BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

EEG CLASSIFICATION MODEL FOR EMOTION DETECTION USING PYTHON

EEG CLASSIFICATION MODEL FOR EMOTION DETECTION USING PYTHON

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **VERONIKA VENGEROVÁ**
AUTOR PRÁCE

SUPERVISOR **Dr. SOYIBA JAWED, Msc**
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

143556

Institut:	Department of Computer Systems (UPSY)
Student:	**Vengerová Veronika**
Programme:	Information Technology
Specialization:	Information Technology
Title:	**EEG Classification Model for Emotion Detection Using Python**
Category:	Signal Processing
Academic year:	2022/23

Assignment:

1. Study and learn about the various emotions.
2. Get acquainted with feature extraction and selection, classification methods, and their application to the detection of emotions.
3. Find out the challenges for emotion detection and the limitations of the existing methods.
4. Design a model for feature extraction and feature selection, and classification of emotion detection.
5. Implement and evaluate the designed model.
6. Conduct critical analysis and discuss the achieved results and their contribution.

Literature:

- Based on the supervisor's recommendation.

Requirements for the semestral defence:

- Fulfillment of Items 1 to 3 of the assignment.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Jawed Soyiba, Dr., MSc**
Head of Department:	Sekanina Lukáš, prof. Ing., Ph.D.
Beginning of work:	1.11.2022
Submission deadline:	10.5.2023
Approval date:	31.10.2022

Abstract

This thesis deals with the task of recognizing emotions from electroencephalogram (EEG). Two models were trained for binary classification of emotions, where one classifies neutral emotion or fear and the other classifies happiness or sadness. During the work on this thesis many different architectures were tried, and the best result was obtained using a model with two branches of CNN-LSTM connected before the output layer. The resulting accuracy was 87.309% for sad-happy classification and 84.865% for neutral-fear emotion.

Abstrakt

Táto práca sa zaoberá rozoznávaním emócií z elektroencefalogramu (EEG). Dva modely na binárnu klasifikáciu emócií, kde jeden model klasifikuje neutrálnu emóciu alebo strach a druhý šťastie a smútok. Počas práce boli vyskúšané mnohé rôzne architektúry, pričom najlepšie výsledky boli dosiahnuté modelom pozostávajúcim z dvoch vetiev KNN-LSTM spojenými pred výstupnou vrstvou. Výsledná presnosť bola 87.309% na klasifikáciu šťastia a smútku a 84.865% na klasifikáciu neutrálnej emócie a strachu.

Keywords

deep learning, machine learning, EEG, Electroencephalography, emotion recognition, emotion classification, LSTM, GRU, CNN, convolutional neural networks, recurrent neural networks, raw EEG

Klíčová slova

hlboké učenie, strojové učenie, EEG, Elektroencefalografie, rozoznávanie emóciií, klasifikácia emócií, LSTM, GRU, CNN, konvolučné neurónové siete, rekurentné neurónové siete, nespracované EEG

Reference

VENGEROVÁ, Veronika. *EEG Classification Model for Emotion Detection Using Python.* Brno, 2023. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Dr. Soyiba Jawed, Msc

Rozšířený abstrakt

S emóciami sa stretávame každý deň. Emócie sú odozvou na externé a interné podnety ovplyvňujúce ľudské správanie, pamäť, motiváciu, rozhodovanie, sústredenie, kreativitu a medziľudské vzťahy. Emócie možno rozoznať pomocou výrazu tváre alebo slovných vyjadrení, no ľudia môžu emócie maskovať alebo neprejavovať, čo môže sťažovať prácu napríklad psychológom. Rozoznávanie emócií priamo z mozgu je preto atraktívny smer rôznych štúdií.

Táto práca sa zaoberá rozoznávaním emócií zo záznamov Elektroencefalografie (EEG). Na rozpoznanie emócií sa môžu využívať rôzne metódy ako strojové učenie alebo "hlboké učenie". Pre dosiahnutie čo najlepšieho rozonávania emócií bolo potrebné preštudovať základné koncepty funkcie neurónov, ľudkého mozgu a zachytávanie mozgovej aktivity pomocou EEG. Ďalej bolo nutné zoznámiť sa s procesom učenia v neurónových sieťach a rôzynmi existujúcimi architektúrami. V práci sú potrebné informácie na pochopenie týchto súvislostí poskytnuté, pričom väčší dôraz sa kladie na vysvetlenie princípov hlbokého učenia.

Cieľom práce bolo natrénovať model rozoznávajúci emócie. V tejto práci sa využíva metóda "hlbokého učenia" z nevyčistených segmentovaných EEG dát bez predspracovania. Tento prístup bol zvolený s cielom neurónovej siete rozoznať globálne vzory predstavujúce emócie priamo z mozgu. EEG dáta využívané v tejto práci sú z SEED-IV datasetu. Jednotlivé EEG záznamy boli rozdelené podľa poskytnutých tried a segmentované po 1-sekundových častiach. Dataset obsahuje EEG záznamy zamerané na štyri emócie – smútok, neutrálnu emóciu, šťastie a strach. Toto umožnilo aby vznikli dva binárne modely na rozoznávanie strachu alebo neutrálnej emócie a šťastia alebo smútku z EEG dát.

Počas práce bolo vyskúšaných viacero architektúr pozostávajúcich z LSTM, GRU a CNN vrstiev, ktoré sú často využívané na rozoznávanie emócií. Jednotlivé modely využité v práci sú bližsie vysvetlené v práci, pričom jedným modelom boli dosiahnuté výrazne lepšie výsledky ako ostatnými.

Model ktorý dosiahol najlepšie výsledky pozostával z dvoch vetiev, kde každá vetva pozostáva z konvolučnej vrsty po ktorej nasleduje aktivačná vrstva "elu", ďalšie dve konvolučné vrstvy, aktivačná "elu" funkcia a na záver "MaxPooling". Jednotlivé vrstvy sa líšili veľkosťou jadra konvolúcie, pričom jedna vetva mala veľkosť jadra 5 a druhá 7.

Model bol vyhodnotený na podmnožine SEED-IV datasetu na ktorom nebol trénovaný. Na týchto pre model nových dátach bola dosiahnutá presnosť (po anglicky "accuracy") klasifikácie šťastia alebo smútku 88.43% a klasifikácie neutrálnej emócie alebo strachu 84.865%. Pre model boli taktiež vypočítané a pozorované iné hodnotiace metriky, pričom AUC skóre hovoriace o tom ako dobre vie model rozlišovať jednotlivé triedy bolo viac ako 0.92 pre oba modely.

EEG Classification Model for Emotion Detection Using Python

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Dr. Soyiba Jawed, Msc. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Veronika Vengerová
May 9, 2023

Acknowledgements

I would like to thank the thesis supervisor Dr. Soyiba Jawed, Msc, for her guidance during the writing of the thesis. Despite her being busy, she always found the time to answer any questions or problems I raised. I would also like to thank her for the opportunity to work on such an interesting topic.

Contents

List of Figures

Chapter 1

Introduction

Emotions are a popular topic since the times of ancient philosophers. The understanding of emotions and emotion theories as such differ through time. Some believed that the human body actions lead to the emotions being formed, some that the thoughts and mental state are the main reason for the emotions to form, while other believed that the activity in human brain creates the emotions. It can be seen that humans always wanted to better understand emotions.

Emotions are organized responses typically to internal or external stimuli that influence person's memory, decision making, task prioritization, focus, creative thinking, planning for future, motivation and inter-human relationships. Psychologists can use the emotion recognition of their patients in order to create more accurate diagnosis and help with treatment. Another use case for recognizing emotions is user interface, where the emotional response from the user in human-computer interfaces can be used for improvement. Currently there are two models of emotions utilized in affect computing: dimensional and discrete models.

Emotions are in dimensional models represented in a dimensional form. A two-dimensional model has valence (ranging from positive to negative) as one dimension and arousal (ranging from boredom to excitement) as the second dimension called VA model. Three dimensional model with valence, arousal and dominance (ranging from no control to full control over situation) dimensions called VAD model. Discrete or categorical emotion models consider that humans feel the primary emotions such as fear, sadness, joy, disgust, surprise and anger and labels them. People then choose one of the emotion from the model.

Emotions can be recognized by speech, the tone of voice, by facial expressions, body language etc. However, as the emotions recognized by people can be subjective or the emotion can be masked, the emotion recognition through brain waves using EEG is recently getting more frequently explored. The main idea behind EEG emotion recognition is to recognize the brain patterns representing emotions from brain.

EEG captures the activity of the brain, which is a signal with complex dependencies. The patterns representing emotions are not recognizable by eye so more advanced methodologies are used to interpret the brain waves. Machine learning and it's sub-field deep learning are often chosen methodologies for emotion recognition with good evaluation metrics. The goal of this thesis is to train and create a deep learning model capable of recognizing emotions.

This thesis is organized into five parts. In chapter 2 all necessary information to understand how the brain works are provided. The following chapter 3 explains the electroencephalography (EEG) and how it manages to capture the brain activity. In chapter 4 the concepts of deep learning are introduced. Chapter 5 presents the dataset used in this

thesis and the designed models used in this thesis are introduced. The following chapter 6 discusses the results achieved.

Chapter 2

Introduction to brain activity for EEG

In order to understand how emotion can be recognized from the data, an insight into how the brain works and what brain activity is provided in this chapter.

Human brain's neural activity starts in the prenatal development between the 17th and 23rd week, which is an indicator of the brain functions and the state of the body [25]. The most significant functional unit of the central nervous system is neuron.

2.1 Neurons

In this section the information obtained from *The Human Brain Book: An Illustrated Guide [6]* and *EEG/ERP Analysis: Methods and Applications [21]* is summarized into a compact format.

A neuron is a self-contained functioning unit of the central nervous system. The human brain contains billions of neurons, which process and transmit information through electrical and chemical signals.

Neurons consist of a cell body (*soma*) and long, thin, threadlike extensions coming out of its core – *neurites*. Neurites can be mainly of two types – *axons* and *dendrites*. Usually, dendrites receive nerve signals, and axons send them further. Nerve signals compose of series of discrete impulses – action potentials. A nerve impulse is a tiny spike of electricity traveling through the neuron. When looked at in more detail, it is caused by the movement of the ions (chemical particles with electrical charges) through the cell's outer membrane surrounding the neuron. The nerve signal moves along the membrane as a sequence of depolarization and repolarization.

When no impulse is passing through the neuron, the neuron has a resting potential of -70mV. In this state there are more potassium and negatively charged ions inside the cell body and more sodium and positive ions outside the cell's membrane. This causes the cell's outside to have positive electrical potential and inside the cell's body to have negative electrical potential – polarization. When an electrical current arrives through dendrites, the potential is slowly increasing until a threshold of approximately -55mV is reached(A). This process of increasing the potential is called depolarization. After the threshold of -55mV is reached, the cell membrane is completely open to the positively charged sodium ions flowing from the outside of the cell membrane to the inside. After the cell is completely open to the Na+ ions, a positive action potential of approximately 30mV is reached (B).

After reaching the peak of 30mV, restoring the balance of the electrical charge of the neuron's body is needed. The positive potassium ions are pumped out across the membrane to restore the resting potential (C). This process is also called repolarization. During this change, the cell membrane potential drops below the resting potential (D), also known as hyper-polarization, and then increases to the cell membrane's resting potential of -70mV. Action potential (a sequence of changes in the voltage across the cell's membrane) is carried to the other neurons through the axons. As the neuron has many axons and dendrites, the neuron becomes activated when the total electrical current incoming from all axons exceeds a certain threshold, and the information is transmitted further.

The strength and duration of most nerve impulses are the same – around 100 millivolts for strength and around one millisecond for the duration. The speed of the impulse might vary.

As the electrical current generated by a single neuron is small, the neural oscillations generated by large groups of neurons are usually monitored to observe the central nervous system.

2.2 Brain waves

Neural oscillations that electroencephalography investigates are called brain waves. The human brain can produce five major brain waves that are classified by frequency ranges.

Delta waves have a frequency range of 0.5-4.0 Hz. These waves have the highest amplitude, occur primarily in adults and are associated with deep sleep [21].

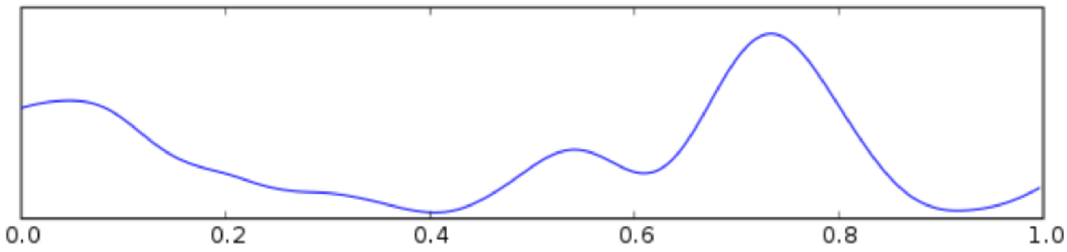


Figure 2.1: Figure depicting delta wave. Taken from *EEG/ERP Analysis: Methods and Applications 1st Edition* [21].

Theta waves (4 – 8 Hz) are normally observed in young children. In adults, these waves are observed in a state of drowsiness, arousal, or meditation. Their excess can indicate abnormal activity.

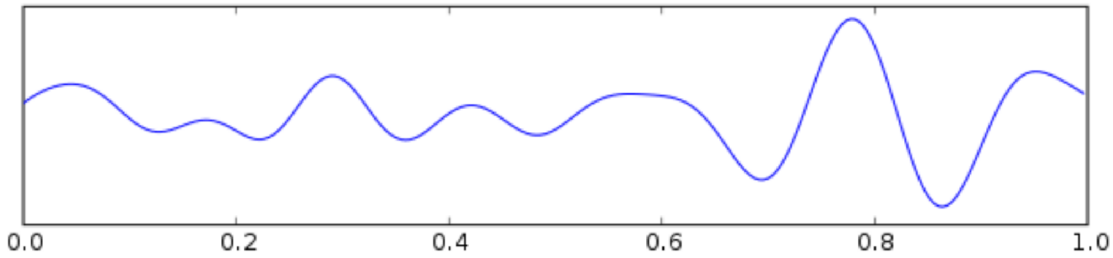


Figure 2.2: Figure depicting theta wave. Taken from *EEG/ERP Analysis: Methods and Applications 1st Edition* [21].

Alpha waves (8 – 13 Hz) are associated with wakefulness, closing the eye and creativity. Alpha waves have higher amplitudes near the occipital areas. The name alpha waves comes from the fact that these were the first waves that were detected.

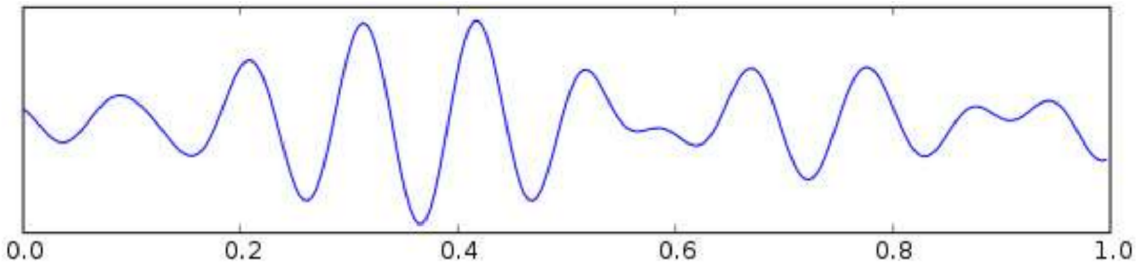


Figure 2.3: Figure depicting alpha wave.Taken from *EEG/ERP Analysis: Methods and Applications 1st Edition* [21].

Beta waves (14 -26 Hz) are found in normal adults. They represent active attention, active thinking, focus on the outside world, and solving concrete problems. They are usually of low amplitude under 30 microvolts.

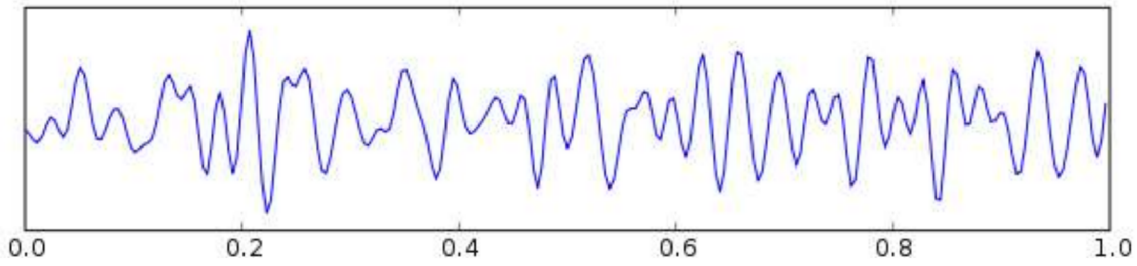


Figure 2.4: Figure depicting beta wave.Taken from *EEG/ERP Analysis: Methods and Applications 1st Edition* [21].

Gamma waves (30-100 Hz) help bind the neurons' populations together. They rarely occur, only when simultaneously processing more senses, such as sound and sight.

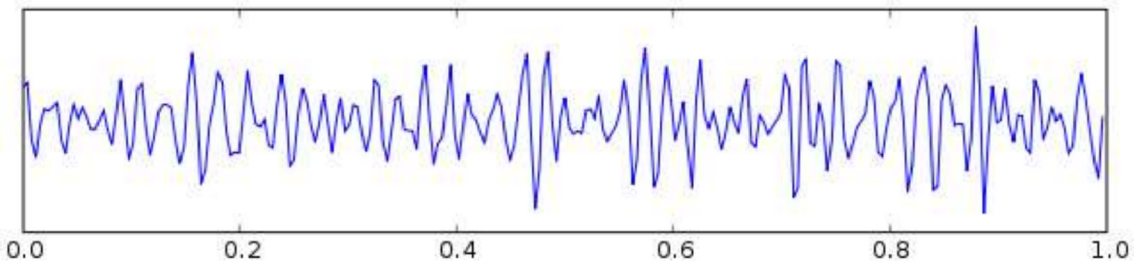


Figure 2.5: Figure depicting gamma wave.Taken from *EEG/ERP Analysis: Methods and Applications 1st Edition* [21].

2.3 The human brain

Different types of brain waves and various parts of the brain are associated with distinct functions. The human brain consists of three main parts: the cerebellum, cerebrum, and the brainstem:

Brainstem
Brainstem is involved in automatic mental activities such as breathing, heartbeat, vomiting, sneezing, and coughing [6].

Cerebellum
Cerebellum is located at the back of the head and is also called the little brain. It's function is to coordinate body movements such as keeping balance, keeping posture, and muscle control [6].

Cerebrum
Cerebrum is the largest part of the brain and is divided into left and right hemispheres linked with nerve fibers [6]. Its function is responsible for day-to-day functions. Cerebrum is subdivided into six parts: The frontal lobe, Parietal Lobe, Temporal Lobe, Occipital Lobe, Insular Lobe, and Limbic Lobe.

