Faculty of Electrical Engineering, Information Technology and Mathematics (EIM)
Signal and System Theory Group (SST)

Emotion recognition using EEG Signals and Wristband Data

Project Group

Report - Summer Semester 2024

by
Abdur Rafay, Omar Anber, Omar Ubaid, Surya Mani Kumar Jakka
Jatin Yerawadekar, Kushal Rao, Michael Siegmund

submitted to:
Prof. Peter Schreier

supervised by:
Dr.-Ing. Tanuj Hasija
Dr.-Ing. Mohammad Soleymani

Paderborn, October 14, 2024

Abstract.

Emotion recognition using physiological signals is a key area of research in affective computing, with applications ranging from mental health monitoring to human-computer interaction. In this project, we explore emotion recognition by combining electroencephalography (EEG) data, commonly used in brain-computer interface (BCI) systems, with data from wristband sensors. EEG captures electrical brain activity, forming a direct communication pathway between the brain and external devices, while the wristband measures physiological signals such as blood volume pulse, electrodermal activity, and temperature. These multimodal signals are processed to extract relevant features, helping to distinguish between emotional states.

Data acquisition involves a series of experiments where participants are exposed to video stimuli designed to evoke target emotions. In this project, five emotions have been considered, namely: *excitement*, *fear*, *happiness*, *relaxation*, and *sadness*. The EEG data is recorded using the *Emotiv EPOC Flex* headset, while physiological data is captured using the *Empatica E4* wristband. A systematic approach is developed, containing steps such as preprocessing for artifact removal and segmentation, feature extraction covering frequency and time-domain analysis, and classification using machine learning algorithms such as support vector machines and neural networks. Furthermore, dimensional reduction techniques like principal component analysis and linear discriminant analysis are utilized to reduce the large dimensions of the raw data, while retaining critical information. Classifiers are then trained to identify the emotional states from the processed signals. Evaluation of the models is performed using accuracy, precision, and F1-score metrics, highlighting the effectiveness of using EEG and wristband data for emotion recognition.

The Classification models achieves test accuracies ranging from 62% to 85%, with the highest-performing model, the Ensemble (CNN + LSTM + Hybrid), reaching 85% accuracy. This significantly outperforms random chance, which would result in a 20% accuracy in a five-class classification setup, demonstrating the success of the approach.

This report presents data acquisition, the processing pipeline, the methodology regarding feature extraction, dimensional reduction techniques, and classification results. Additionally, the intermediate project report outlines the challenges and future improvements for enhancing emotion recognition using multimodal physiological signals.

Contents

1 Introduction

Emotion recognition is a rapidly evolving field within affective computing, with applications ranging from mental health monitoring to human-computer interaction and adaptive systems. The ability to detect emotional states using physiological signals offers valuable insights into both cognitive processes and emotional responses. Electroencephalography (EEG), which captures electrical brain activity, and wristband sensors, which measure peripheral signals such as blood volume pulse (BVP), electrodermal activity (EDA) and temperature, are two primary sources of data that can be used to decode emotions. Therefore, this project explores the fusion of EEG and wristband data to classify various emotional states. By leveraging both EEG and peripheral data simultaneously, the goal is to create a system that improves the accuracy and robustness of emotion recognition.

The project follows a systematic approach, beginning with the selection of hardware and acquisition of data. The *Emotiv EPOC Flex* headset is used to record EEG data, capturing brainwave activity across 32 channels, while the *Empatica E4* wristband monitors peripheral physiological signals. In a series of controlled experiments, participants are exposed to video stimuli designed to evoke specific emotions. Synchronized data collection from both devices is ensured, allowing for a multimodal analysis of the emotional responses. The data acquisition process and the hardware involved are discussed in chapter 2, where the devices and the experimental setup are introduced. Furthermore, an automated approach using a self-made graphical user interface (GUI) is employed.

Processing the recorded data is covered in individual chapters for each device. This has been done to focus on the specific traits of the hardware and data before combining them. Consequently, the project group has been divided into two subgroups, one for each device.

Chapter 3 delves into the detailed processing and classification of EEG data, beginning with the acquisition of raw signals. Preprocessing techniques are employed to remove artifacts and noise, ensuring cleaner data for further analysis. Afterward, the EEG recordings are synchronized with experimental timestamps to facilitate segmentation, which paves the way for feature extraction. A combination of statistical analysis and frequency-domain methods is applied to transform the raw EEG signals into meaningful metrics. To manage the high dimensionality of the data, dimensionality reduction techniques such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are used before feeding the data into machine learning classifiers, which are discussed in detail in Section 3.5.

Support Vector Machines (SVMs) are extensively used in EEG-based emotion recognition due to their effectiveness with high-dimensional datasets. For example, Duan et al. (2013) achieved an accuracy of 83% using SVMs with differential entropy features for

classifying emotions like joy, sadness, and fear [DZL13]. Similarly, Atkinson and Campos (2016) reported accuracies of up to 84% with kernel-based SVMs and feature selection techniques [AC16]. Additionally, the k-Nearest Neighbors (kNN) classifier demonstrated effectiveness in EEG applications, achieving 90.08% accuracy in Brain-Computer Interfaces and 83.26% in emotion classification [SFJI20]. Complex classifiers, such as neural networks, are also explored to enhance emotion recognition accuracy.

In Chapter 4, the wristband data is processed, focusing on physiological information such as blood volume pulse (BVP), electrodermal activity (EDA), and temperature. The approach presented is comparable to the solution discussed in chapter 3 regarding EEG data. However, an automated pipeline is deployed for data handling up to the feature extraction stage, allowing for higher volumes of data. Its flexible architecture enables easy addition of functionality and simplified reconfigurability. Afterward, key features are extracted, and dimensionality reduction and classification techniques are applied to the wristband data.

