face detector (HOG-based) and then create the facial landmark predictor

```
print("[INFO] loading facial landmark predictor...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])
```

#Extracting the eye regions

```
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
```

 # start the video stream thread

```
print("[INFO] starting video stream thread...")
```

```
vs = VideoStream(src=args["webcam"]).start()
time.sleep(1.0)
```

 # loop over frames from the video stream

```
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

 # detect faces in the grayscale frame

```
rects = detector(gray, 0)
```

#Facial landmark detection to localize each of the important regions of the face:

 # loop over the face detections, determine the facial landmarks for the face region, then convert the facial landmark (x, y)-coordinates to a NumPy array

```
for rect in rects :
    shape = predictor(gray, rect)
    shape = face_utils.shape_to_np(shape)
```

 # extract the left and right eye coordinates, then use the coordinates to compute the eye aspect ratio for botheyes

```
leftEye = shape[lStart:lEnd]
rightEye = shape[rStart:rEnd]
leftEAR = eye_aspect_ratio(leftEye)
rightEAR = eye_aspect_ratio(rightEye)
```

 # average the eye aspect ratio together for both eyes

```
ear = (leftEAR + rightEAR) / 2.0
```

#Visualize each of the eye regions

 # compute the convex hull for the left and right eye, then

```
leftEyeHull = cv2.convexHull(leftEye)
rightEyeHull = cv2.convexHull(rightEye)
cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0),1)
cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0),1)
```

#Check to see if the person in our video stream is starting to show symptoms of drowsiness:

 # check to see if the eye aspect ratio is below the blink# threshold, and if so, increment the blink frame counter

```
if ear < EYE_AR_THRESH:
    COUNTER += 1
```

 # if the eyes were closed for a sufficient number of# then sound the alarm

```
if COUNTER >= EYE_AR_CONSEC_FRAMES:
```

 # if the alarm is not on, turn it on

```
if not ALARM_ON:
    ALARM_ON = True
```

 # check to see if an alarm file was supplied,and if so, start a thread to have the alarm sound played in the background

```
if args["alarm"] != "":
    t = Thread(target=sound_alarm,
    args=(args["alarm"],))
    t.deamon = True
    t.start()
```

 # draw an alarm on the frame

```
cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
  cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

 # otherwise, the eye aspect ratio is not below the blink threshold, so reset the counter and alarm

```
        else:
            COUNTER = 0
            ALARM_ON = False
```

#Displaying the output frame:

 # draw the computed eye aspect ratio on the frame to help with debugging and setting the correct eye aspect ratio thresholds and frame counters

```
        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

 # show the frame

```
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
```

 # if the `q` key was pressed, break from the loop

```
        if key == ord("q"):
            break
```

 # do a bit of cleanup

```
        cv2.destroyAllWindows()
        vs.stop()
```