- **Frontal lobe**
  Frontal lobe is responsible for executive functions such as emotion control, movement, speech, and problem-solving [6][21].
- **Temporal lobe**
  Temporal lobe is responsible for auditory sensations processing and their emotional tone [6][21].
- **Parietal lobe**
  Parietal lobe perceives pain, taste sensation and is active during problem solving [21].
- **Occipital lobe**
  Occipital lobe is mainly responsible for vision-related tasks and information processing [21].
- **Insular lobe**
  Insular lobe assumes role in pain transmission [6].

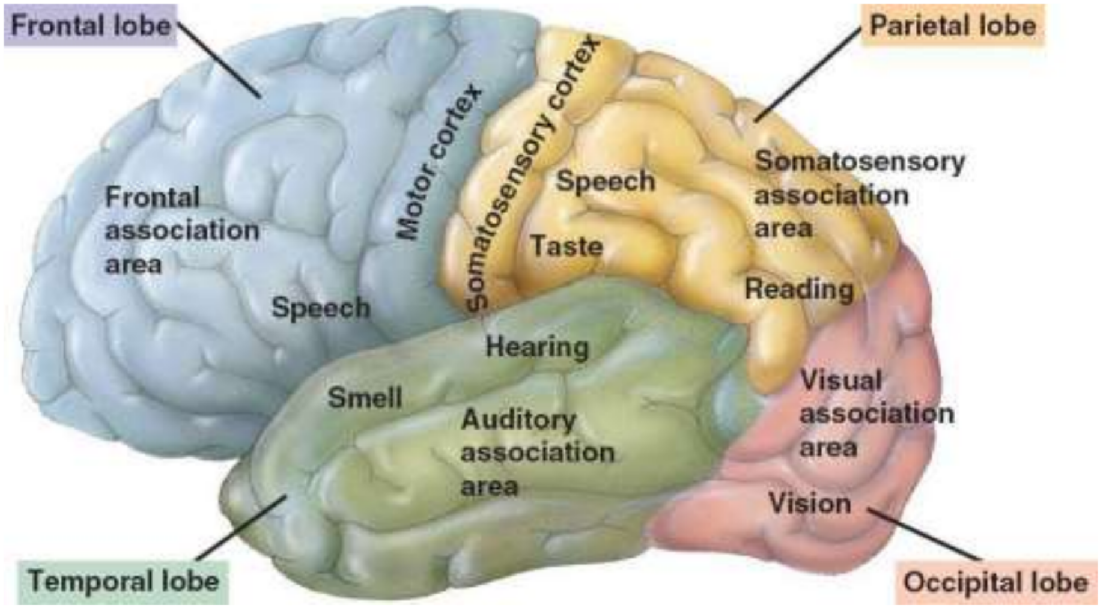


Figure 2.6: Figure depicting the different parts of brain and their functionality. Taken from *EEG/ERP Analysis: Methods and Applications 1st Edition* [21].

Chapter 3

Electroencephalography

In this chapter, information was mainly obtained from book *EEG/ERP Analysis: Methods and Applications 1st Edition* [21].

Electroencephalography investigates brain waves (neural oscillations). The first person to discover brain activity in the form of electrical signals in small animals was Richard Caton (1877). Vladimir Pravdich-Neminsky was the first to measure the electrical activity in the brain of dogs in 1912. V. Pravfich-Neminsky used the term electrocerebrogram to describe this recording. Later, Hans Berger recorded the first human electroencephalography recordings (1929). Berger named the brain's electrical activities „Electroenkephalogram". Since the first brain activity recording, the concepts of recording electrical neural activity were improved and combined into electroencephalography. The name Electroencephalogram stands for *electro*, which refers to the brain's electrical activities, *encephalo* referring to signals emitting from the head, and *gram* for drawing or writing of the signal [25]. The abbreviation EEG is commonly used instead of the full name electroencephalogram and will be used interchangeably in the future in this work. Electroencephalogram is the recording of the electrical activity generated by the firing of the neurons within the brain, usually recorded at the scalp [14]. The electrical impulse generated by a single neuron is too small for EEG is to capture. EEG records the sum of the synchronous activity of a big group of neurons (thousands of millions) with similar spatial orientation. The signal has to go through multiple resistive layers during EEG recording, and the voltage of two electrodes close together can be quite similar. The activity deep in the brain is, therefore, difficult to be captured by the EEG. EEG has low spatial resolution (where in the brain activity happened) and good temporal resolution (when the brain activity happened).

EEG can be recorded either by non-invasive or invasive methods. During invasive EEG, doctors during surgery implant the electrodes in some regions of the brain. During non-invasive EEG, the electrodes are placed on the scalp. Electrodes can be placed on the scalp by using special headcaps or headsets. During a non-invasive EEG recording, a gel or saline can be used. Gel and saline improve electric conductivity and impedance, resulting in better-quality recordings. As using gel or saline is not always convenient, headcaps and EEG recording systems that do not need gel or saline and still obtain satisfactory results were invented and are often used.

The EEG measurement system consists of electrodes, amplifiers with filters, an analog-to-digital converter, and a recording device.

3.1 Electrodes

Electrodes are small metal disks typically of a diameter from 0.4cm to 1 cm. There are many types of electrodes, and the most commonly used are Ag-AgCl electrodes made of silver and covered with a silver chloride coating.

In general, electrodes can be classified into three types according to their roles in the recording – active, reference, and ground electrodes. Ground electrodes reduce the noise by grounding the circuit. The reference electrode is chosen as a reference point for capturing the electrical charges between the active electrodes during the recording. The reference electrode is chosen based on the montage used for recording, to understand the montage types it is helpful to understand the placements of electrodes first.

3.2 Electrode Placement

In 1958 a standard EEG electrode placement protocol was formulated, later revised and called the 10-20 System. This system informs about the locations of 75 electrodes on the scalp's surface. At first, the nasion (front of the head), the inion (back of the head), and two pre-auricular points are determined. Based on these locations, the placement of the other electrodes is determined. The electrodes are placed at positions with distances of 10% and 20% along the front-back length (nasion-inion) and left-right length. That is why the international name for this electrode placement is 10-20 System. Each electrode has its specific name to be able to identify its position on the scalp. The name consists of one or two letters and a number. Letters represent the brain region of the position of the electrode where **Fp** stands pre-frontal, **F** for frontal lobe, **C** for central, **P** for parietal, **O** for occipital and **T** for temporal. Numbers represent the distance from the midline (line from the nasion to the inion), where the higher the number, the greater the distance. Midline electrodes are labeled with the letter 'z'. Odd numbers are used for the notation of electrodes placed on the left hemisphere and even numbers on the right hemisphere.

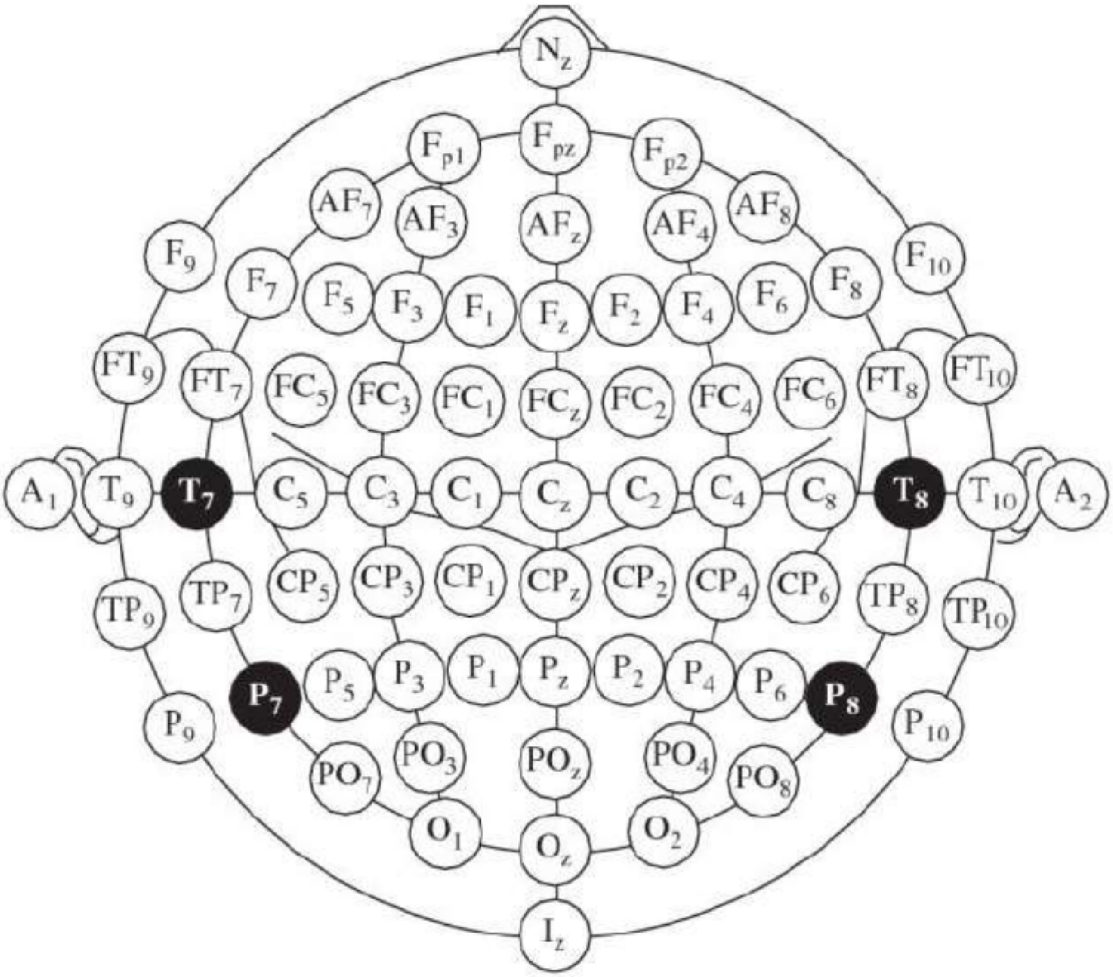


Figure 3.1: Figure depicting the 10-20 system. Taken from *EEG/ERP Analysis: Methods and Applications 1st Edition* [21].

3.3 Montages

There are two main montage types: bipolar and referential.

Bipolar montage

In a bipolar montage, a difference between two neighboring electrodes is measured. Channels are arranged in chains where the first channel's second electrode is the next channel's first electrode. So the channel capturing the difference in voltage between the electrode Fp1 and electrode F3 is followed by the channel capturing the voltage difference between the electrodes F3 and C3, and so on. In this type of montage, the names for the channels consist of the electrode names such as Fp1-F3 and F3-C3. This montage type easily cancels out external noises as it amplifies the local potential.

3.3.1 Referential montage

In a referential montage, each channel in the recording represents the difference between an active electrode and one chosen referential electrode. This reference electrode is placed in a different position than the recording electrodes but has no standard position. If the referential electrode is placed on one hemisphere, it can cause the amplification of the signal on this hemisphere. One of the places the reference position that could be selected is ears.

3.3.2 Average reference montage

Another montage type is the average reference montage. A signal is obtained by summing and averaging the outputs of all amplifiers, which is then used as a common reference signal for each channel.

3.4 Artifacts in EEG

Data obtained from EEG recording is composed of EEG recording and noises that are not necessarily related to brain activity. All non-EEG signals are referred to as artifacts [1]. Artifacts can be divided into 2 categories: Physiological and non-physiological [1]. Physiological artifact are artifacts originated from the body such as movement of the eye, eye movement, head movement, heartbeat or muscular noise.

Physiological artifacts

Physiological artifacts can be removed from EEG using the recordings capturing the signals from Electrooculograp (EOG) for vision-related stimuli, Electromyogram (EMG) for muscle contraction, and Electrocardiogram (ECG) for potential difference in the cardiac muscle. Ocular-related artifacts sum linearly on top of the brain activity. They can be relatively easy to detect as they appear in EEG as one big spike during blinking or a rectangular-like spike during eye movement. These artifacts also occur only near the occipital and frontal regions.

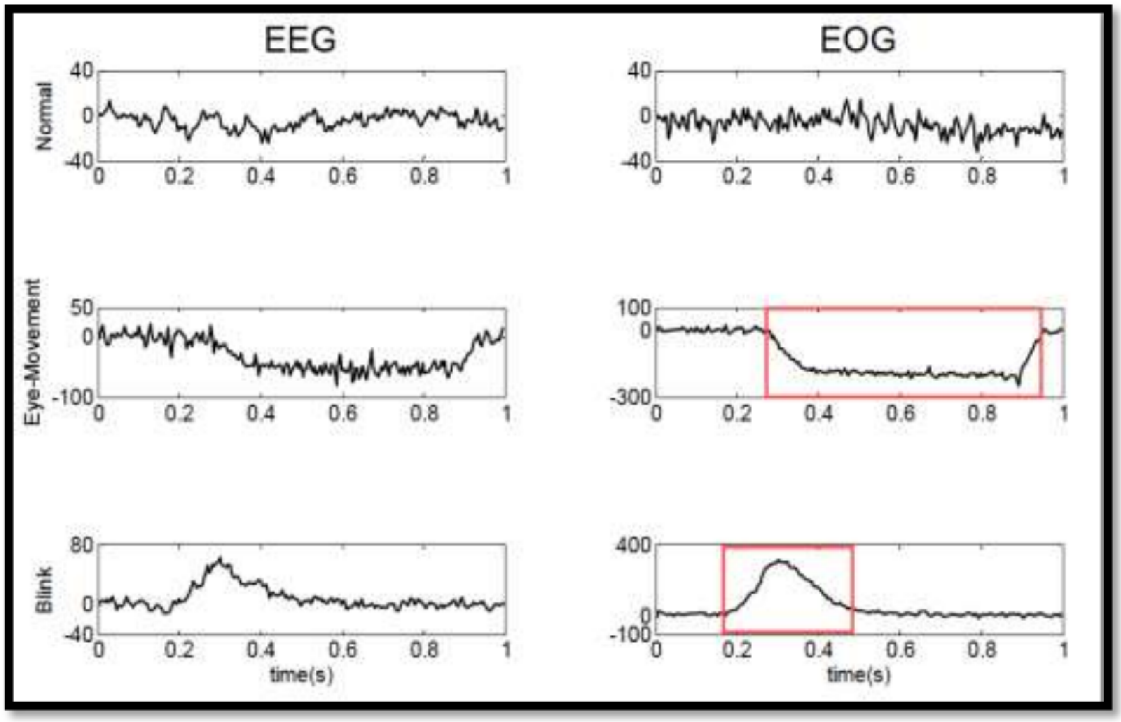


Figure 3.2: Figure showing EEG recording with eye blink artifact and eye movement artifact (left) with corresponding EOG recording(right). Taken from *EEG/ERP Analysis: Methods and Applications 1st Edition* [21].

Electromyography artifacts are known as high-frequency activities. These artifacts are generated by the contraction of the muscles representing neuromuscular activities. The signal appears to be very spiky when plotting the captured electrical amplitudes with electromyography artifacts as can be seen in Figure 3.3.

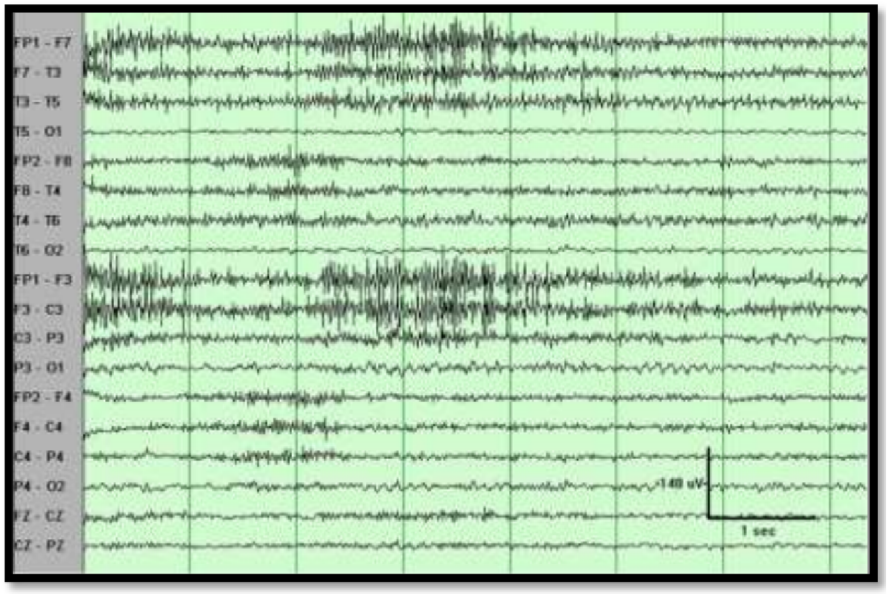


Figure 3.3: Figure showing EMG artifact in EEG recording. Taken from *EEG/ERP Analysis: Methods and Applications 1st Edition* [21].

ECG artifacts occur during heart activity can be seen in Figure 3.4. The location of the ECG artifact and the magnitude varies from subject to subject based on the width of the person's neck.

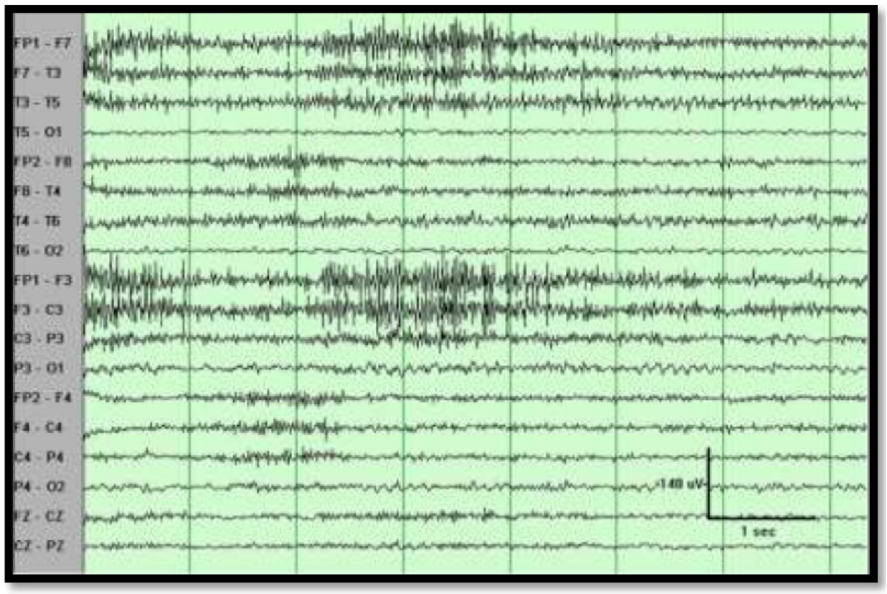


Figure 3.4: ECG artifact in EEG recording(blue), corresponding ECG signal (green) and the signal after ECG was suppressed (red).Taken from *EEG/ERP Analysis: Methods and Applications 1st Edition* [21].