Finally, the project's key findings are summarized in Chapter 5. The general approach is reflected upon, and limitations are highlighted. Additionally, as there is still room for improvement in capturing emotional variations, future work is discussed, focusing on the individual shortcomings of the current approach and exploring potential additions such as new devices or more advanced techniques. Overall, this project demonstrates that the fusion of multimodal physiological data is a promising option for enhancing emotion recognition.

2 Fundamentals

This chapter provides an overview of the process for obtaining data to be analyzed in subsequent chapters, such as chapter 3 and chapter 4, to sense emotional states. First in section 2.1 the devices used are presented. In section 2.2, a basic model for distinguishing between various stimuli is presented, along with a description of the types of experiments conducted. Additionally, the experimental workflow will be covered which has been automated. Finally, the automation approach implemented through a graphical user interface will be shown. To facilitate better work distribution, third-party datasets were utilized in the early stages and are referenced in this chapter.

2.1 Devices Used

Gathering data, adequate for sensing emotional states, can be done by many different devices. In this project, an electroencephalography headset and a wristband comparable to an older smartwatch are used. During experiments, these devices record at the same time to extend each other, collecting as much data from different sensors and thus potentially useful information as possible.

2.1.1 Electroencephalography (EEG) Headset

The EEG headset used for this project is the *Emotiv EPOC Flex*, a highly adaptable EEG platform for scientific research and development [WMdW+20]. The 32 EEG channels and 2 reference channels (CMS and DRL) are set up based on the international 10-20 system, providing extensive spatial resolution for comprehensive neural signal acquisition through the subject's scalp [Emo23]. Fig. 2.1a highlights the full Emotiv EPOC Flex package comprising headgear with 32 EEG electrodes connected to flexible sensor placement positions, a controller, the USB receiver for connectivity with the end device, and a charging cable. An illustration of wearing the headset with the fitted receiver during the experiment is shown in Fig. 2.1b. The successful implementation of this EEG-based system relies on the precise configuration of both hardware and software components, ensuring optimum data collection and signal quality.

Hardware: The electrodes interfaced with the controller are securely integrated into the cap and operate at a rate of $128\,{}^{\mathrm{Sa}}/_{\mathrm{s}}$. The 32-channel configuration ensures comprehensive and symmetrical scalp coverage, with 16 electrodes for EEG positioned on both sides of the head. The common-mode-sense (CMS) input and driven-right-leg (DRL) output are located bilaterally to enhance signal quality by minimizing common-mode noise and interference. The USB receiver connects to the computer to establish wireless communication with the headset, facilitating EEG signal detection.

(a) Emotiv Cap with Electrode Sensors, Controller, USB Receiver, and charging cable

(b) The EEG headset in use

Figure 2.1: *Emotiv EPOC Flex* EEG headset. Images sourced from the Mindtec Store [Min23].

Software: In this project, *Emotiv Pro* official software is employed to provide real-time feedback on the contact quality of the electrodes with the subject's scalp. It also enables the assessment of EEG signal quality for each electrode, a critical step to ensure proper data collection. During the hardware setup on the subject, Emotiv Pro is used manually to confirm appropriate contact quality before collecting data. The EEG data acquisition process is covered in detail in Chapter 3.

2.1.2 Wristband

The wristband used in this project is the *Empatica E4* device, which can be seen in figure 2.2. It has been announced in 2014 and its official support will be discontinued in February 2025, which will be before the end of the second half of this project [Emp18]. Compared to its successor *Empatica EmbracePlus* [Emp22] or alternative new devices such as the *EmotiBit* [Con21], the E4-wristband mainly falls short in terms of sensory

(a) Frontview [Emp18].

(b) Backside of the used device showing all relevant sensors.

Figure 2.2: Wristband *Empatica E4*.

Sensor	Speed	Range	Resolution
Accelerometer (ACC)	$32\,{}^{\mathrm{Sa}}/_{\mathrm{s}}$	$\pm 2\,{}^{\mathrm{m}}/_{\mathrm{s}^2} \ldots \pm 8\,{}^{\mathrm{m}}/_{\mathrm{s}^2}$	$8\,{}^{\mathrm{mm}}/_{\mathrm{s}^2} \ldots 30\,{}^{\mathrm{mm}}/_{\mathrm{s}^2}$
Photoplethysmography (PPG)	$64\,{}^{\mathrm{Sa}}/_{\mathrm{s}}$	-	0,9 nW
Electrodermal Activity (EDA)	$4\,{}^{\mathrm{Sa}}/_{\mathrm{s}}$	10 nS... 100 µS	0,9 nS
Temperature (TMP)	$4\,{}^{\mathrm{Sa}}/_{\mathrm{s}}$	−40 °C... 115 °C	0,02 °C
Battery (BAT)	-	0 %... 100 %	1 %
Tag-Button	-	0 ... 1	-

Table 2.1: Specifications of the E4-wristband according to [Emp14].

resolution, and additionally lacks a humidity sensor or magnetometer. More precise specifications of sensors available on the E4 device can be seen in table 2.1.

For the experiments in this project the PPG sensor also called BVP sensor, EDA sensor and temperature sensor are relevant. The only actual sensor left unused is the accelerometer as the experiments conducted require the wristband to be stationary, which ideally would yield a constant output for this sensor. Regarding EDA data two silver coated electrodes at the actual band are pressed against the skin of the wearer. At these electrodes a 8 Hz sinusoidal signal with up to 100 µA peak to peak current is applied and the resulting voltage is measured yielding the resistance of that specific part of skin. Temperature data is acquired with a regular infrared thermopile sensor [Emp14]. To gather BVP data, the E4 device uses two sets of a red LED, a green LED and a photo diode each. Although there are two receiving elements, which could be used as two separate channels, they are internally used for the same and only channel. The main idea of photoplethysmography is that especially the light of the green LED is prone to be absorbed by oxygenated blood, meaning that during a heart pulse cycle the amount of reflected light depends on the volume of blood per moment. Measuring the amount of reflected light over time ideally yields an aortic pressure curve also referred as pulse curve [Emp20a]. This kind of measurement can be used even at rather big distances to the heart like fingertips or toes. Using fingers or toes might decrease the amplitude of the frequency components of interest but the characteristic form of the curve stays about the same. However, in a realistic scenario the amount of blood at the wrist is not just influenced by the heart alone. Fast arm movement or even moving a finger via its tendon does impact the measurement at the wrist. Depending on the kind of movement the resulting artefacts might make guessing the pulse curve impossible [MMG24]. Furthermore, the E4-wristband uses an internal algorithm for zero-meaning the BVP data, which eliminates the overall pressure and inverts the remaining function [Emp20a][Emp20b]. The details of Empaticas algorithm, e.g. how many previous data points are used for it or whether distort the output data in some way, are unknown to users. More details about that and the resulting data in general will be explained in chapter 4.1.