Non-physiological artifacts

Non-physiological artifacts can be caused by the surrounding environment, such as DC noise, high impedance, cable movement, poor placement of electrodes, or EEG recording system malfunction. One of the most common causes is the alternating power supply which varies in different continents/countries. This frequency is 50 Hz in Europe and in the USA 60 Hz, which can be removed by applying a notch, which is a filter that attenuates frequencies of some range, but others pass unaltered.

3.5 EEG data preprocessing

This subchapter has been adopted from[14]. EEG recordings can contain different noises called artifacts that can affect the recorded signal as was already explained in section 3.4. The usual way to process the EEG data for a machine learning algorithm is to clean the data using preprocessing before handcrafting the features from the EEG.

There are many methods to preprocess the EEG data. Preprocessing usually consists of segmenting the raw data without change (reorganization) and removing artifact-ridden data from EEG recording. Frequently used EEG preprocessing methods are filtering, extracting data epochs and removing baseline values, removing and interpolating bad channels, removing bad epochs, or removing EEG artifacts using independent component analysis.

This preprocessing is not needed for deep neural networks that is learning from the raw data as in deep learning the neural networks learn from the noise in the data, and with more advanced preprocessing, the global patterns can be smoothed out and removed from the data. Usually if the raw data are passed to the neural network the only preprocessing is segmenting the data without any change to the data itself.

There are however methods that can be used for preprocessing that change the original values of the data which will be explained in the next sections and are not used in this thesis.

Filtering

It is advised to use filtering as the first preprocessing method as it can cause filtering artifacts on the already segmented data. Digital filters can be applied to raw EEG data to reduce the artifacts. Four types of filters are usually used: low-pass filter, high-pass filter, band-pass filter, and band-stop filter.

Extracting data epochs and removing baseline values

Sometimes it is necessary to change two-dimensional EEG data (timesteps×channels) to three-dimensional data (epochs[1] × segment timesteps × channels). This can be necessary if some cognitive tasks want to be observed with marked events in the EEG recording. During the segmenting of EEG data based on these tasks choosing the event onset is important. Event onset is the time point 0 which can be either the behavior response to the stimulus or the stimulus. After the segmentation of EEG data where event-related potentials are observed, it is necessary to remove the mean baseline value from each segment as the electrical potential at each segment differs. During resting-state datasets, where

[1]Epochs in this context mean the number of segments created from the EEG data

a neural activity not elicited from a specific task is observed[4], the baseline correlation is not necessary even after segmenting the data.

Removal and interpolation of bad channels

Many mistakes might occur during the recoding of EEG, such as electrode malfunctioning or improperly placed electrodes. Channels with malfunction can be called „bad channels“. These bad channels might be removed as part of the preprocessing not to cause noise to the data. However, which channels are considered to be bad can differ from subject to subject during EEG recording, which makes preprocessing the data for a group of subjects difficult. Interpolation of the bad channels based on the good channels is often used instead. However, this lowers the spatial resolution as the bad channel is dependent on the rest of the channels after interpolation.

The removal of bad epochs

If a segment of the EEG recording contains too many artifacts, such as eye movements or giggling, the epoch might be rejected. Rejection of a segment is usually done after visual inspection and manual flagging of the bad epochs. If observing event-related tasks, the epochs where the subject made an error response might also be removed. Experts may disagree on whether these epochs should be removed from the datasets.

Removal of EEG artifacts using independent component analysis

Independent component analysis (ICA) is a statistical method for decomposing EEG data into a set of components. This method separates components from multichannel EEG signals into independent signals. This can be used for artifact removal after identifying the component related to the artifacts.

Chapter 4

Deep learning

Mitchell created a broad definition for the term learning algorithm:

> "*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E [20].*"

This thesis is concerned with a supervised machine-learning classification task. In reference to the definition of machine learning quoted above, the task T in this thesis is the task of classifying emotions from EEG, the performance measure P is the percentage of correctly classified EEG recordings, and the experience E is a database of EEG labeled samples.

This chapter introduces neural networks and their application in the recognition of emotions from EEG. It consists of an introduction to the pipeline of EEG recognition, what neural networks are and how they learn, and the special neural network architectures of neural network architectures that can be used for EEG classification tasks.

A typical machine learning pipeline for processing EEG data for recognizing emotions is:

1. Obtain or record EEG data
2. Preprocess the EEG data
3. Extract and select features
4. Pass the features to a classifier

Feature extraction and feature selection identify smaller feature sets from raw EEG data that are then passed to a model to reduce the time needed for model training and increase the accuracy.

Time domain analyses (such as event-related potential, independent component analysis, or principal component analysis), frequency domain analyses (such as power spectral density, higher-order spectrum, or fast Fourier transform), time-frequency domain analyses (such as short-time Fourier transform or wavelet transform) or nonlinear analyses (such as permutation entropy, approximate entropy, power spectrum entropy) can be used to extract features from the EEG data [13].

Selecting features can also reduce the number of features. Features can be selected using linear discriminant analysis, principal component analysis, independent component analysis, or other methods. This process of extracting and selecting features can be laborious, and machine learning practitioners need to have thorough understanding of the EEG

data to extract the correct features. Deep learning algorithms extract and select features automatically and globally as they generate patterns from noise.

The pipeline changes to:

1. Obtain or record EEG data
2. Preprocess the EEG data
3. Train the neural network

The visualization of the pipeline for EEG classification is in Figure 4.1. After the prediction an estimate of how good the model is performing is usually computed and based on this further improvements to the model are done.

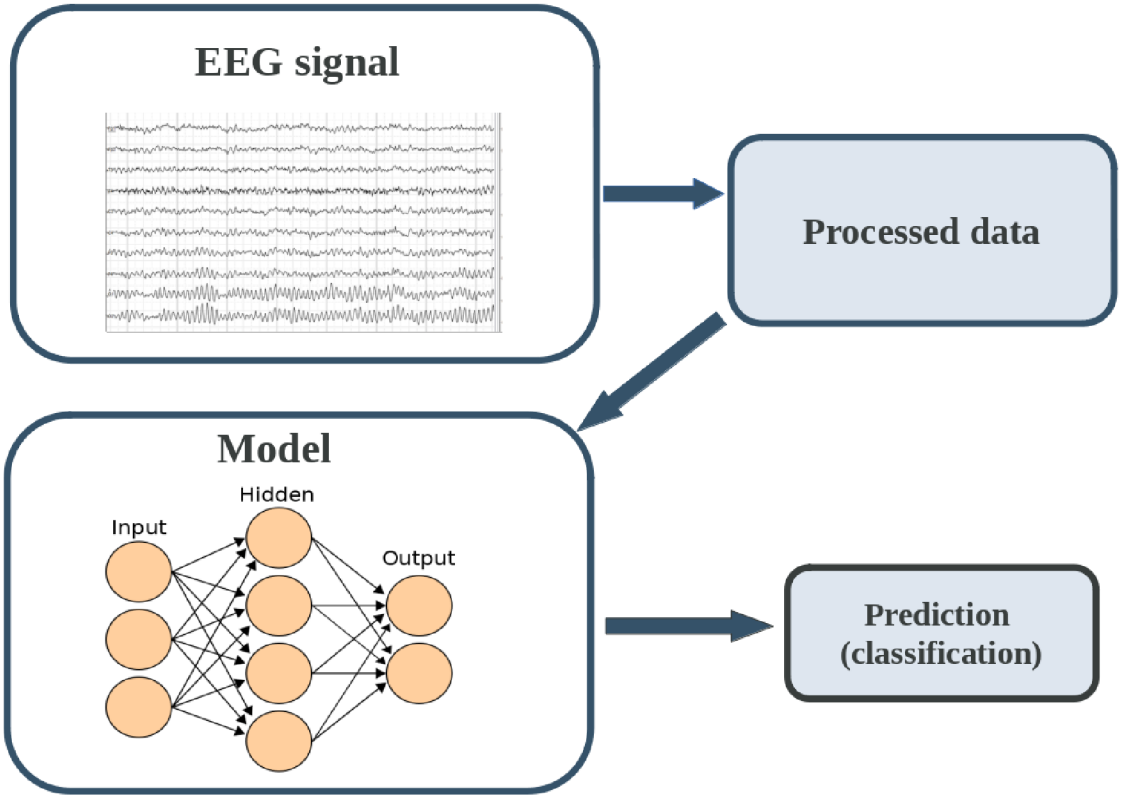


Figure 4.1: Figure depicting the pipeline for EEG classification using deep learning. Picture for neural network is taken from [2].

In the next section, the term neural network will be explained.

4.1 Neural networks

The information in this chapter is mainly based on *Deep learning illustrated book*[18] and *Fundamentals of Deep Learning*[5]. Multiple artificial neurons combined and assembled into layers are called neural networks. When using non-linear functions as activation functions, neural networks are able to compute any continuous function [22]. The term deep in neural networks refers to the depth of the neural network (the number of layers). Neural network

with few layers is called a shallow neural network and when a neural network has many layers and artificial neurons, then the network is called deep neural network. In neural network a layer is a container of neurons also called units in the same depth. In a broad sense the same depth neurons can be the neurons whose input is the same or number of operation computed to get the input to the neuron is the same [12]. For example when the neural network has 2 layers, the first layer of neurons was given the input vector $\mathbf{x}$, the second layer of neurons has as the input the activation values from the neurons from the previous layer (columns in the Figure 4.2 represent one layer of a neural network).

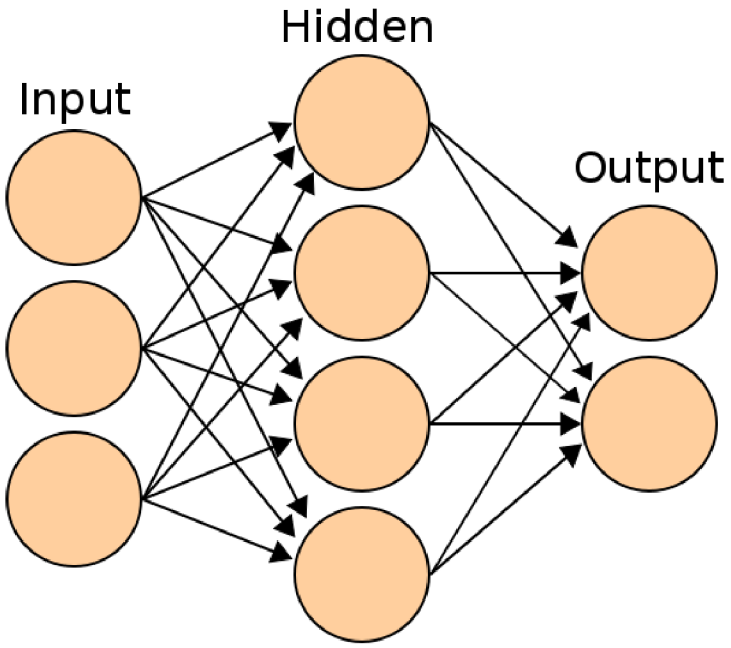


Figure 4.2: Figure of a neural network with one hidden layer. Taken from [2].

Deep neural network consists of three types of layers. Input layer which is reserved for the input data fed into the the network, one or more hidden layer for learning representations from the input data and an output layer reserved for the output values [18] as can be seen in Figure 4.2. The term hidden refers to the layer output from hidden layers not being shown and observed during training explicitly [10][12].

In neural network there is only one input at the beginning of the neural network and one output layer at the end of the neural network, but the number of hidden layers is not defined. Usually when talking about the number of layers of the neural network, the input layer is not counted, so when talking about three-layered neural network, the network actually has 4 layers: one input, two hidden and one output layers [10].

4.1.1 Artificial Neuron

Artificial neurons are the basic building blocks of neural networks. As explained in Section 2.1, neurons are comprised of the cell body, dendrites, and axons. Dendrites receive signals from other neurons, and if the sum of the signals reaches a certain threshold, a signal called an action potential is transmitted through axons to other neurons. This inspired the first computer representation of a neuron, the McCullocks-Pitts neuron, which is described

as a simple logic gate with multiple inputs arriving at one point with a threshold deciding whether the signal will be transmitted further or not (binary output) [24].

Later in 1957, Frank Rosenblatt introduced the perceptron algorithm based on this McCullocks-Pitts neuron[24] where the different inputs were given different importance (weights). An example of a perceptron with different weight of input can be seen in Figure 4.3.

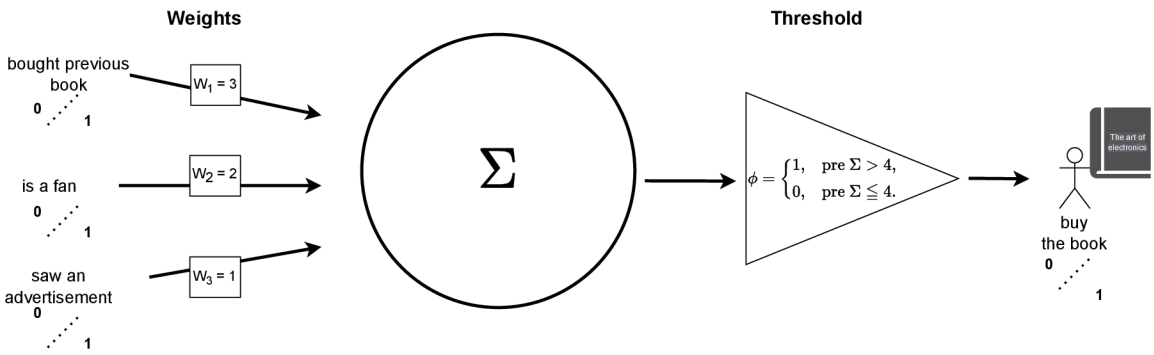


Figure 4.3: Figure of a perceptron with added weights to different input nodes that predict whether a person buys a book. This perceptron has three binary input nodes and a threshold function acting as an activation function.

The input and weights can be formed into vectors $\mathbf{x}$ and $\mathbf{w}$. It is also common when talking about artificial neurons to talk about the threshold as bias (the negative threshold). The bias is a real number and can be considered a „measure of how easy it is to get the perceptron to output a 1“ [22].

The most important equation in neural networks is:

$$z = \mathbf{w} \cdot \mathbf{x} + b \tag{4.1}$$

Originally the output of the perceptron was either 0 or 1 based on whether the z (from the equation 4.1) value was negative or positive. Later nonlinear functions were introduced to perceptron to which the z value is passed. By introducing a nonlinear function, when using multiple neurons in neural networks, the networks can compute any continuous function [22]. The functions at the end of the neuron are called activation functions and are also noted in the literature as σ.

$$a = \sigma(z) \tag{4.2}$$

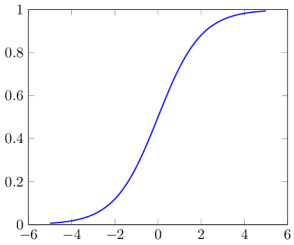
4.1.2 Activation functions

The most popular activation functions are furrther introduced.

SIGMOID

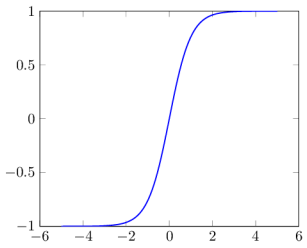
- Was first used to introduce nonlinearities to neural networks.
- The learning of a sigmoid neuron can be slower as the mean of the activation function is 0.5 and the sign for all inputs is the same [26][8].
- Due do vanishing gradient are usually avoided deep in networks

- Most common use-case for binary classification in the output layer of a neural network.
- Equation: $\sigma(z) = \frac{1}{1+e^(-z)}$



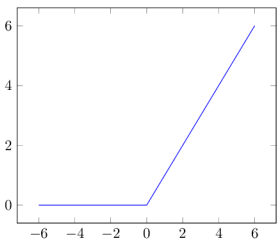
Tanh

- Is similar to the sigmoid function but ranges from -1 to 1.
- The mean of the function is near 0, and the gradient is steeper, making the saturation of a neuron less likely[8][18].
- Tanh still has the problem of vanishing gradient.
- Equation:$\sigma(z) = \frac{e^z-e^(-z)}{e^z+e^(-z)}$



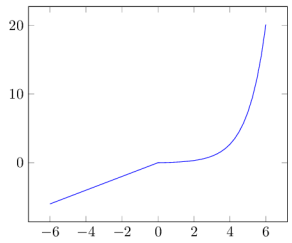
RELU

- Rectified linear unit
- Inspired by biological neurons as the neurons fire only in positive mode[18].
- One of the most popular activation functions today.
- During learning, only some neurons are activated at the same time.
- During the learning of the weights and biases, the parameters can be updated to such a value that the neuron will not be activated and will not recover during the rest of the training (also known as the dying RELU problem) [27].
- Equation:$\sigma(z) = max(0.0, z)$



ELU

- Variant of RELU.
- ELU introduces the log curve for negative values.
- Equation: $\sigma(z) \begin{cases} z & z \geq 0 \\ a * e^z - 1 & z < 0 \end{cases}$



Softmax

- Combination of multiple sigmoid functions.
- Used as output layer activation function in classification problems.
- Returns probabilities for each class.
- The sum of the output of all nodes in the layer where softmax is used is 1.
- Equation: $\sigma(z)_j = \frac{e^{z_j}}{\Sigma_{k=1}^{Classes} e^{z_k}}$, for j = 1,...,Classes

4.2 Special types of neural networks

The knowledge presented in this section was obtained from the book *Python Machine Learning* [24]. Machine learning is constantly evolving and new and more efficient neural network architectures for different machine learning tasks are invented.

In this section the convolutional neural network and recurrent neural networks will be introduced as they are a common choice for deep learning architectures for EEG classification tasks [7].

4.2.1 Convolutional Neural networks

The idea of convolutional neural network architecture came in the 1990s when Yann LeCun used this new architecture to train a classifier on a dataset of handwritten digit images with good accuracy. Convolutional neural networks, or CNN, are mainly used in computer vision. Because of good results, they are also used in other fields such as anomaly detection in time series [29] or natural language processing [31]. Convolutional neural networks are good at recognizing spatial patterns. Similarly, as in artificial neural networks, the network consists of neurons with weights, biases, and activation functions. Convolutional neural networks, however, also utilize the math operation convolution. Using convolution in neural networks creates two main advantages:

- Sharing of the parameters (filters)

- Sparsity of connections (not fully connected)

CNN usually consists of a convolutional, pooling, and a fully-connected layer called a dense layer.

Convolutional layer

The main idea behind convolutional neural networks is to create feature maps from the input data using convolution. A feature map is created by applying a filter to the data in a sliding window approach. The equation for convolution used in convolutional neural networks is:

$$\mathbf{y} = \mathbf{i} * \mathbf{f} \rightarrow y[j] = \Sigma_{k=0}^{m-1} \mathbf{i}[j + m - k]\mathbf{f}[k] \tag{4.3}$$

Where m is the size of the filter, and i is the index element of the feature map. The size of the filter applied decides how many pixels in an image or points in a data sample are used to extract one feature element in a feature map. Convolution can also be used for data with more than one channel. In the case of multiple channels, the kernel usually has the same number of channels as the input, so each channel has its weights. The kernel across all channels allows the neural network to learn and detect the spatial features for a specific task.

Padding

Zero padding can influence the output dimensionality of the convolution. Three general modes of padding can be applied to the input data:

- **Full** with padding that increases the dimensionality of the convolution output
- **Same** with the same dimension of the convolution output as the input
- **Valid** with no zero padding that reduces the dimension of the convolution output data

Full padding is usually not used in convolutional neural networks. Sometimes the input data are not shifted to the left by one data point but by stride (s) data points during the convolution to compress the convolution output.

Pooling layer

The pooling layer has two parameters, the size of the window the pooling is applied to for computing one output element (also applied in a sliding window approach), and the stride the window should be applied to the data similarly as in the convolutional layer.

There are two types of pooling layers.

- Max pool layer – that takes the maximum value from the window
- Average pool layer – that computes the average of the window

Pooling layers are used to reduce the data's dimensionality and make the extracted features more robust.

4.2.2 Recurrent neural networks

Recurrent neural networks, abbreviated RNN, are a family of neural networks created to process sequential data.

There are four types of sequence modeling tasks.

- many-to-one with an output of fixed size vector (EEG emotion classification).
- one-to-many where the input is in a standard format, and the output is a sequence (music generation or an image description generator).
- many-to-many where the number of inputs is the same as the number of outputs, and both are sequences (video classification with output for each frame in the video).
- many-to-many where the number of inputs differs from the number of outputs, and both are sequences (machine translation).

The weights are shared similarly as in the convolutional neural networks. The difference in recurrent neural networks is that the weights are shared across time. In recurrent neural networks, the recurrent unit has two inputs. Input from the same hidden unit from the previous time step and input from the input layer at the current time step. In the recurrent unit, three weight matrices are shared across the whole sequence. The three shared weight matrices are W_{xh} between the input and the hidden layer, W_{hh} between the recurrent edge, which is between the activations at different timestamps, and W_{yh} between the hidden and the output layer (for reference see Figure 4.4).

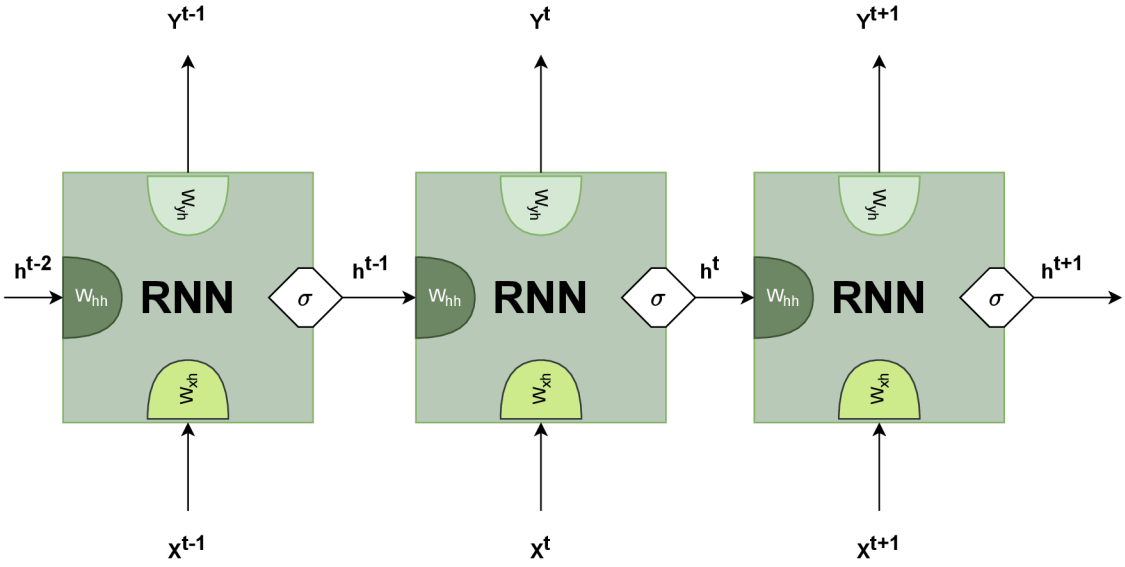


Figure 4.4: Figure depicting recurrent neural network "unrolled" through time t. Figure was inspired by a similar figure from the *Python Machine Learning* book [24].

As these weights are shared across the whole sequence, a problem such as vanishing or exploding gradient can occur during the learning process. The gradient is then either too small, and the weights are unable to update, or too big, and the change will not reach optimum. To solve the vanishing gradient problem gates were introduced to a recurrent

neural unit. There are also many different ways to utilize gates to reduce the vanishing and exploding gradient, but no variants utilizing the gates beat both the Gated recurrent unit and long-short term memory[12].

Long short-term memory units

The information in this subsection is obtained from the book *Python Machine Learning*[24] and from the video *Long Short-Term Memory (LSTM), Clearly Explained* [28]. There are three gates in the modern LSTM unit that deal with the vanishing and exploding gradient of a recurrent neural network unit.

- **Forget gate**

  Forget gate tells what percentage of the cell state should be passed further.

  This gate allows the cell state to reset the cell state if the network decides the value should be supressed and is not important for other timesteps.

- **Input gate**

  Input gate is deciding what percentage of the input data should be remembered long-term.

  Closely related to the input gate is the input node which computes potential long-term memory to update the cell state with.

- **Output gate**

  Output gate decides what percentage of the cell state to pass to the hidden units as a short-term memory.

  Which is then used for computing the hidden units at current timestep t.

All gates use the sigmoid function and each gate has its own weights and biases to learn. The number of parameters for the network to learn is greater than in plain recurrent neural networks, and the time needed for the training of the LSTM unit is also longer. This type of architecture was successfully used for many tasks, such as speech recognition and video modeling.

Gated recurrent units

The main difference between LSTM and gated recurrent unit (GRU) is that gated recurrent unit has one gate that decides whether the cell state should be reset or updated [12]. GRU units have less parameters that need to be trained.

4.3 The process of learning of a neural network

In order to detect and diagnose mistakes that can occur during the training of the neural network, it is important to know more about how the neural network is learning its parameters.

4.3.1 Forward Propagation

The process of computing the output of the neural network from the input layer to the output layer through all hidden layers is called forward propagation [18]. Forward propagation is computed through all neurons using a set of two equations, where L refers to the layer number and $A^{[0]}$ is the input layer:

$$\mathbf{Z}^{[L]} = \mathbf{W} \cdot \mathbf{A}^{[L-1]} + b, \text{ where L“1} \tag{4.4}$$

$$\mathbf{A}^{[L]} = \sigma(\mathbf{Z}^{[L]}) \tag{4.5}$$

During forward propagation, the values Z and A are saved. After the output (prediction) of the neural network is computed, the error of the predicted value with regard to the known output is calculated.

4.3.2 Cost function

The error of the predicted value with regard to the known output is computed using the loss function, also called the cost function. The output of a neural network, when given a data sample, is a prediction noted $\hat{y}$. The loss function is a function that defines how good the predicted values of the given data are compared to the actual expected values [9][23]. An ideal neural network would predict the exact expected values, and the loss function would be zero.

There are many functions that can act as a loss function. Loss functions that are often used in supervised learning are mean squared error for regression problems and categorical cross-entropy for classification tasks.

- Mean squared error

  Equation:

$$C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{4.6}$$

- Categorical cross-entropy

  Used for classification problems.

  Equation:

$$C = -\frac{1}{n}\sum_{i=1}^{n}[y_i ln(\hat{y}_i) + (1 - y_i) ln(1 - \hat{y}_i)] \tag{4.7}$$

Cost function enables the neural network to learn. As the cost function gives estimate of how „bad“ the neural network is, the neural network is able to learn by finding the minimum of the cost function using a combination of optimizers – gradient descent and back-propagation[18].

4.3.3 Gradient descent

Gradient descent is an optimizer used in machine learning, statistics, and data science to find the minimum of any differentiable function. Gradient refers to the vector of the directions where the function is increasing the fastest, which is the partial derivative with respect to every parameter of a function.

The values the parameters should be updated with (the step size) are computed as the negative gradient multiplied by a small number called **learning rate**. The learning rate ensures that the parameters are updated by a large number if a value is far from the optimum (gradient is a large value) and a smaller number when close to the minimum (gradient is a smaller value).

The gradient descent algorithm consists of 5 steps:

1. Randomly initialize parameters of a function
2. Compute function gradient
3. Compute step size using learning rate α and gradient
4. Update values
5. Repeat number 2-4 until reaching a satisfactory result where the step is very close to zero or smaller than a prefixed small value, or a prefixed number of iterations is reached.

The learning rate α is an important hyperparameter of neural network training. When the value is chosen too small, the learning will be very slow, and if the learning rate is chosen too big, the learning rate might not reach the local minimum and might diverge away.

A typical gradient descent algorithm minimizes the cost function on all the training data samples, but that can be computationally extremely expensive. An approach where the network parameters are updated after computing the loss on a subset of the whole dataset is nowadays often used. This adjusted algorithm is called stochastic gradient descent (SGD) if the model updates its parameter after computing the loss for one data sample. Mini-batch gradient is a gradient descent whose loss function is computed for a number of samples called *batch size* and then adjusted.

4.3.4 Back-propagation

Back-propagation is an algorithm for numerically evaluating partial derivatives "backward" from the output of the neural network making use of the chain rule[12]. The chain rule defines that if a variable f is dependent on the variable z and variable z is dependent on variable x, then the partial derivative of f with regard to x can be computed using the partial derivative of the f with regards to z multiplied by the partial derivative of z with regard to x (as can be seen in the equation 4.8).

$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial z} * \frac{\partial z}{\partial w} \tag{4.8}$$

The chain rule is the most important building block of back-propagation. It is used with the loss function. The loss function is a multivariate function with every parameter of the neural network so a partial derivative of the loss function with regard to every parameter can be computed. During back-propagation, the partial derivative is calculated using the values saved during the forward propagation and the network parameters are then updated correspondingly using gradient descent algorithm.

4.4 The theory of tuning neural network

During the training of a deep neural network, the machine learning practitioner is constantly tuning the neural network hyperparameters. [18].

> "*The deep learning practitioner typically spends little of her time engineering features, instead spending it modeling data with various artificial neural network architectures that process the raw inputs into useful features automatically [15].*"

As quoted above, the data preprocessing and feature extraction for deep neural networks are not time-consuming or extensive. However, creating deep neural network architectures can be challenging as the network can be adjusted using many different hyperparameters. The difference between parameters and hyperparameters is that hyperparameters are for the machine learning practitioner to tune, and parameters are for the machine learning algorithm itself to tune automatically during the training. There are many hyperparameters to tune during the training.

A list of a few possible hyperparameters is:

- The number of hidden layers in a neural network.
- The number of hidden units in a hidden layer.
- The choice of the activation functions.
- The learning rate in gradient descent.
- Batch size in mini-batch gradient descent.
- Number of epochs the neural network is trained on.

The whole training of a deep neural network in deep learning is an iterative process. It is highly unlikely that the first initial design with chosen hyperparameters of the model would perform with satisfactory results. During the design of a deep neural network, first, an idea of the neural network architecture has to be designed, implemented, and then trained and evaluated. From the obtained results, new experiments and ideas to improve the neural network need to be thought of and again implemented and trained (visualization of the iterative process can be seen in Figure 4.5).

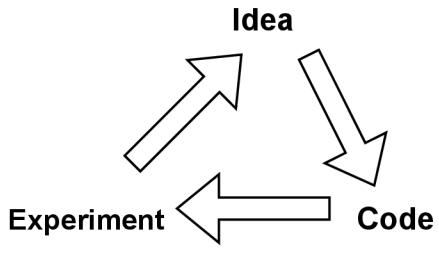


Figure 4.5: Figure depicting the iterative process of designing a neural network model, inspired from [11].

4.4.1 The problem of high bias and high variance in neural networks

Deep neural networks can recognize and extract global patterns from data. However, two main problems can occur during the training of a neural network, making this recognition more difficult.

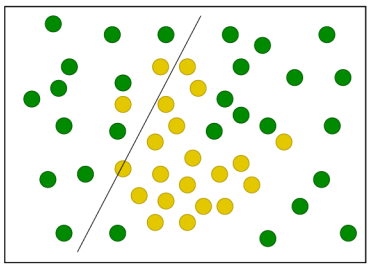


Figure 4.6: Figure showing underfit in two-dimensional data

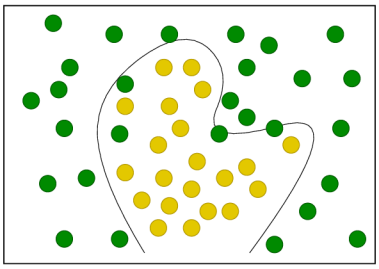


Figure 4.7: Figure showing good fit in two-dimensional data

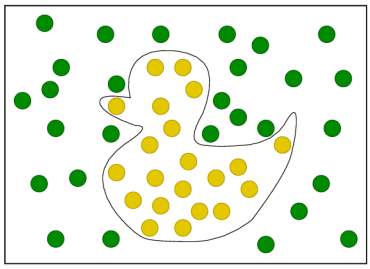


Figure 4.8: Figure showing overfit in two-dimensional data

High variance

High variance is a state of a neural network when the neural network does not generalize well to the new data. When a model has high variance, it is also called overfitting. Deep neural networks are very prone to overfitting as neural networks have many parameters. When a neural network architecture is too complex, or the number of epochs the neural network was trained on is too big, the neural network tends to learn the training examples rather than the general pattern contained in the data. It is the same as if a student is trying to learn to solve a particular type of math problem, but instead of understanding the concept, the student memorizes the answers, which makes him unable to perform well on new unseen examples.

If the input data were two-dimensional, the way to spot overfitting is straightforward (as can be seen in Figure 4.8). However, the neural network inputs are usually multidimensional data that are not easily imaginable or visualized. Overfitting is diagnosed through the difference between the error on training data compared to the error on validation data. If the difference between these errors is big and the training error is small, then it can be said that the neural network is overfitting.

Reducing high variance of neural network

Overfitting can be solved using multiple methods. One of the methods to reduce overfitting is by giving the neural network more training data. That might not always be possible, so another option is to use data augmentation by adding transformations such as shifting or flipping the images or adding noise to the original data.

Another method is L1 or L2 regularization, called LASSO and ridge regularization. L1 and L2 regularizations can be used for weights, biases or for both. Usually they are used for weights. These regularization techniques penalize the usage of parameters by adding the penalized parameters to the cost function. How severe the penalty of the parameter is, is given by the value λ. This results in the reduction of the impact of hidden units on the cost function.

When using L2 regularization, the value added to the cost function is $\frac{1}{2}\lambda w^2$ for every weight w. In L1 regularization the value added to the cost is $\lambda|w|$ for every weight w.

> " *L1 regularization tends to lead to the inclusion of a smaller number of larger-sized parameters in the model, while L2 regularization tends to the inclusion of a larger number of smaller-sized parameters [18].*"

The last method used is the dropout layer. Dropout is a regularization method specific for neural networks where random neurons are ignored during the training. Dropout is added to hidden layers and tells the percentage of the hidden units that should be ignored during a learning step. For example, if a hidden layer has ten neurons and a dropout with 0.1 is set to this layer, then one random neuron during training will be ignored during each round of training/epoch.

High bias

Underfitting or a high bias problem is when a neural network is not a good fit even for the training samples. An example in two-dimensional data can be seen in Figure 4.6. It is caused when the neural network architecture needs more complexity or the number of epochs is too small for the model to learn the problem.

High bias is diagnosed by looking at the error of the model on the training data. If the training error is big, then the network suffers from high bias. What is considered a big training error differs from problem to problem based on the Bayes error.

Reducing high bias of a neural network

Underfitting can be solved by adding more hidden units to a layer or more hidden layers to the neural network. It might also be helpful to train the neural network longer.

Chapter 5

Proposed methodology

As said in chapter 4, a deep learning pipeline consists of obtaining data, processing them, and then training a neural network. In this chapter, the dataset used will be described, and the initial models will be later implemented and tuned.

5.1 SEED-IV dataset

The SEED-IV [32] dataset was published in 2018 by Wei-Long Zheng, Wei Liu, Yifei Lu, Bao-Liang Lu, and Andrzej Cichocki and is used for emotion recognition in this thesis. This dataset contains the EEG recordings of 15 subjects. This dataset contains the EEG recordings recorded while the subjects were watching 72 different video clips targeting different emotions. The targeted emotions were neutral emotions, sadness, fear, and happiness (target emotions are depicted in Figure 5.1). The videos for targeted emotion were selected from a total of 168 clips after 44 participants were asked to evaluate the key emotions. The evaluations were then processed, and the final 72 film clips with the highest matching were selected.

Videos were shown to subjects in three sessions, where each session was recorded on a different day and showed a different set of 24 clips to the subjects, where each emotion was equally represented by six clips. In this study, both males (7) and females (8) of ages ranging between 20 and 24 participated. The EEG was recorded with ESI NeuroScan System using a 1000 Hz sampling frequency on 62 channels and later downsampled to 200 Hz (the electrode placement can be seen in Figure 5.2). During this study, eye movements were also recorded, but will not be used in this thesis.

Figure 5.1: Figure depicting the target emotions for SEED-IV dataset

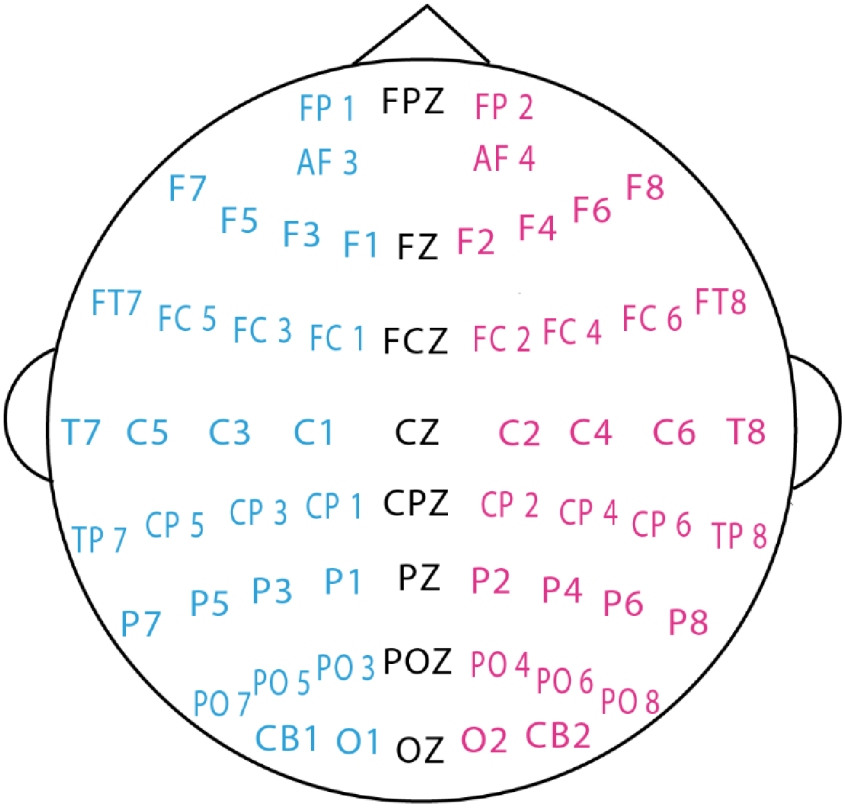


Figure 5.2: Figure depicting the channels used for the EEG recording in SEED-IV dataset taken from [32]

5.1.1 Data preparation

I have decided to create two binary models classifying either sadness and happiness or neutral and fear emotions from the SEED-IV dataset. The dataset came with pre-defined labels, so only a preprocessing of filtering the recordings into different folders for each class

was needed. The EEG recordings were of different lengths, so I decided to segment the recordings into a 1-second long recording of 200 timestamps. The length of one second was chosen in order to obtain more data samples, as deep learning models work well with larger datasets.