To get actual data which is usually stored in *.csv* files there are three options:

1. *E4 realtime*: Default App for Android or iOS
2. *E4-Manager*: Desktop Client for Windows or Mac OS X
3. *Streaming Server*: API based tool for Windows

Regardless of which option is used, every session can be inspected on and downloaded from their website *Empatica connect* [Emp24]. Additionally the *Streaming Server* application allows for real time data collection and is thus used in this project regarding the E4-wristband, as it is the only viable choice for an automated approach. Using *Streaming Server* allows to connect even multiple E4 devices at once using a specific Bluetooth dongle. It opens a virtual server on the local network to which a user has to connect. In this project this has been done using the *socket* package provided by *Python*.

2.2 Data Acquisition

Getting data in an appropriate way is the most important part of this project. General options regarding experimental types reign from physical exercises over subjects fulfilling certain social interactions to basic presentations. In this instance videos being presented to the subjects under evaluation has been chosen for multiple reasons. Physical exercises are not useful for emotion sensing as the main stimuli comes from the physical activity itself, overshadowing certain emotional reactions. Specific social interactions need certain preparations while videos can been shown from a laptop that is required to record data anyway. Using videos keeps the overhead for testing subjects small resulting in more time for actual tests. Another important reason is the ability to reproduce conducted experiments as precisly as possible. Unlike specific social tests, showing videos only requires specific surroundings and position. The final reason for deciding against any activity that involves movement is data quality, especially regarding BVP data. Any movement can lead to artefacts, meaning the possibility of wearing the wristband on the non-dominant arm, which can rest flat on a desk while watching videos, should yield the maximum amount of usable data.

2.2.1 Valence-Arousal Model

Observing different emotions is the key quality of this project. There are many different types of plots and systems used to categorise different stimuli in humans, developed over the last decades. One of the simpler and still most frequently used options is the *valence-arousal model* [Nef24][PRP08]. The valence-arousal model uses just two axis to determine between different emotions which are the arousal axis and the valence axis. The arousal axis is used to describe the intensity of a certain emotional state while the valence axis is used to describe whether an emotional state is pleasant or not [PRP08]. For example, excitement is often referred to by saying "I can't wait to visit or present xyz", indicating an intense emotional state. However the experience being under emotional stress over some time is not necessarily pleasant. Happy on the other

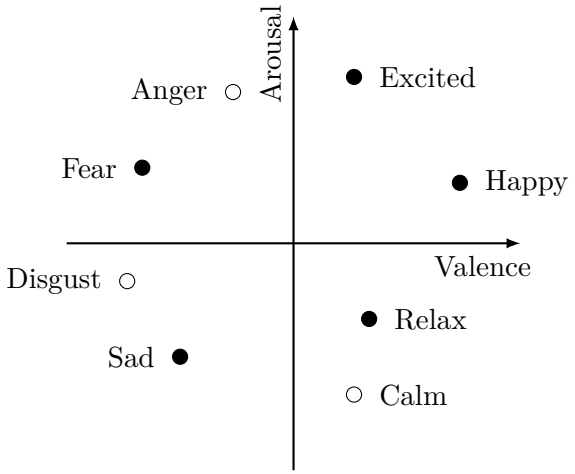


Figure 2.3: Simplified Valence-Arousal Model. Used emotions are shown as solid points. It is based on [Nef24].

hand is much more pleasant while not stressing the subject like excitement does, thus happy is located at a high valence position, but only slightly above the middle ground of arousal. Both examples alongside others are located in figure 2.3. The right half shows more positive emotions, while the left half shows the negative ones. The upper half contains stronger stimuli whilst the lower half contains the weaker ones. In this project all quadrants are covered as shown in figure 2.3.

2.2.2 Experiments

As mentioned earlier the experiments of choice consist of videos to keep testing simple and efficient. More specifically videos used are music-videos taken from the *DEAP* dataset, which is one of the biggest public available options already used in many studies comparable to this project [KMS+12]. DEAP is a database for emotion analysis that uses 120 different videos for 30 different emotions. A detailed list has been provided, containing the anticipated stimuli, the youtube link and detailed valence-aroussal raiting for each individual video. Additionally, access regarding data recorded in their experiments, using EEG, physiological peripheral and a face camera, has been granted. For this project so far only five target emotions have been used covering each sector of the valence-arousal model:

Emotion	Excited	Fear	Happy	Relaxed	Sad
Induced by	Video	Video	Video	Blank screen	Video

Table 2.2: Used emotions and their method of being induced during experiments.

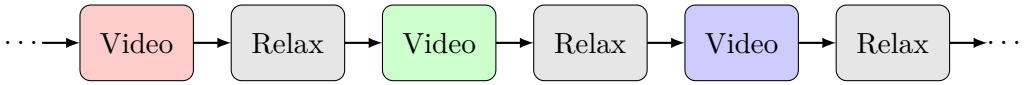


Figure 2.4: The general video flow used in this project.

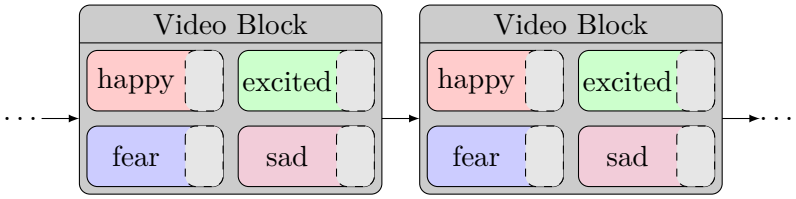


Figure 2.5: Video Blocks containing a video for each emotion + the relax phase in gray after each video.

Relaxed is considered the neutral point thus instead of videos a blank screen with a countdown is shown. Based on the valence-arousal model, videos of different emotions being in close proximity to the remaining four chosen emotions will also be used for them to maximize the amount of testing data. More details about how this has been done is explained in appendix A.1.

Each subject will be able to watch each video exactly once as re watching yields less of an emotional reaction and thus worse data quality. Videos will be shown in a specific pattern and after each video there will be a relaxation phase as shown in figure 2.4. The relaxation phase is needed as videos targeting the same emotion will not been played after each other. To ensure minimal interference of different stimuli subjects need to rest between videos. To ensure about equal amounts of data of each desired emotion even when stopping experiments prematurely, videos will be shown in blocks as sketched in figure 2.5. Each block contains videos of the four targeted emotions exactly once. The sequence in which the stimuli appear, just as the specific video shown, is randomly selected. For example, after watching the first *happy* video, the next *happy* video can only appear, after the first video of the other three stimuli has been watched. Stopping an experiment after a block guarantees equal amounts of videos shown for each targeted emotion.

However, the length of the individual videos differs quite a bit ranging from about 2 to 6 minutes. The average video length is slightly below 4 minutes which compared to $4 * 60\,\text{sec} = 4\,\text{minutes}$ of relax time per block seems quite reasonable. Taking cut-away overhead into account this changes however. As an induced stimuli might continue on beyond the specific video inducing it, the beginning of any subsequent video or relax phase is deemed contaminated. For example the sweat activity after a *fear* video can not just disappear or stop in an instant just because the inducing video ended. Consequently the start of the relaxation phase after this fear inducing video will have a higher sweat activity compared to the expected values associated with relaxation. To prevent this

kind of contaminated data from being used later, the leading part of each video or relax phase will be cut away. However, this affects the relax phase four times as often as it is always placed between videos. More details about which parts are used is described in chapter 4.2.2.

2.2.3 Graphical User Interface

To automate the data recording, we implemented a web application. The web application is developed using a *Flask* [Pal24] back-end which is based on *Python* [Fou24] and a simple user interface which is rendered using a templating engine and basic *HTML* [Cor24b], *CSS* [Cor24a], and *JavaScript* [Cor24c] code. The web application is hosted locally on one single device where all the experiments are conducted to ensure all the data is recorded and stored at one location. However, the web application can easily be cloned and set up on any other device or hosted on the cloud for online access. Figure 2.6 shows the simplified flow of the GUI which consists of three parts.

The initial landing page is shown in figure 2.7. It captures the basic user details like *name*, *age*, *gender*, and whether they consent to have their data stored and accessed for research. Once submitted the data is stored in an *SQL* [Cor24d] database, which is queried when onboarding new users to ensure no duplicates. Clicking on submit leads to the next page, checking the connected devices.

The GUI is interfaced with the wristband and the EEG devices via Bluetooth, and once the user details are submitted the GUI checks for the quality of the connection for both devices. In the case of the wristband for example the UI calls a Python script that records data for multiple seconds, ensuring that at least two seconds containing data for every relevant sensor are present. The script then returns the number of samples it received for those two seconds divided by the number of samples it expected. If the connection quality is poor, meaning less than 95% of the expected amount of data per time interval has been received, an error message is shown with a retry button. If the connection quality is okay, the user can proceed to the video player screen. Both potential results are shown in figure 2.8. By clicking on *Start Tests* a user transitions to the video player which starts the recording of the wristband and EEG signals by invoking a subprocess each, handling the data recording till the main UI process is terminated.

For the video player screen, a video is chosen randomly from the DEAP-dataset, while ensuring all emotion classes are equally represented. Once the user plays the video the start time is recorded as a timestamp along with the title of the video currently playing.

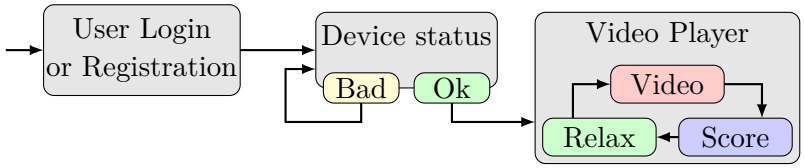


Figure 2.6: Simplified GUI flow

This is to ensure that in case the user resumes an incomplete session in the future, the same set of videos is not displayed again. Once the video finishes playing, the stop timestamp is recorded as well. The captured timestamps, video title and other details are stored in the *session.csv* file that is unique to each user based on an internally handled unique user ID.

Upon completion of a video, the user is shown a feedback page. The feedback page is used to capture how strongly the user felt the intended emotion from the video and can be used to filter the data based on the rating assigned by the user. On submitting the rating the user is shown an empty page with a 60 s countdown to relax. The time stamps for the relaxation period as well as the feedback score by the user are also recorded and stored in the user's *session.csv* file.