5.1.2 Implementation of the data preparation

The implementation of the data preprocessing was done in the scripting language Python 3.10.11[1]. The recordings were first divided into different folders for corresponding labels. As the dataset was in the `.mat` format, I used the scipy loadmat function and segmented and saved them into `.npy` files containing EEG segments of equal length. This was done in script `segment_samples.ipynb`.

After segmentation, 31650 samples belonged to the happy class, 40715 to the sad class, 36750 to fear, and 40575 to neutral emotion. The dataset for corresponding model training was always loaded to the memory without differentiating between sessions.

To obtain the performance of the model, I have decided to use the holdout method, where the whole dataset was divided into three sets: 75% of the data for model training, 10% of the data for validation, and 15% as a test of the performance of the model on unseen data. Before dividing the dataset into these three sets, the data were shuffled with a seed of 42 to obtain reproducible splits for the training, validation, and test datasets.

5.2 Training process

The process of training a neural network consisted of:

- Split data into validation, test, and training dataset as mentioned in subsection 5.1.1.
- Design and implement the neural network.
- Train the neural network.
- Diagnose validation and training loss graph for the training duration.
- Evaluate the performance of the neural network on the test dataset.

The implementation of the models was done in scripting language Python 3.10.11 using the Keras[2] (version 2.12.0) open-source library.

The neural networks were trained using Google Colab[3]. Google colab is cloud-based Jupyter notebook allowing fast training of the neural network in `.ipynb` files.

After I have read about the models used for EEG classification tasks [7] and studied the different architectures I then chose model as my base models that I will try to further tune for my classification task.

5.2.1 Keras

Keras makes creating and training new models accessible as the API contains computationally effective implementations of neural network layers. There are three ways models can be created in Keras.

[1] https://www.python.org/
[2] https://keras.io/
[3] https://colab.research.google.com/

1. **Sequential model**

   The sequential model allows neural networks to have only one input and one output.

   In sequential models, the layers are stacked linearly into a `tf.keras.model`, which does not make it able to create, for example, skip connections.

2. **Functional application programming interface (API)**

   Functional API allows the creation of more complex structures with skip connections and multiple inputs and outputs.

3. **Create subclass of the model class**

   Reserved for use in research use cases.

In my thesis, I am using functional API to implement the models.

Both the sequential model and the models created using functional API can be trained easily using the built-in training loop (the `fit()`) method) and evaluated using the built-in `evaluation()` method. The model creation methods offer method `save` for saving the model, trained weights, state of the optimizer, and model training configuration, allowing further retraining of the models.

Adam optimizer is my choice for the network optimization algorithm[16] as it is "computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradien" as is said in the Keras page for the Adam API[4].

Deep neural networks are prone to overfitting, so I have decided to use the callback `ModelCheckpoint` that Keras offers for saving the model with the lowest loss on the validation set and with the highest validation accuracy. This allows me to save the best neural network before overfitting starts.

As I have decided to divide the task into two binary classification problems, I have chosen the `binary_crosentrophy` loss function for the training of the models.

5.2.2 Model designs

As the classification of the four emotions can be considered as two binary classification problems, both neural networks have a single output node with a sigmoid activation function as the output layer.

The input layer consists of EEG data with the shape 200x62, where 200 is the time dimension, and 62 represents the number of channels.

Single LSTM layer neural network

The first model architecture I have decided to apply to my binary classification problem is a model with a single LSTM layer with multiple hidden units. I have used this architecture because LSTM units were designed to handle sequence data. The model design is depicted in Figure 5.3.

[4]https://keras.io/api/optimizers/adam/

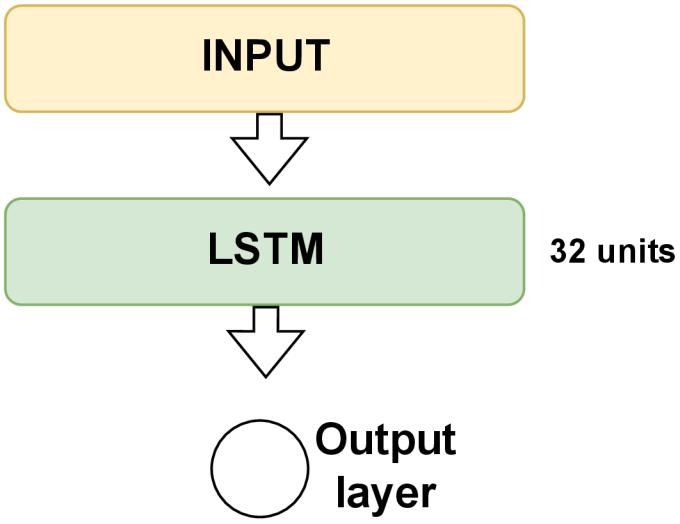


Figure 5.3: Figure showing a model diagram with only one hidden LSTM layer

Single GRU layer neural network

The second model architecture I have tried to apply to the binary classification problem is a model with a single GRU layer with multiple hidden units. I have decided to use this architecture because GRU units have fewer parameters and should be trained faster. The model design is depicted in Figure 5.4.

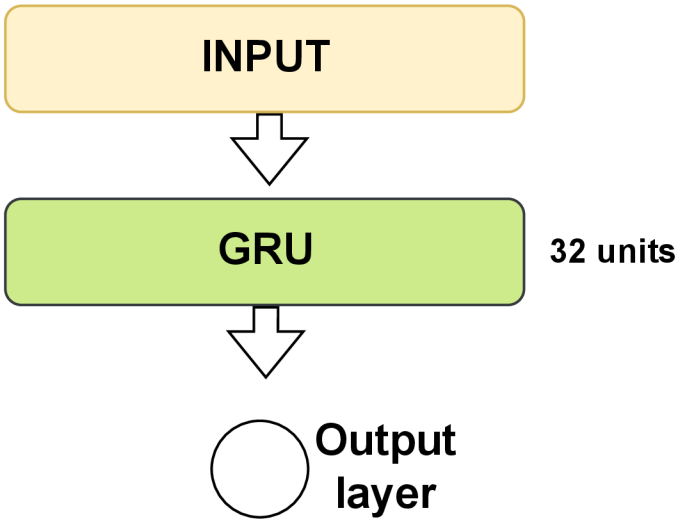


Figure 5.4: Figure showing a model diagram with only one hidden GRU layer

Two GRU layer neural network

I have decided to use two hidden layers to add complexity to the neural network and chose GRU for faster computation as the number of parameters to train is fewer. A similar type of neural network with two LSTM layers, adjusted activation functions to RELU and sigmoid with a Dropout layer between them was already once successfully used for raw EEG data with an accuracy of 85.45% for valence, 85.65% for arousal and 87.99% for dominance using DEAP dataset [3]. The model design is depicted in Figure 5.5.

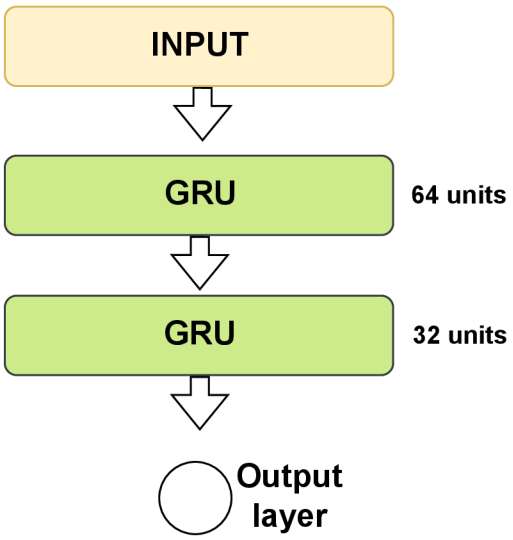


Figure 5.5: Figure showing a model diagram with two hidden GRU layers

Two-branch CNN-LSTM neural network

The last type of architecture that I tried was architecture inspired by the paper *Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysi* [30]. This architecture used skip connection. The neural network in this study has four branches concatenated before the classification output layer. Each branch has a one-dimensional convolutional layer with different kernel sizes, pooling, and batch normalization, followed by an LSTM layer. This model was used for sentiment analysis from sentences.

I have adjusted this neural network. The changed neural network only has two deeper branches with different kernel sizes on each branch, and instead of concatenation, I am using the Add layer (Changed model can be seen in Figure 5.6).

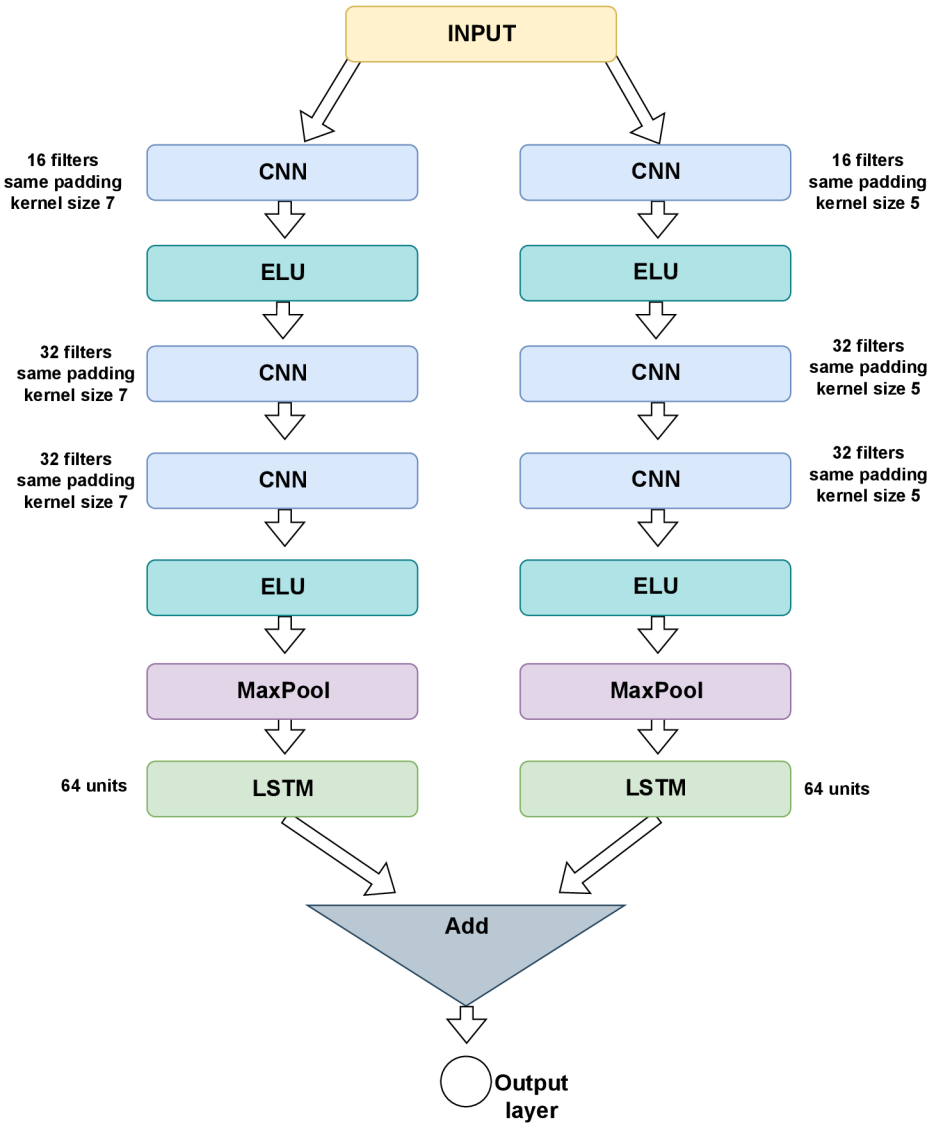


Figure 5.6: Figure showing a block diagram of the neural network. The neural network consists of two branches. The branch on the left of the Figure consists of convolutional and activation layers, followed by two convolutional layers and an activation layer. The convolutional layers have a kernel size of 7. After the convolutional and activation layers, a Maxpool layer follows, which is then passed to the LSTM layer. The right branch has the exact same structure, but the kernel size of the convolutional layers is 5. The outputs from LSTM from both branches are added using add layer and then passed to the output layer for classification.This architecture was inspired by the paper *Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysi* [30]

5.2.3 Evaluation metrics

The theory for evaluation metrics used in this section is taken from the book *Python Machine Learning* [24]. The key step in building a machine learning model is to get an estimate of how the model would work on new unseen data. Evaluation metrics are used for the

output of the neural network for the test dataset to get insight into how well the neural network performs on new data.

The main evaluation metric of a classification model is the confusion matrix. A confusion matrix in a binary classification is a matrix that reports the counts for correctly predicted one class (true positives TP), correctly predicted second class (true negatives TN), wrongly classified predictions for class one (false positive FP), and wrongly classified prediction for the second class (false negative FN).

A confusion matrix can be easily computed by comparing the predicted values with the known values, but I have decided to use the `confusion_matrix` method from the scikit-learn library. Based on the confusion matrix, other metrics that tell more about the quality of the neural network can be computed. These metrics are accuracy, precision, recall (also called sensitivity or true positive rate), and specificity, also called true negative rate.

Accuracy is the main focus during the evaluation of my neural networks. Accuracy is computed as the ratio of correctly classified samples to all samples.

$$accuracy = \frac{TP + TN}{TP + FP + TN + FP} \tag{5.1}$$

If I have focused only on the accuracy, the results might have been misleading.

During the training it can sometimes occur that the model learns to predict only one class. That is why other metrics are also important to give better insight into the quality of the model.

$$precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$recall = \frac{TP}{TP + FN} \tag{5.3}$$

$$specificity = \frac{TN}{TN + FP} \tag{5.4}$$

$$f1 = 2 * \frac{precision * recall}{precision + recall} \tag{5.5}$$

As I have decided to implement a binary classification model a threshold for differentiating the classes is needed. An ideal binary classification model would output 0.0 for one class and 1.0 for the other.
In neural networks with single hidden unit the output is a prediction in the range from 0 to 1 so it is necessary to choose good threshold. For choosing the correct threshold the area under the curve and receiver operating characteristic curve (AUC-ROC) are used. This is metric used to gain insight into how good the model performs classification with different thresholds. The AUC tells how good the model is at distinguishing the classes. ROC curve is the curve between the false positive rate on the x-axis and true positive rate on y-axis for chosen threshold. The ROC and AUC curve were computed using the `roc_curve` and `roc_auc_score` functions from the scikit-learn library.

Chapter 6

Results

As shown in Figure 4.5, training a deep neural network is an iterative process. This chapter introduces the training and the results previously achieved on the SEED-IV dataset in section 6.1, and the rest of the chapter describes the achieved results during my neural network training.

During the training of a neural network I am using batch size of 32. Each neural network is first trained on 50 epochs with Adam optimizer with the Keras default learning rate of 0.001. During the training of the neural networks a `ModelCheckpoint`, as mentioned in subsection 5.2.1, was used, so the result of every training process was three neural networks. One with the lowest validation loss, one with the highest validation accuracy, and the one that was trained the longest. The results shown are always of the neural network with the best performance from the three models mentioned above.

6.1 Previous results on the dataset

In 2019 a paper using the SEED-IV dataset for emotion classification was published [19]. The data were preprocessed by a notch filter, removing the raw data's frequency artifacts, then re-referenced to a moving average re-referencing method and smoothed using a one-dimensional 10th-order median filter. Then features from the time and frequency domain were extracted. The figure depicting all the feature extraction methods the paper is using can be seen in Figure 6.1.

Feature Type	Feature Name
Time domain (Statistical Features)	Mean, Root mean square, Std-deviation, First difference, Normalized first difference, Second difference, Normalized second difference, Kurtosis, Skewness, Variance, Mobility and Complexity (Hjorth Parameters)
Wavelet or Frequency Domain features of 5 band	Band energy, Power spectral density, Differential entropy, Average band power
Other features	Hurst exponential and Permutation entropy

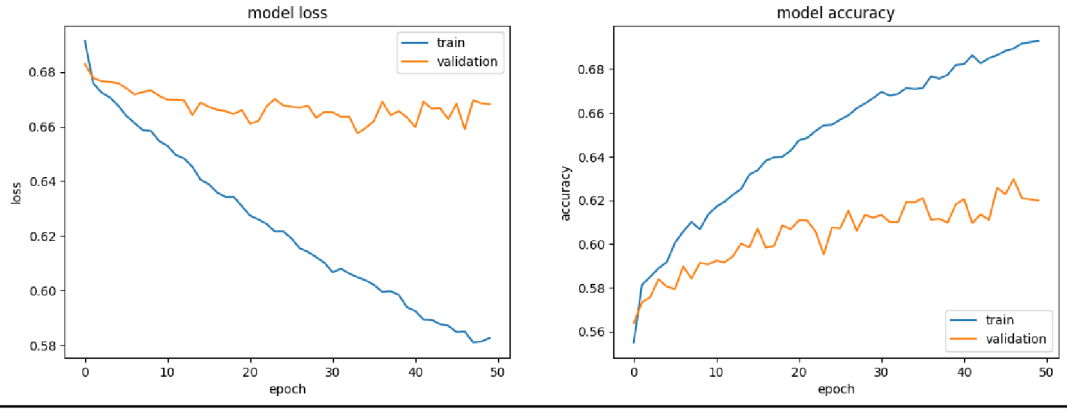
Figure 6.1: A table depicting the features extracted from the EEG signal in study [19]. Taken from [19].

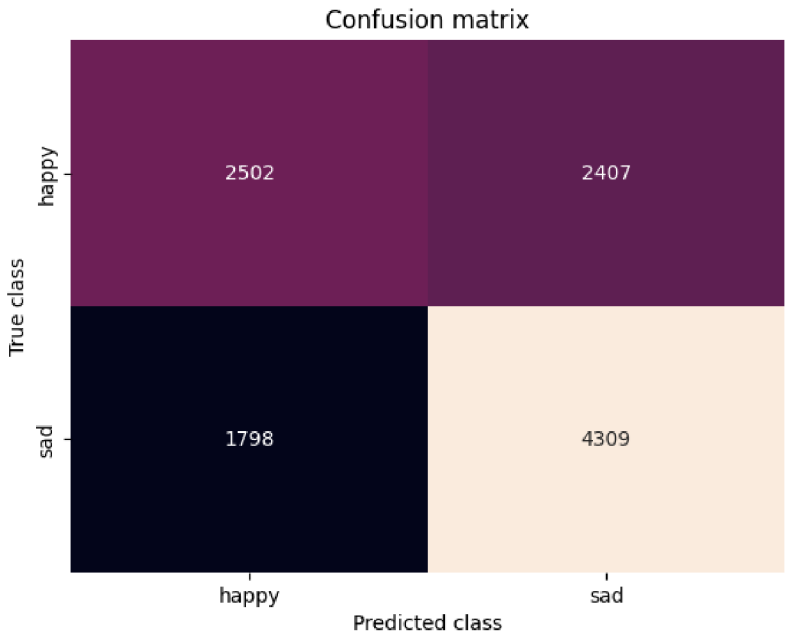
This study works with only 32 channels of the SEED-IV dataset as it also compares the machine learning algorithm for DEAP dataset[17].

Extracted features were then passed to an SVM classifier, where the data for training was from 32 subjects, and data from 13 subjects were used for testing. This study created a subject-independent performance estimate by training the neural network on 32 subjects and testing on 13 others. The four classes were classified with accuracies 79%, 76%,77%, and 74%, respectively.

6.2 Single LSTM layer neural network experiments

After training the single LSTM hidden layer neural network described in the model designs (Figure 5.3) with initial hyperparameters of 50 epochs, the results are shown in the table 6.1.

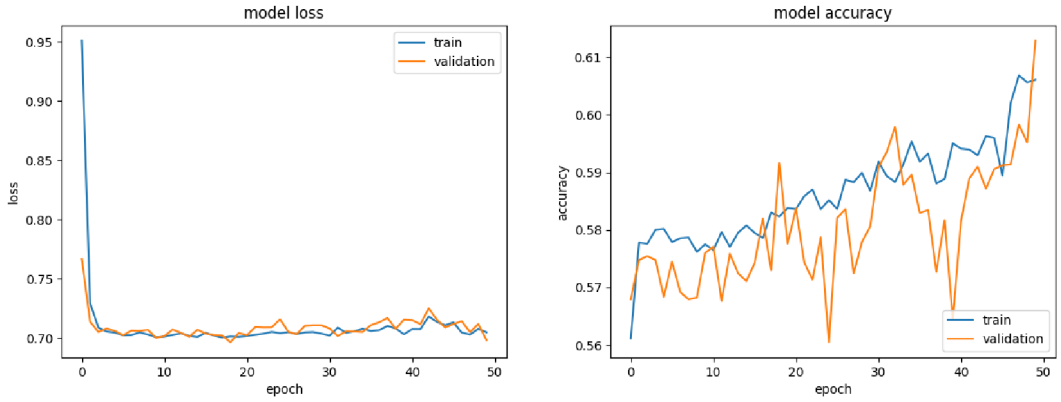


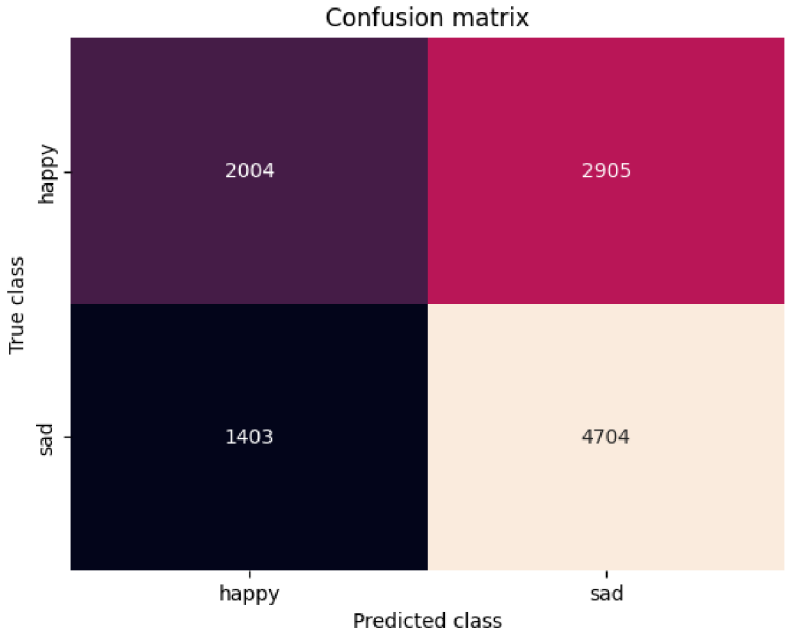


accuracy	precision	recall	specificity	f1-score	AUC
61.828	50.968	58.186	64.160	54.338	0.6534

Table 6.1: Table depicting the loss, accuracy, confusion matrix, and other performance metrics of the results after first training of the neural network with a single LSTM layer (Figure 5.3)

The loss difference between the loss on the training data and validation loss was too big, so I tried using default values of a Keras regularizer L2 on the kernel (weights).





accuracy	precision	recall	specificity	f1-score	AUC
60.893	40.823	58.820	61.822	48.196	0.6294

Table 6.2: Table depicting the loss, accuracy, confusion matrix, and other performance metrics of the neural network with a single LSTM hidden layer (Figure 5.3) after using L2 regularizer for weights

This reduced the difference between the validation and training loss but did not improve performance as can be seen in table 6.2. The neural network seemed to stagnate, but in the last epoch, the loss decreased. Because of the decrease in the last epoch, I tried to train this neural network for additional 100 epochs. Both the training and validation loss increased after 50 epochs. After another 20 epochs, the loss decreased to approximately the same value as before the increase, and then after some epochs again increased (as can be seen in table 6.3).

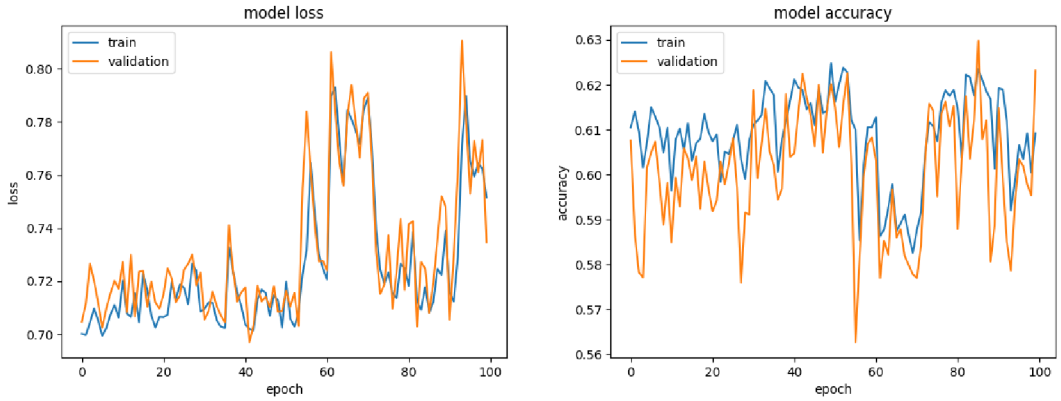


Table 6.3: Table depicting the loss and accuracy of the neural network with a single LSTM hidden layer (Figure 5.3) after further training with L2 regularizer used for weights

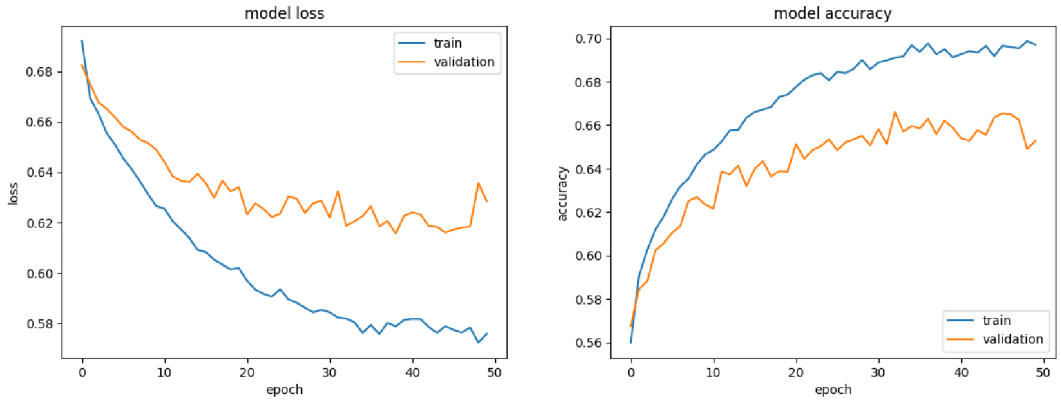
The two possible reasons for this behavior that might have caused this are:

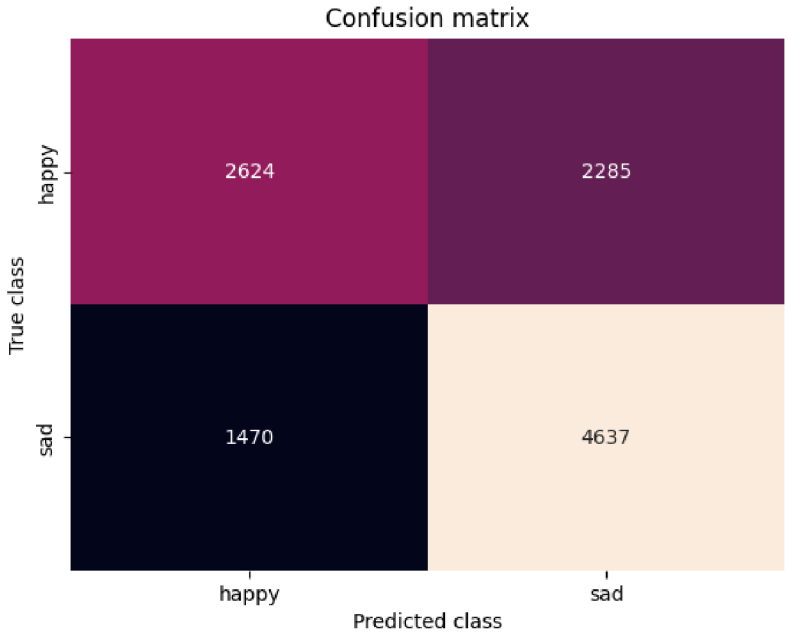
1. The neural network is not complex enough (underfitting).
2. Adam optimizer parameters need to be tuned (learning rate, exponential decay rate)

As the neural network has only a single LSTM layer, it might not be able to pick up the complex patterns representing emotion from the brain.

6.3 Single GRU layer neural network experiments

The results after training the neural network with a single hidden GRU layer described in Figure 5.4 for 50 epochs with initial Adam parameters are shown in table 6.4.

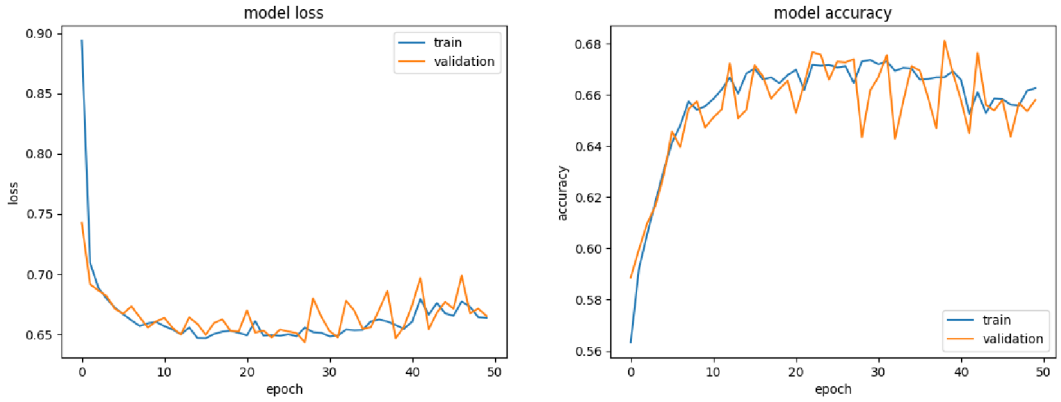


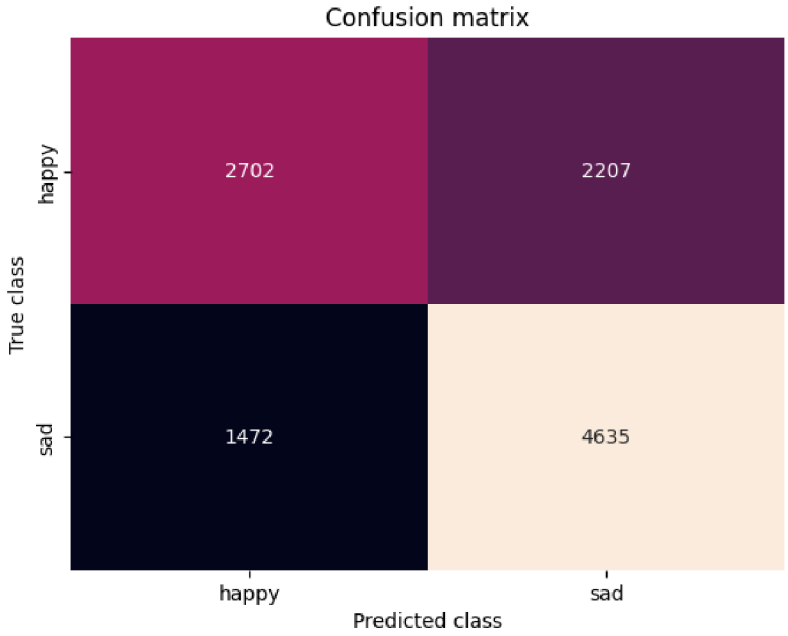


accuracy	precision	recall	specificity	f1-score	AUC
65.913	53.453	64.094	66.989	58.292	0.7110

Table 6.4: Table depicting the loss, accuracy, confusion matrix, and other performance metrics of the results after first training of the neural network with a single GRU layer (Model is shown in Figure 5.4)

The loss difference between the training and validation loss was similar to the training of the LSTM, but in this GRU neural network, I used kernel regularizer L2.

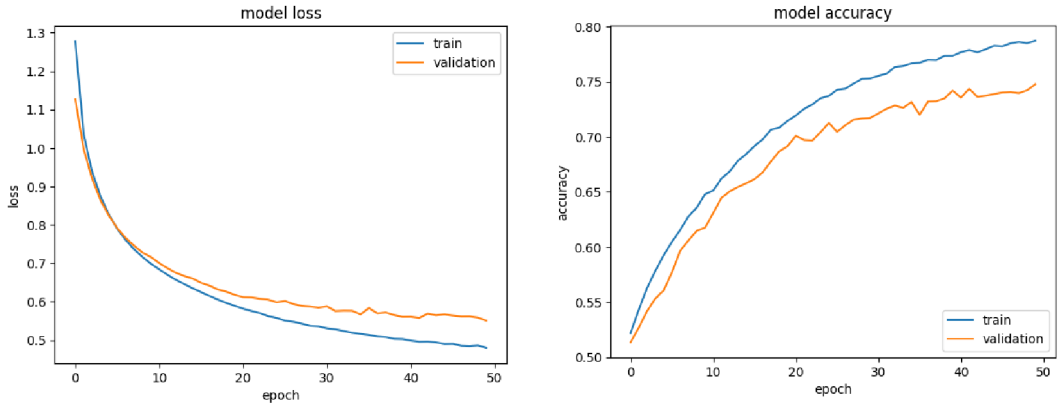


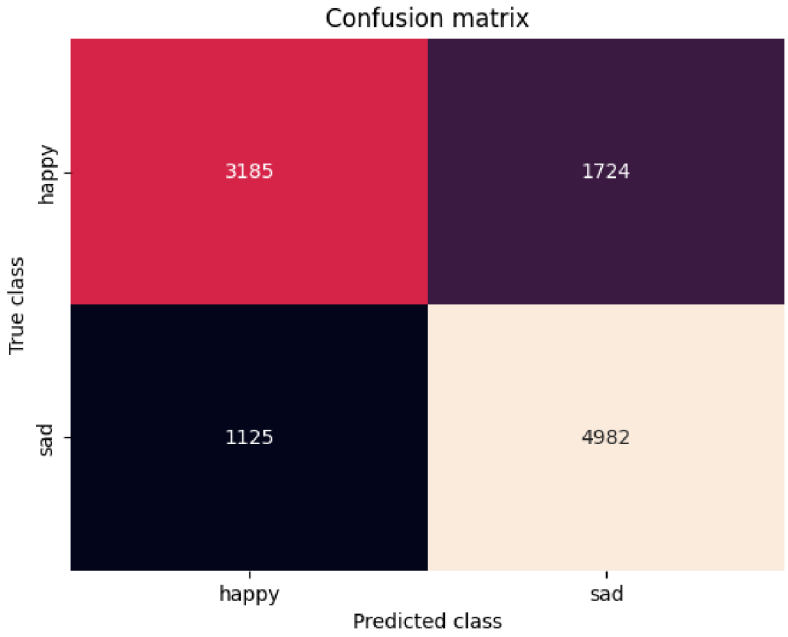


accuracy	precision	recall	specificity	f1-score	AUC
66.603	55.042	64.734	67.743	59.496	0.7139

Table 6.5: Table depicting the loss, accuracy, confusion matrix, and other performance metrics of the neural network with a single GRU hidden layer (Model is shown in Figure 5.4) after further training with L2 regularizer used for weights

This significantly reduced the difference between the validation and training loss, and the performance improved as well as can be seen in table 6.5. As the neural network seemed to stagnate the same way the LSTM model did in table 6.2, I have tried to reduce the learning rate of the Adam optimizer to 0.0001.