After the relax timer expires, the user has the option to proceed, which repeats the

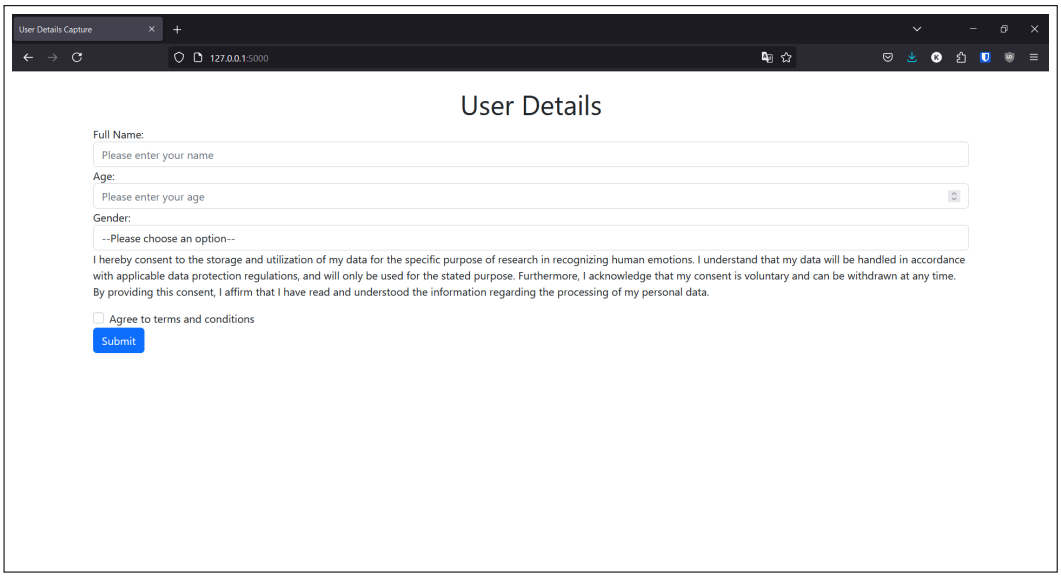
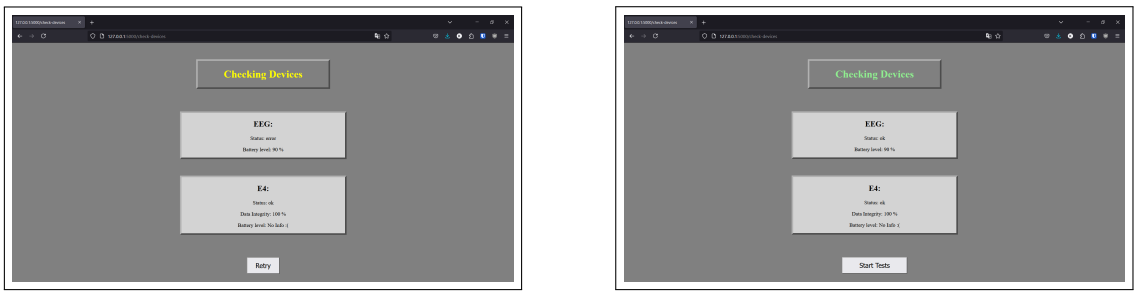


Figure 2.7: Landing page.

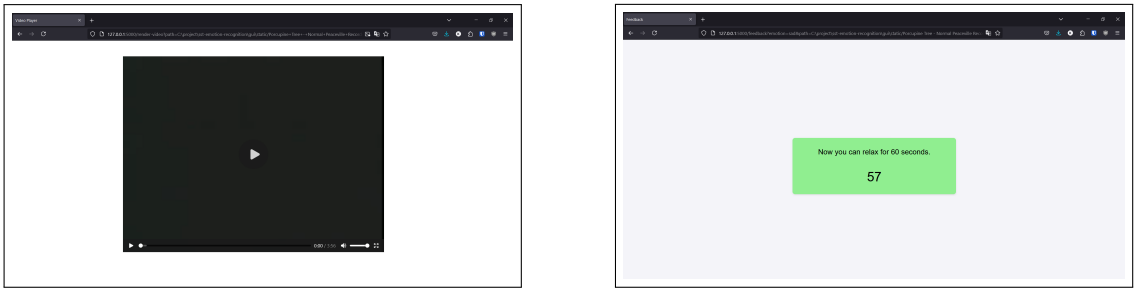


(a) Status check failure

(b) Status check success

Figure 2.8: Status check page.

video-score-relax cycle, shown in figure 2.6, for a new video. If the user chooses to end the session, a controlled exit process is triggered to ensure that all data from the wristband and EEG is successfully saved in *.csv* files before closing the GUI. Stopping the GUI successfully yields a user file, raw wristband data files for each sensor, and a raw EEG data file. Raw wristband data files of multiple sessions can be stored in the same folder as they are named based on the timestamp when the recording has started, e.g. *bvp_raw_1725470399.csv*. Recording multiple sessions does not yield multiple user files as if one already exists a new session is appended to the existing file.



(a) Video Player Page. (b) Relax Page.

Figure 2.9: Feedback and relax screens.

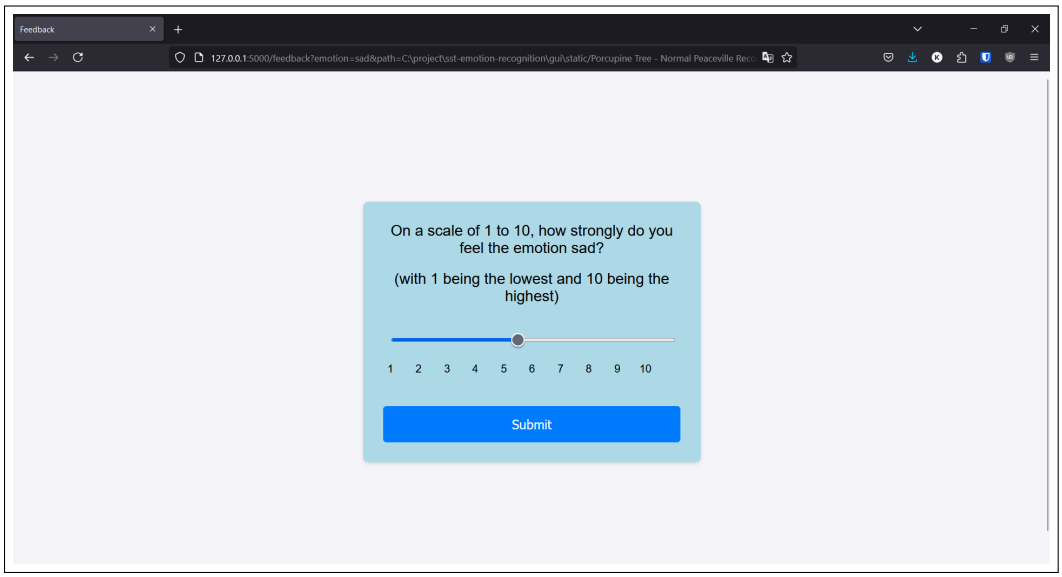


Figure 2.10: Feedback Page.