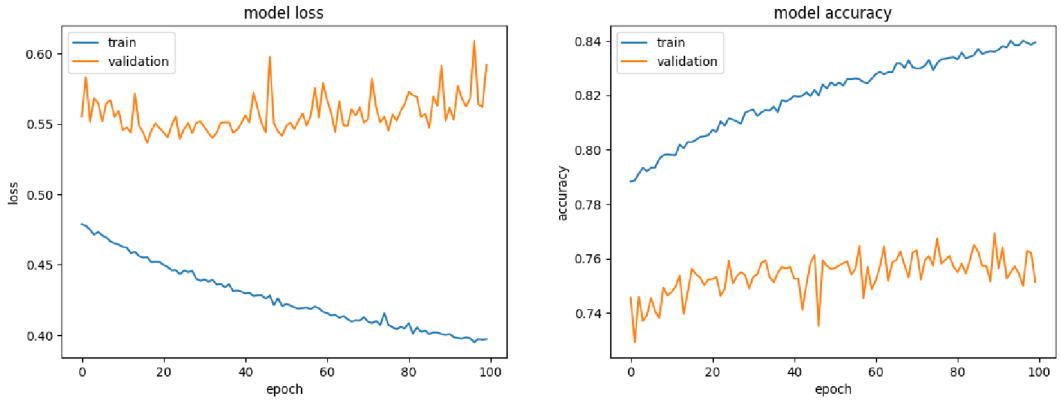


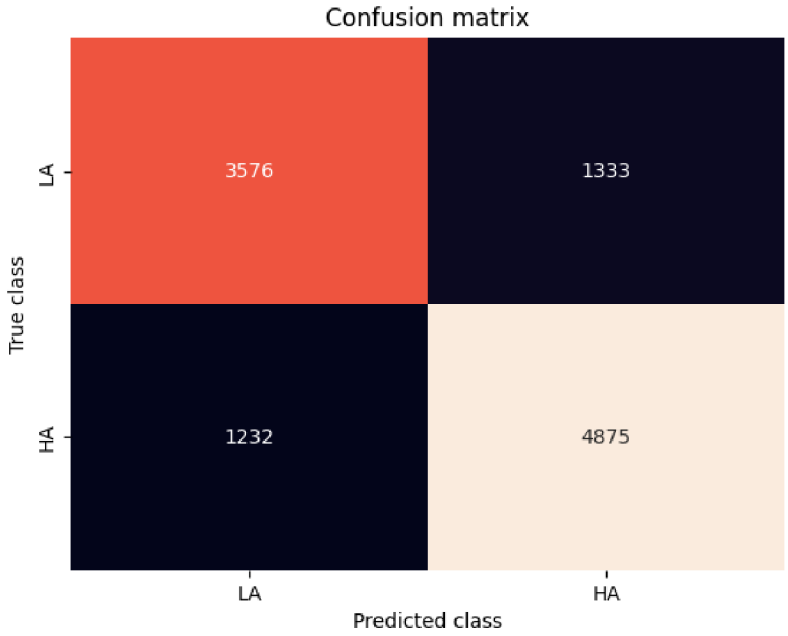
accuracy	precision	recall	specificity	f1-score	AUC
74.138	64.881	73.898	74.292	69.096	0.8146

Table 6.6: Table depicting the loss, accuracy, confusion matrix, and other performance metrics of a single layer GRU neural network (The model is shown in Figure 5.4) with L2 regularization and learning rate adjusted to 0.0001

The performance of the GRU network improved significantly by this adjustment (results can be seen in table 6.6). This also shows how important the learning rate is in neural networks.

Since the validation loss (shown in table 6.6) did not increase after 50 epochs, I have tried to train the GRU network for additional 100 epochs as in the LSTM model (for the previous results see table 6.3).





accuracy	precision	recall	specificity	f1-score	AUC
76.716	72.846	74.376	78.528	73.603	0.8445

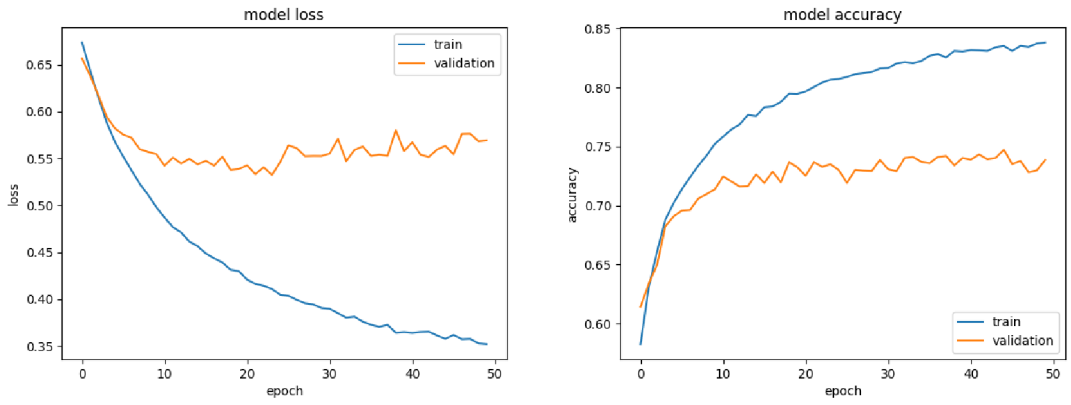
Table 6.7: Table depicting the loss, accuracy, confusion matrix and other performance metrics of a single layer GRU neural network (Model is shown in Figure 5.4) with L2 regularization and learning rate adjusted to 0.0001 after learning for additional 100 epochs

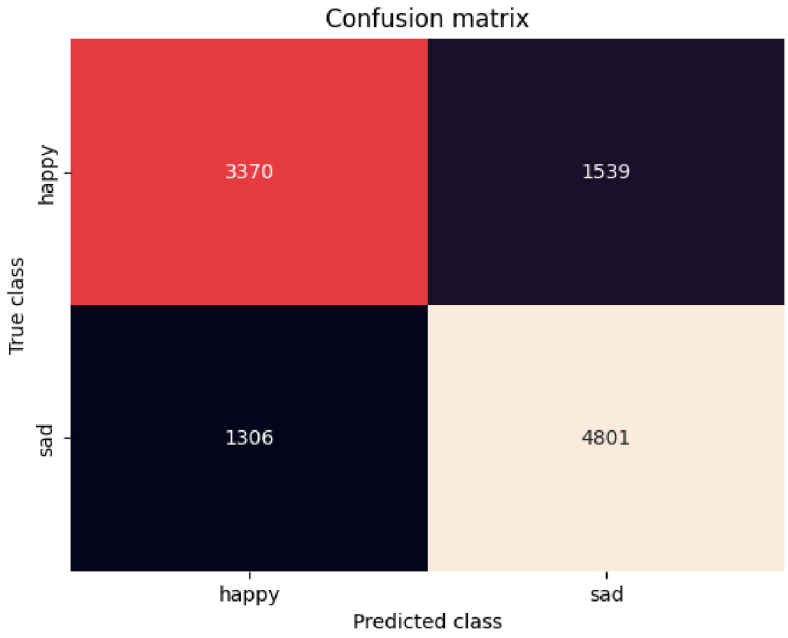
The validation loss after 100 epochs did not improve and even started to increase. The training loss is also decreasing slowly (results in table 6.7).

The increase in validation loss indicates that the neural network started to overfit the training data. I have, however, decided to train more complex neural networks further.

6.4 Two GRU layer neural network experiments

The results after training the neural network with two hidden GRU layers (model design depicted in 5.5) with initial hyperparameters of 50 epochs and Adam with learning rate 0.001 are shown in 6.8.



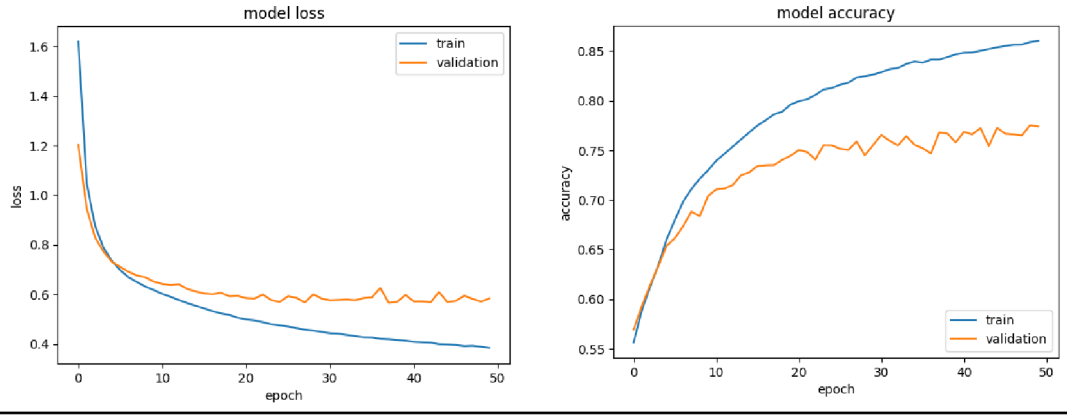


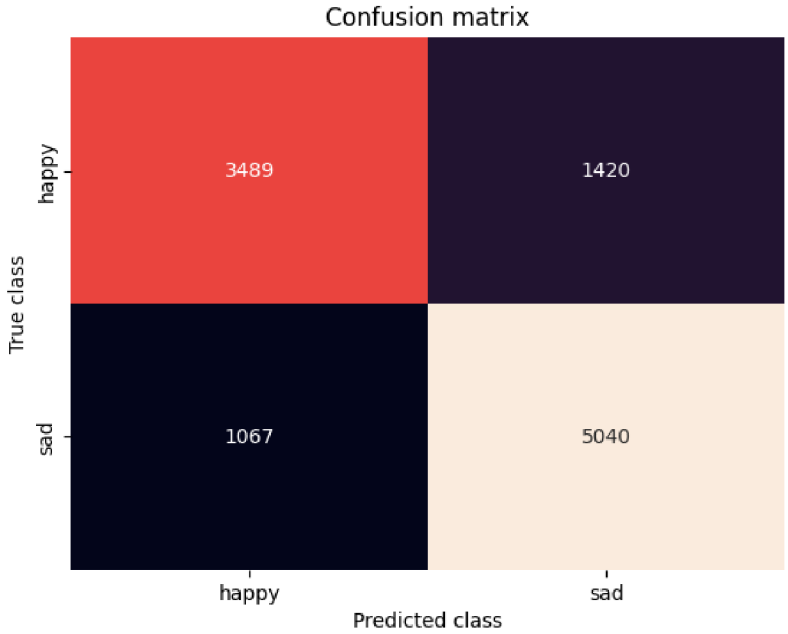
accuracy	precision	recall	specificity	f1-score	AUC
74.174	68.649	72.070	75.726	70.318	0.8164

Table 6.8: Table depicting the loss, accuracy, confusion matrix, and other performance metrics of a neural network with two GRU hidden layers (Model shown in Figure 5.5)

It can be seen that initial results for this type of neural network are much better than the single-layer neural networks. The accuracy without any adjustments was higher by 10% than a single hidden layer GRU network and 15% than a single hidden layer LSTM network.

From the previous experience on the single hidden layer neural networks, I have adjusted the learning rate of the Adam optimizer to 0.0001 and used L2 regularization for the weights on both layers of the neural network. This improved the performance of the neural network as can be seen in table 6.9.





accuracy	precision	recall	specificity	f1-score	AUC
77.424	71.074	76.580	78.019	73.724	0.8497

Table 6.9: Table depicting the loss, accuracy, confusion matrix, and other performance metrics of a two hidden layers GRU-GRU neural network (Model shown in Figure 5.5) with L2 regularization and learning rate adjusted to 0.0001

After inspecting the difference between the losses on validation and training datasets, I used another regularization technique. I added a Dropout layer between the two GRU layers of the neural network as can be seen in Figure 6.2.

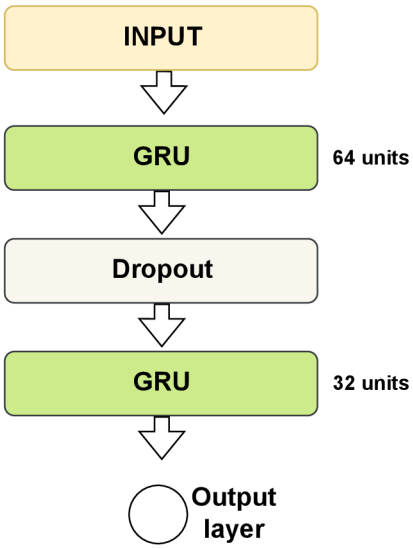
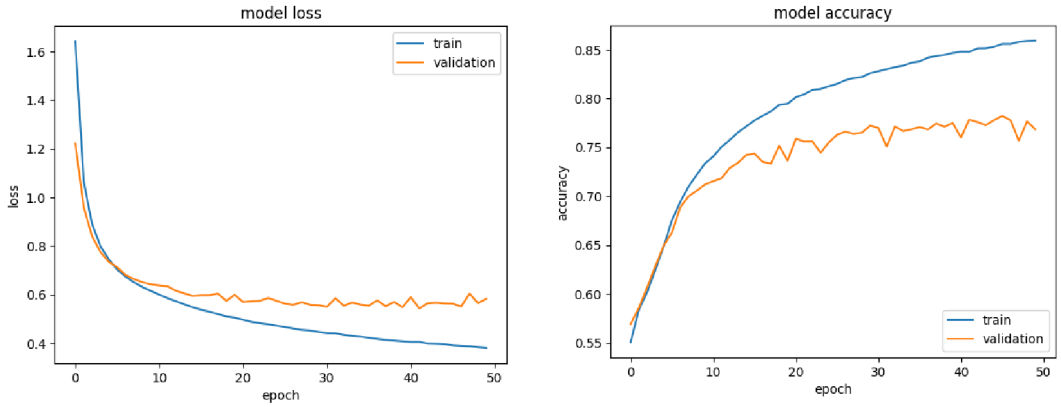
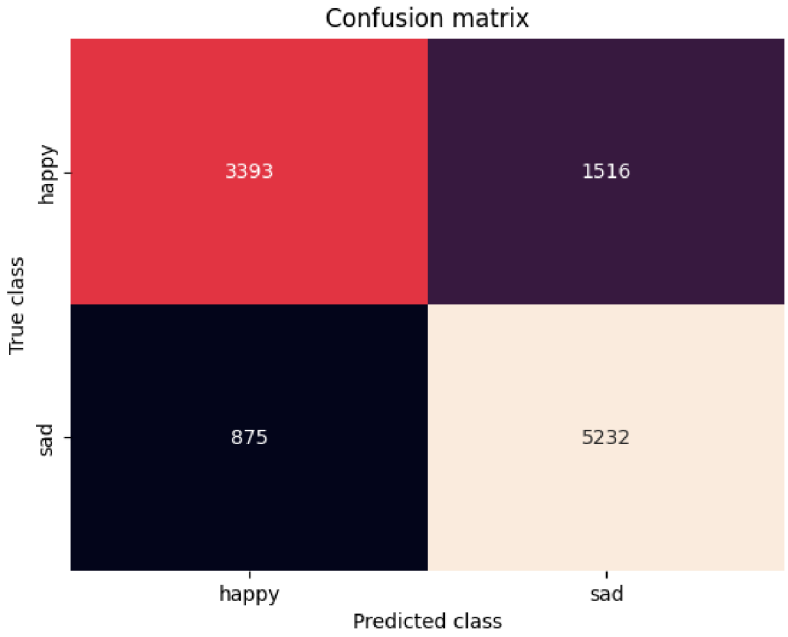


Figure 6.2: Figure showing a model diagram with two hidden GRU layers and added dropout layer between them

This did not improve the performance of the neural network much, as is shown in table 6.10.





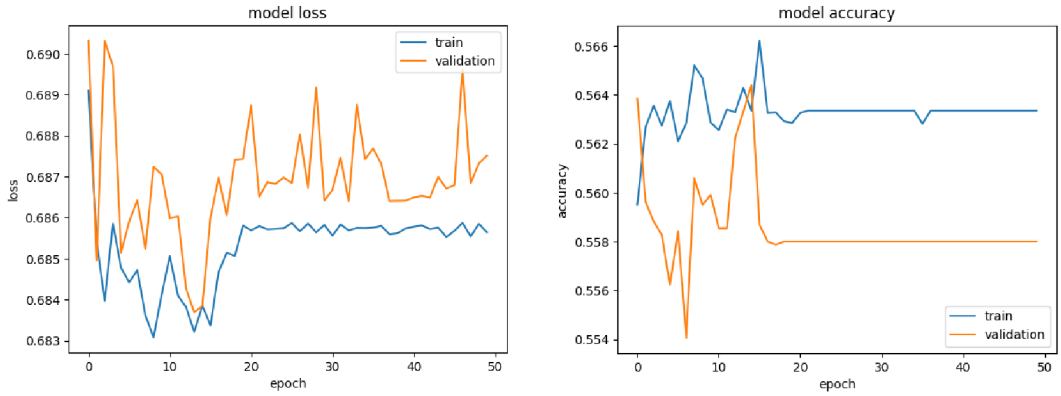
accuracy	precision	recall	specificity	f1-score	AUC
78.295	69.118	79.499	77.534	73.946	0.8579

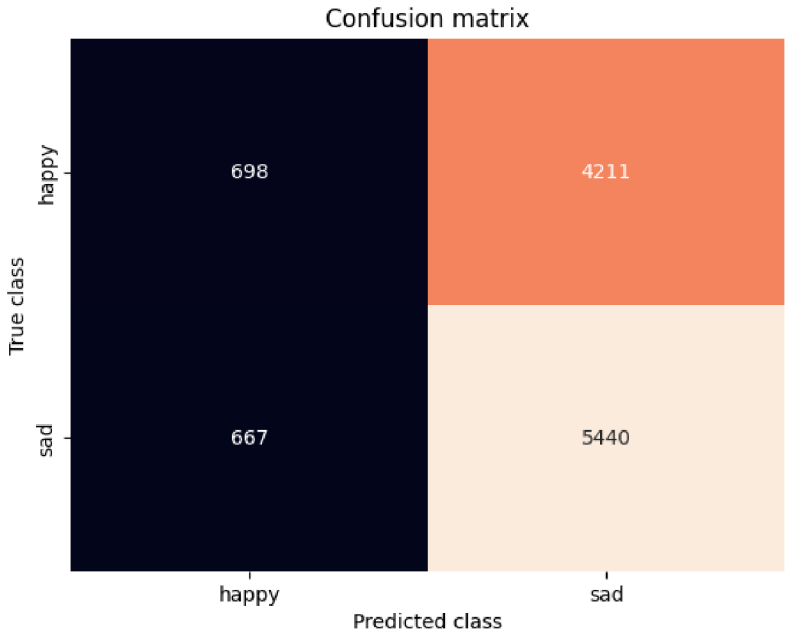
Table 6.10: Table depicting the loss, accuracy, confusion matrix, and other performance metrics of a two hidden layer GRU neural network (Model shown in Figure 5.4) with L2 regularization, a learning rate of 0.0001, and added dropout layer

Adding a dropout layer to GRU-GRU neural network did not significantly improve the performance of the neural network. The initial performance of the neural network, however, increased with the increased complexity of the neural network, so I focused more on the more complex neural networks.

6.5 Two-branch CNN-LSTM neural network experiments

The neural network with two branches of CNN-LSTM layers (Model design shown in Figure 5.6) was first trained with initial hyperparameters of 50 epochs and Adam with a learning rate of 0.001.