2.3 Recorded Dataset

This section is used to give an overview of the sessions held to acquire data. The gathered data alongside some data from the DEAP dataset is used for every analysis, example or result in chapter 3 and chapter 4. There has been a major session using six participants, multiple mid semester sessions using two to four participants and some individual initial test sessions which have been recorded using Empaticas website. Apart from the initial test sessions all data has been acquired using the automated approach presented in the previous section 2.2.3. Some sessions have been corrupted due to for example Excel converting timestamps from 1722439228653 to 1.72243E+12, which means the more detailed information is lost. Others are left out because the wristband data is unusable as the wristband has not been worn properly. Sessions where at least the wristband data is usable are listed in the following tables:

Date	Gender	Age	Videos	Relax	Video time	Relax time	Devices
14.06.2024	male	23	4	4	13 min	4 min	WB (website)
14.06.2024	male	24	8	7	22 min	7 min	WB (website)
12.07.2024	male	23	5	5	19 min	5 min	WB
02.08.2024	male	29	5	5	21 min	5 min	WB
04.09.2024	male	31	9	9	34 min	9 min	EEG + WB
04.09.2024	male	26	10	9	33 min	9 min	EEG + WB
04.09.2024	male	29	9	9	31 min	9 min	EEG + WB
04.09.2024	male	27	8	8	23 min	8 min	EEG + WB
04.09.2024	male	23	8	8	34 min	8 min	EEG + WB

Table 2.3: Usable recorded sessions using at least the wristband device.

Date	Gender	Age	Raw segments	Segments after filter	Acceptance ratio
14.06.2024	male	23	23	1	4 %
14.06.2024	male	24	44	0	0 %
12.07.2024	male	23	36	35	97 %
02.08.2024	male	29	40	37	93 %
04.09.2024	male	31	63	53	84 %
04.09.2024	male	26	62	54	87 %
04.09.2024	male	29	57	57	100 %
04.09.2024	male	27	43	41	95 %
04.09.2024	male	23	66	66	100 %

Table 2.4: Stats taken from the wristband pipeline. Used precut duration and segmentation length are 30 s and the acceptance threshold is set to 10 %.

3 EEG

In this chapter, we explore the steps involved in processing EEG data for emotion recognition. The data acquisition section outlines the setup and configuration of the EEG equipment, followed by the preprocessing, feature extraction, dimensionality reduction, and classification methods used to interpret the EEG signals.

In section 3.1, we begin by detailing the setup for EEG data collection, using the Emotiv EPOC Flex headset. The software tools employed—Emotiv Launcher, Pro software, and Cortex API—are crucial for ensuring proper connection and signal quality. The data collection process follows a systematic approach where participants engage with video stimuli, and the resulting EEG data is recorded for further analysis.

The preprocessing part in section 3.2 includes segmentation and alignment to match timestamps between EEG signals and user session logs. After which, the data is stacked across all participants to create a generalized dataset. It is then followed by artifact removal using a bandpass filter to eliminate unwanted noise and interference.

In section 3.3, involves transforming raw EEG signals into meaningful metrics that can be used for classification. We use windowing techniques to divide the signal into smaller segments and calculate statistical features such as mean, variance, and power spectral density (PSD). Additionally, frequency-domain analysis and differential entropy are computed to capture the underlying emotional states.

To reduce the complexity of the data, in section 3.4, we apply dimensionality reduction techniques like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). PCA reduces the feature set while retaining most of the data variance, and LDA maximizes class separability, ensuring the most relevant features are kept for classification tasks.

In section 3.5, for the classification of emotional states, we implement various machine learning models, including Support Vector Machines (SVM), Deep Neural Networks (DNN), and hybrid models. These classifiers predict emotional categories based on the features extracted from the EEG signals. The performance of these classifiers is evaluated using metrics like accuracy, precision, and F1-score, and improvements are made through techniques such as data augmentation and ensemble methods.

Finally, in section 3.6, we evaluate the performance of the classifiers using confusion matrices and classification reports. These evaluations help us assess the effectiveness of our methods in distinguishing between emotional states.

3.1 Data Acquisition

The data acquisition pipeline is a component that involves several steps of collecting and preparing EEG data for Emotion recognition. This section outlines the detailed procedure to gathering EEG data using the Emotiv headset and associated software tool, which are employed in this project. The data acquisition process is organized in the following: data collection setup, software and applications and data collection process. Further detailed step of data acquisition process explained in the sections below.

3.1.1 EEG Data Collection Setup

The initial step in the data acquisition process involves setting up the necessary hardware and software. For this experiment, the Emotiv Headset, Emotiv launcher and Cortex API are used. The Emotiv headset, represented in Fig. 2.1 Emotiv EPOC Flex EEG headset. Serves as the primary device for EEG data collection.

3.1.2 Software and Applications

Emotiv Launcher and Pro Software:

The Emotiv Pro software is used to configure the headset, monitor and adjust the contact quality and EEG quality of the headset by adjusting the electrodes. The Emotiv Launcher is an application that displays the device's connection status, indicating whether it is connected. In this experiment, the device is connected via Bluetooth. Fig. 3.1 presents the Emotive launcher and Fig. 3.2 shows the Emotiv Pro software.

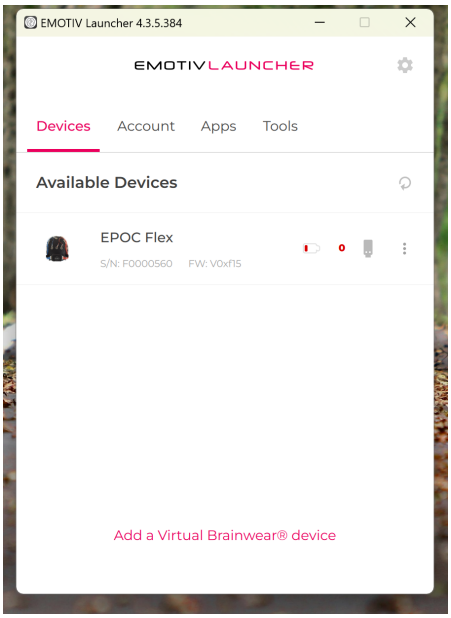


Figure 3.1: The Emotiv Launcher.

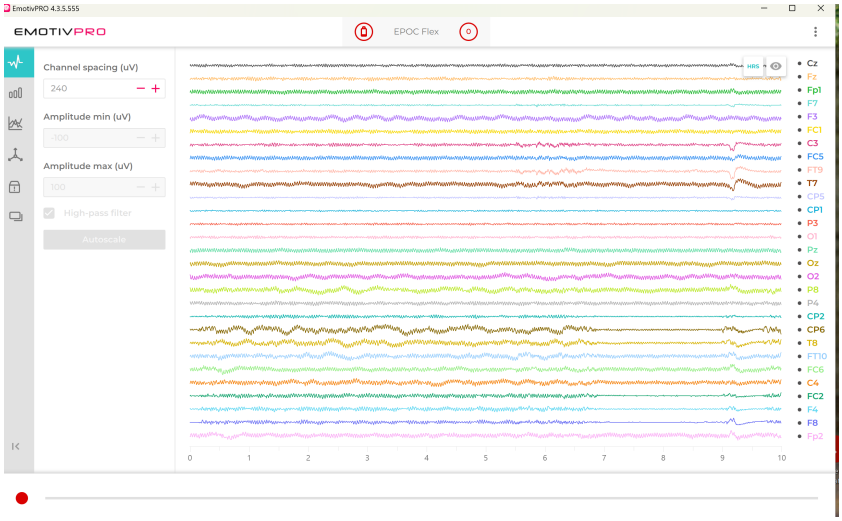


Figure 3.2: The Emotiv Pro software.

Cortex API:

The Cortex API is a binding function. The primary goal is to integrate with the GUI being developed to initiate the data collection process. The Emotiv gitbook provides information on Cortex API binding functions. Using Cortex API, we develop automated functions to integrate the GUI with the Emotiv EPOC Flex device along with a wristband. This setup allows data to be collected simultaneously from both the EPOC Flex device and the wristband, ensuring that the data remains perfectly synchronized when the GUI is launched.

3.1.3 Data Collection Procedure

The data collection process requires the Emotiv EPOC Flex device, alongside the Emotiv launcher and cortex API, to function correctly. The following steps outline the procedure for collecting EEG data.

Verification of Device:

Once the Cortex API is successfully integrated with GUI, and Emotiv headset is functioning correctly, we need to verify the connection through the Emotiv launcher. This is the first step before beginning data collection. We must ensure that the headset is properly connected via the Emotiv launcher.

Contact Quality Check:

After confirming the connection, the next step is to use the Emotiv pro software to ensure the headsets contact quality is perfect. The electrodes must be configured as required. Once the headset contact quality reaches a certain threshold, typically around

50 to 70 percent, we are ready to start data collection. Contact Quality is represented in Fig. 3.3.

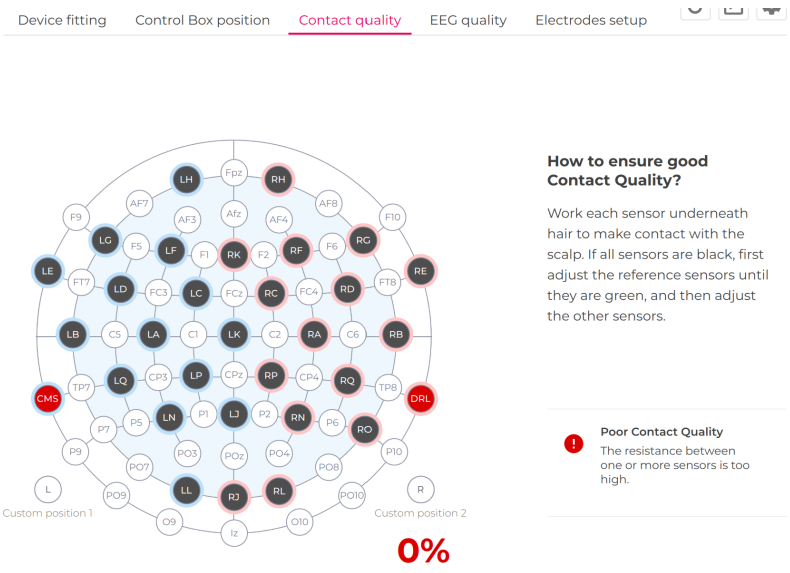


Figure 3.3: Contact Quality.

Data Collection Process:

Once the contact quality is confirmed, data collection can be started. During this phase, Participants engage with video stimuli, as specified in section 2.2.2. Upon completion of the session, the EEG data and the user session CSV file are stored in CSV format within a designated folder. This step concludes the data collection process for this part. The EEG data collected consists of columns with timestamps and channel data, saved as a CSV file. Once the GUI is stopped, the data is automatically written back into this CSV format, where each channel's timestamps and corresponding EEG signals are recorded. A sample of CSV is shown in Fig. 3.4.