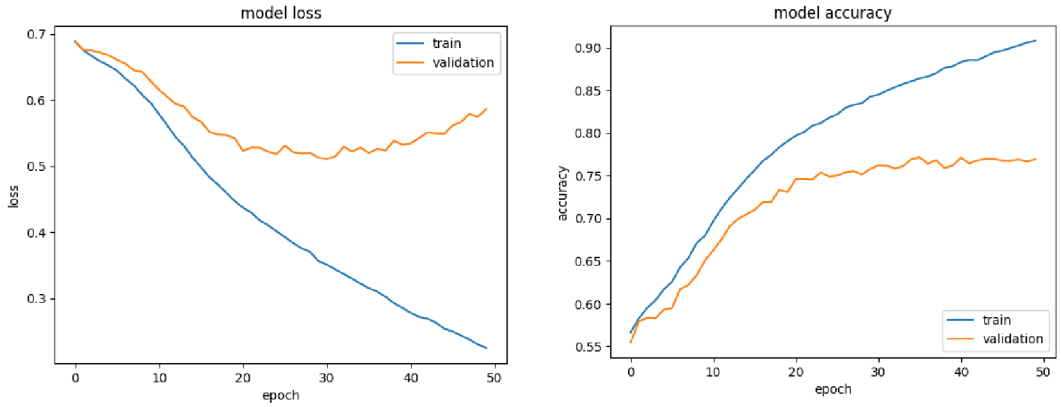


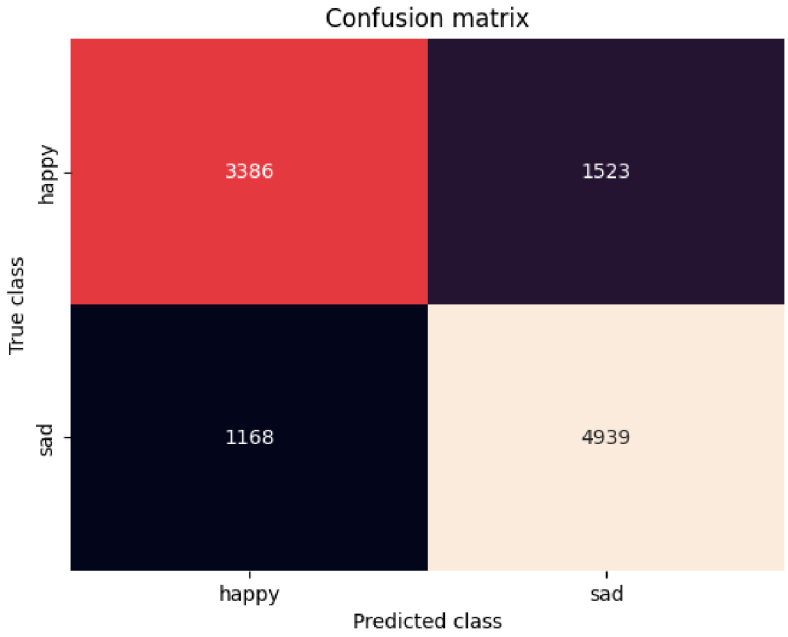


accuracy	precision	recall	specificity	f1-score	AUC
55.719	14.219	51.136	56.367	22.251	0.5359

Table 6.11: Table depicting the loss, accuracy, confusion matrix, and other performance metrics of a neural network with two branches of CNN-LSTM (Model shown in Figure 5.6)

The loss function fluctuated around the same value and then stagnated (results shown in table 6.11). This might be caused during gradient descent by choosing large learning_rate. I changed the learning rate of the Adam optimizer from 0.001 to 0.0001, as in the training of the previous neural networks.





accuracy	precision	recall	specificity	f1-score	AUC
75.572	68.975	74.352	76.431	71.563	0.8289

Table 6.12: Table depicting the loss, accuracy, confusion matrix, and other performance metrics of a neural network with two branches of CNN-LSTM (Model shown in Figure 5.6) after adjusting the learning rate to a smaller 0.0001

It can be seen from the loss function graph shown in table 6.12 that the validation loss keeps increasing while the training loss decreases. I decided to use regularizers to reduce the high variance of the neural networks. I am using L2 regularizers for the weights of every convolutional and LSTM layer of the neural network and a dropout layer with 0.2 after the first convolutional layer and the third convolutional layer in both branches, as can be seen in table 6.3.

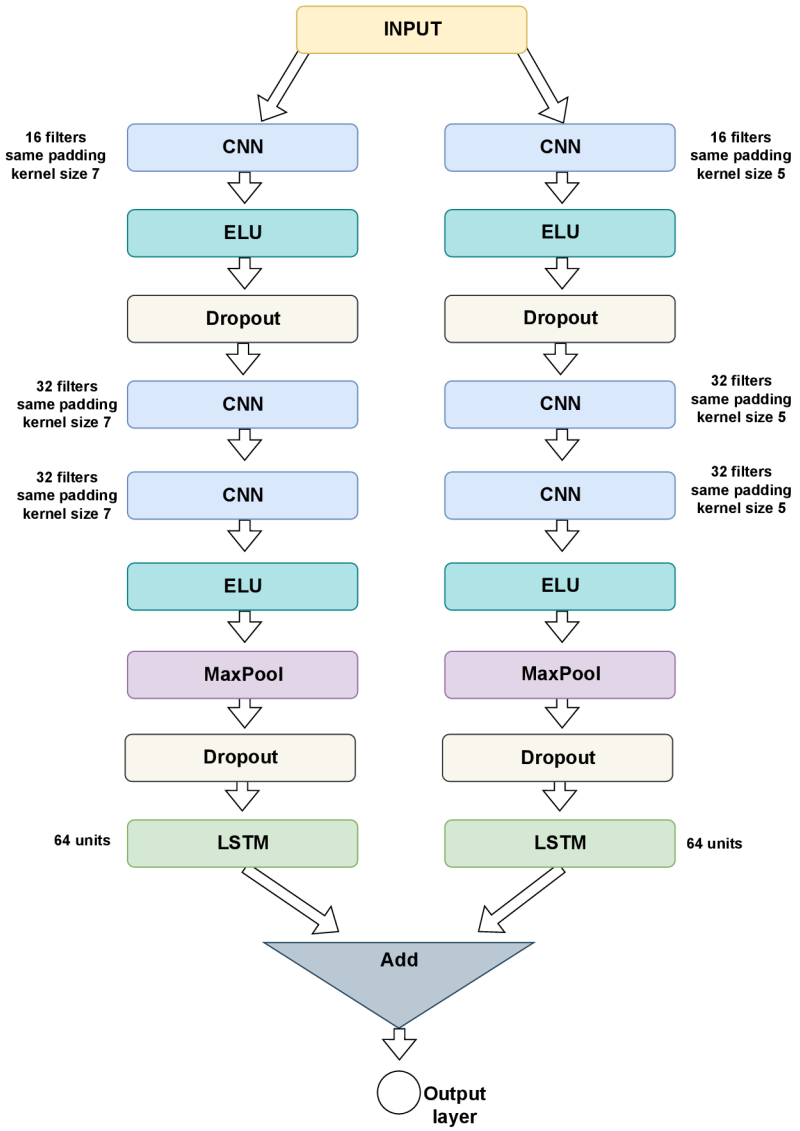
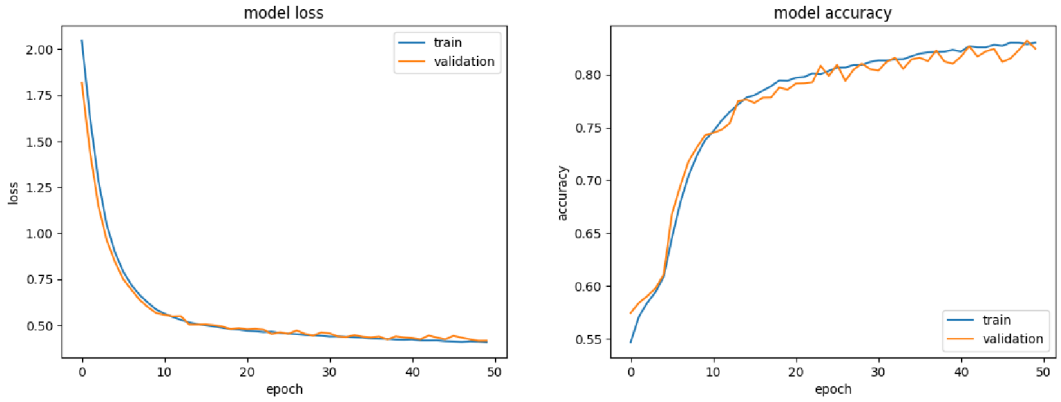
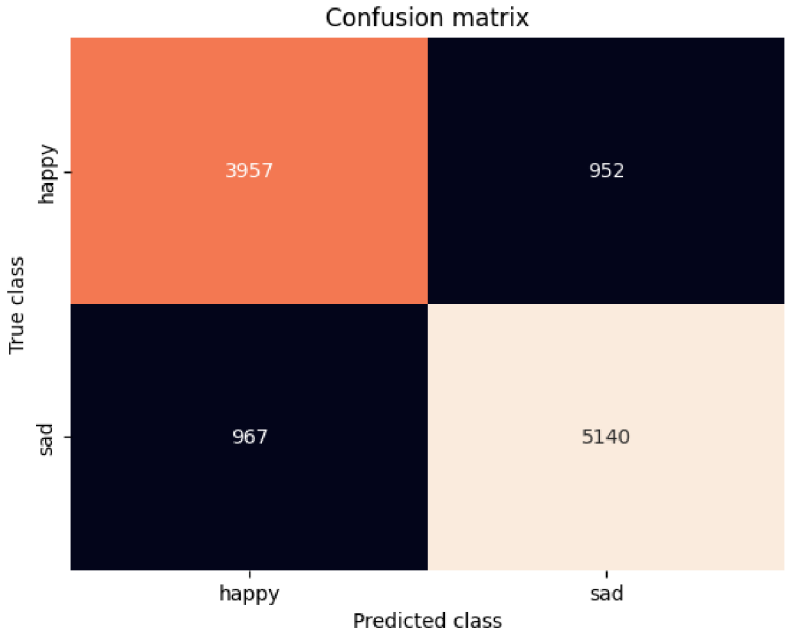


Figure 6.3: Figure depicting the model with two branches of CNN-LSTM layers with added dropout layers

This adjustment resulted in improved loss and accuracy of the neural network (results in table 6.13).





accuracy	precision	recall	specificity	f1-score	AUC
82.580	80.607	80.361	84.373	80.484	0.9077

Table 6.13: Table depicting the loss, accuracy, confusion matrix, and other performance metrics of a neural network with two branches of CNN-LSTM (Model shown in Figure 5.6) after using regularization techniques and learning rate of 0.0001

As the loss function kept decreasing, I trained the neural network for additional 350 epochs. After 150 epochs, the loss function decreased only very slowly. The best achieved accuracy after training for all epochs is an accuracy of 87.309%, precision of 81.809%, recall 88.830%, specificity 86.251%, f1-score 85.175% and AUC score of 0.9465.

This neural network was also trained for the classification problem of recognizing neutral and sad emotions because the best results were obtained with this neural network. Without any adjustment to the architecture, learning rate, or the number of epochs, the achieved results are accuracy of 84.865%, precision of 80.850%, recall 87.095%, specificity 83.031%, f1-score 83.856% and AUC score of 0.9325.

Chapter 7

Conclusion

This thesis studied emotion recognition from EEG using deep learning. EEG, the human brain, and the deep learning methodology were introduced, and different architectures of neural networks were investigated for the problem of emotion classification. These were then implemented using the Keras API, trained on the SEED-IV dataset, and evaluated on unseen data from the same dataset. The examined architectures used LSTM, CNN, and GRU layers.

This thesis aimed to create a model capable of extracting and selecting global features directly from the raw EEG data and classifying emotions. Two binary classification neural networks were trained to classify happy-sad and neutral-fear emotions. The best result for the classification of emotions was achieved with a neural network consisting of two branches with convolutional layers and LSTM layers in each branch with an accuracy of 87.309% for happy-sad and 84.865% for neutral-fear emotions. Both models are good at distinguishing the classes with an AUC curve of 0.9465 and 0.9325 separately which is better than some previous works done on this dataset (as introduced in the thesis).

Further improvements might be achieved by obtaining more data and creating synthetic EEG data since deep learning models can learn more global brain patterns with more data. In the future, the neural network's performance on independent subjects can be studied as the neural network was trained on all of the dataset subjects.

Bibliography

[1] *EEG Signal processing and Feature extraction*. 1st ed. Springer Nature Singapore Pte Ltd., 2019. ISBN 978-981-13-9113-2.

[2] *File:Artificial neural network.svg — Wikimedia Commons, the free media repository*. 2023. [Online; accessed 7-May-2023]. Available at: https://commons.wikimedia.org/w/index.php?title=File:Artificial_neural_network.svg&oldid=753183698.

[3] Alhagry, S., Fahmy, A. A. and El Khoribi, R. A. Emotion recognition based on EEG using LSTM recurrent neural network. *International Journal of Advanced Computer Science and Applications*. Science and Information (SAI) Organization Limited. 2017, vol. 8, no. 10.

[4] Banz, B. C., Yip, S. W., Yau, Y. H. and Potenza, M. N. Chapter 16 - Behavioral addictions in addiction medicine: from mechanisms to practical considerations. In: Ekhtiari, H. and Paulus, M., ed. *Neuroscience for Addiction Medicine: From Prevention to Rehabilitation - Constructs and Drugs*. Elsevier, 2016, vol. 223, p. 311–328. Progress in Brain Research. DOI: https://doi.org/10.1016/bs.pbr.2015.08.003. ISSN 0079-6123. Available at: https://www.sciencedirect.com/science/article/pii/S0079612315001417.

[5] Buduma, N., Buduma, N., Papa, J. et al. *Fundamentals of Deep Learning*. 2nd ed. O'Reilly Media, 2022. ISBN 978-1-492-08218-7.

[6] Carter, R. The Human Brain Book: An Illustrated Guide to its Structure, Function, and Disorders. In:. 3rd ed. Dorling Kindersley, 2019, p. 50–75. ISBN 978-1-4654-7954-9.

[7] Craik, A., He, Y. and Contreras Vidal, J. L. Deep learning for electroencephalogram (EEG) classification tasks: a review. *Journal of neural engineering*. IOP Publishing. 2019, vol. 16, no. 3, p. 031001, [cit. 2023-05-06]. DOI: 10.1088/1741-2552/ab0ab5.

[8] DeepLearningAI. *Activation Functions (C1W3L06)* [online]. August 2017 [cit. 10-04-2023]. Available at: https://www.youtube.com/watch?v=Xvg00QnyaIY.

[9] DeepLearningAI. *Logistic Regression Cost Function (C1W2L03)* [online]. August 2017 [cit. 10-04-2023]. Available at: https://www.youtube.com/watch?v=SHEPb1JHw5o.

[10] DeepLearningAI. *Neural Network Representation (C1W3L02)* [online]. August 2017 [cit. 07-05-2023]. Available at: https://www.youtube.com/watch?v=CcRkHl75Z-Y.

[11] DeepLearningAI. *Why is deep learning taking off? (C1W1L04)* [online]. August 2017 [cit. 07-05-2023]. Available at: https://www.youtube.com/watch?v=xflCLdJh0n0.

[12] Goodfellow, I., Bengio, Y. and Courville, A. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[13] Houssein, E. H., Hammad, A. and Ali, A. A. Human emotion recognition from EEG-based brain–computer interface using machine learning: a comprehensive review. *Neural Computing and Applications*. Springer. 2022, vol. 34, no. 15, p. 12527–12557.

[14] Hu, L. and Zhang, Z., ed. *EEG Signal Processing and Feature Extraction*. 1st ed. Springer Singapore, 2019. ISBN 978-981-13-9113-2.

[15] Kelleher, J. D. *Deep Learning*. 1st ed. The MIT Press, 2019. ISBN 978-0262537551.

[16] Kingma, D. P. and Ba, J. L. Adam: A method for stochastic optimization. In: *3rd International Conference for Learning Representations*. San Diego: [b.n.], 2014.

[17] Koelstra, S., Muhl, C., Soleymani, M., Lee, J.-S., Yazdani, A. et al. DEAP: A Database for Emotion Analysis ;Using Physiological Signals. *IEEE Transactions on Affective Computing*. 2012, vol. 3, no. 1, p. 18–31, [cit. 2023-05-06]. DOI: 10.1109/T-AFFC.2011.15. Available at: https://ieeexplore.ieee.org/document/5871728.

[18] Krohn, J., Beyleveld, G. and Bassens, A. *Deep Learning illustrated*. 1st ed. Addison-Wesley Professional, 2020. ISBN 978-0-13-511669-2.

[19] Kumar, D. K., Nataraj, J. L. et al. Analysis of EEG based emotion detection of DEAP and SEED-IV databases using SVM. *Proceedings of the Second International Conference on Emerging Trends in Science & Technologies For Engineering Systems (ICETSE-2019)*. may 2019. DOI: 10.2139/ssrn.3509130. Available at: https://ssrn.com/abstract=3509130.

[20] Mitchell, T. M. *Machine Learning*. 1st ed. McGraw-Hill Science/Engineering/Math, 1997. ISBN 0070428077.

[21] Nidal, K. and Malik, A. S., ed. *EEG/ERP Analysis: Methods and Applications*. 1st ed. CRC Press, 2013. ISBN 978-1138077089.

[22] Nielsen, M. A. *Neural networks and deep learning*. Determination press San Francisco, CA, USA, 2015.

[23] Opidi, A. and Jha, A. *PyTorch Loss Functions: The Ultimate Guide* [online]. 2023 [cit. 11-04-2023]. Available at: https://neptune.ai/blog/pytorch-loss-functions.

[24] Raschka, S. and Mirjalili, V. *Python Machine Learning*. 2nd ed. Packt Publishing Ltd., 2017. ISBN 978-1-78712-593-3.

[25] Sanei, S. and Chambers, J. *EEG Signal Processing*. John Wiley & Sons Ltd., 2019. ISBN 978-0-470-02581-9.

[26] Sharma, S., Sharma, S. and Athaiya, A. Activation functions in neural networks. *Towards Data Sci.* 2017, vol. 6, no. 12, p. 310–316.

[27] Stanford. *CS231n Convolutional Neural Networks for Visual Recognition* [online]. 2019 [cit. 27-04-2023]. Available at: https://cs231n.github.io/neural-networks-1/#actfun.

[28] Starmer, J. *Long Short-Term Memory (LSTM), Clearly Explained* [online]. November 2022 [cit. 27-04-2023]. Available at: https://www.youtube.com/watch?v=YCzL96nL7j0&t=216s.

[29] Wen, T. and Keyes, R. Time series anomaly detection using convolutional neural networks and transfer learning. *ArXiv preprint arXiv:1905.13628.* 2019.

[30] Yenter, A. and Verma, A. Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis. In: *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON).* 1st ed. New York, NY, USA: IEEE, October 2017, vol. 8, p. 540–546. DOI: 10.1109/UEMCON.2017.8249013. ISBN 978-1-5386-1104-3.

[31] Yin, W., Kann, K., Yu, M. and Schütze, H. Comparative study of CNN and RNN for natural language processing. *ArXiv preprint arXiv:1702.01923.* 2017.

[32] Zheng, W., Liu, W., Lu, Y., Lu, B. and Cichocki, A. EmotionMeter: A Multimodal Framework for Recognizing Human Emotions. *IEEE Transactions on Cybernetics.* 2018, p. 1–13. DOI: 10.1109/TCYB.2018.2797176. ISSN 2168-2267.

Appendix A

Contents of the included storage media

Directory	Description
data/	Directory with the `segment_samples.ipynb` for data preparation of the SEED-IV dataset
fear_neutral/	Directory with the folders of training of different models and `.ipynb` notebooks the models were trained with for the classification of neutral and fear
sad_happy/	Directory with the folders of training of different models and `.ipynb` notebooks the models were trained with for the classification of sad and happy

Table A.1: The directory structure of the included storage media.

The storage media also contains `README.md` file for further describtion of the directory layout.