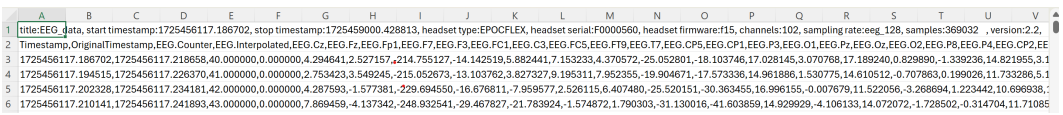


Figure 3.4: The CSV format of an EEG signal.

3.2 Data Preprocessing

Before analyzing the EEG data, several preprocessing steps are applied to ensure data quality and mitigate noise. the first and for most was data segmentation and alignment.

3.2.1 EEG Data Segmentation and Alignment

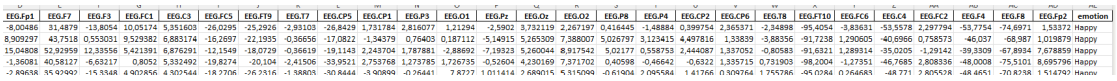
Post collection, the raw EEG data undergoes segmentation,which is an important preprocessing step required for organizing the data for further analysis. This segmentation and merging of the data is done based on timestamps logged during the recording of EEG data while the GUI is running.

The GUI not only captures EEG data, but also generates an additional CSV file containing timestamps and corresponding emotional labels. However, there are different time formats between the two datasets: the user session CSV file records timestamps in milliseconds, where as EEG signals timestamps are in seconds. To address this issue, the timestamps in the user session CSV file are first converted in seconds from milliseconds. This conversion ensures the two data set are good for merge and segmentation.

Once the timestamps from the user session file are converted, the next step is to compile and match the aligned timestamps from both user sessions and the EEG data. This process involves identifying the nearest corresponding timestamps from each file. For example, if the EEG data has a timestamp of 1725467505,75314 and the converted user session data 1725467505,75318, we use these closest matching timestamps to combine the relevant emotional label with EEG data. This process involves matching the nearest timestamps from files and associating the relevant emotion labels with the corresponding EEG data.

The result is a new excel file of segmented EEG signals with Emotion labels, providing a perfect dataset for further analysis, as it establishes a clear relationship between the EEG data and the recorded emotions, enabling more precise interpretations.

We have organized the segmentation and merge of data such that, for example if we have EEG data from various participants saved in a folder, we read all the files accordingly, then create a new excel file along with emotion labels as shown in Fig. 3.5. These files are used for the next steps, including data preprocessing, feature extraction, and classification.



EEG.Fp1	EEG.F7	EEG.F3	EEG.FC1	EEG.C3	EEG.FC5	EEG.FT9	EEG.T7	EEG.CP5	EEG.CP1	EEG.P3	EEG.O1	EEG.Pz	EEG.Oz	EEG.O2	EEG.P8	EEG.P4	EEG.CP2	EEG.CP6	EEG.T8	EEG.FT10	EEG.FC6	EEG.C4	EEG.FC2	EEG.F4	EEG.F8	EEG.Fp2	emotion
-8,00486	31,4879	-13,8054	10,05174	5,351603	-26,0295	-25,2926	-2,93103	-26,8429	1,731784	2,816077	1,21294	-2,5902	3,732119	2,267197	0,416445	-1,48884	0,399754	2,365371	-2,34898	-95,4054	-3,83631	-53,5578	2,297794	-53,7754	-74,6971	1,53372	Happy
8,909297	43,7518	0,553031	9,529382	6,883174	-16,2697	-22,1935	-0,36656	-17,0822	-1,34379	0,76403	0,187112	-5,14915	5,265309	7,388007	5,026797	3,123415	4,497816	1,33839	-3,88356	-91,7238	1,290605	-40,6966	0,758573	-46,037	-68,987	1,019879	Happy
15,04808	52,92959	12,33556	5,421391	6,876291	-12,1549	-18,0729	-0,36619	-19,1143	2,243704	1,787881	-2,88692	-7,19323	5,260044	8,917542	5,02177	0,558753	2,444087	1,337052	-0,80583	-91,6321	1,289314	-35,0205	-1,29142	-39,3309	-67,8934	7,678859	Happy
-1,36081	40,58127	-6,63217	0,8052	5,332492	-19,8274	-20,104	-2,41506	-33,9521	2,753768	1,273785	1,726735	-0,52604	4,230169	7,371702	0,40598	-0,46642	-0,6322	1,355715	0,731903	-98,2004	-1,27351	-46,7685	2,808336	-48,0008	-75,5101	8,695796	Happy
-2,89658	35,92902	-15,3348	4,903856	4,507544	-18,7706	-26,2516	-1,38805	-30,8444	-3,90899	-0,76441	7,8777	1,011414	2,680015	5,515099	-0,61904	2,095584	1,41766	0,509764	1,755786	-95,0784	0,566683	-48,771	2,805538	-48,4651	-70,8238	1,514792	Happy

Figure 3.5: After segmentation the CSV file.

3.2.2 Data Stacking

To enhance the reliability of results and reduce participant-specific bias, EEG data from all six participants are stacked, creating a more generalize dataset. This approach ensures the model could identify emotions across different individuals, increasing the diversity of brainwave patterns associated with each emotion and making the system more robust.

As illustrated in Fig. 3.6, the distribution of the five core emotion representation across the dataset, which is crucial for effective model training.

Each participant is exposed to emotional stimuli designed to evoke five core emotions: sadness, happiness, excitement, relaxation, and fear. The goal is to preprocess, explore, extract features, and classify these emotional states based on EEG readings during the stimuli.

The combined dataset serves as the foundation for further exploration, feature extraction and classification tasks. Consolidating all the data into a single file simplifies the application of machine learning models, data visualization, and insight extraction from EEG readings.

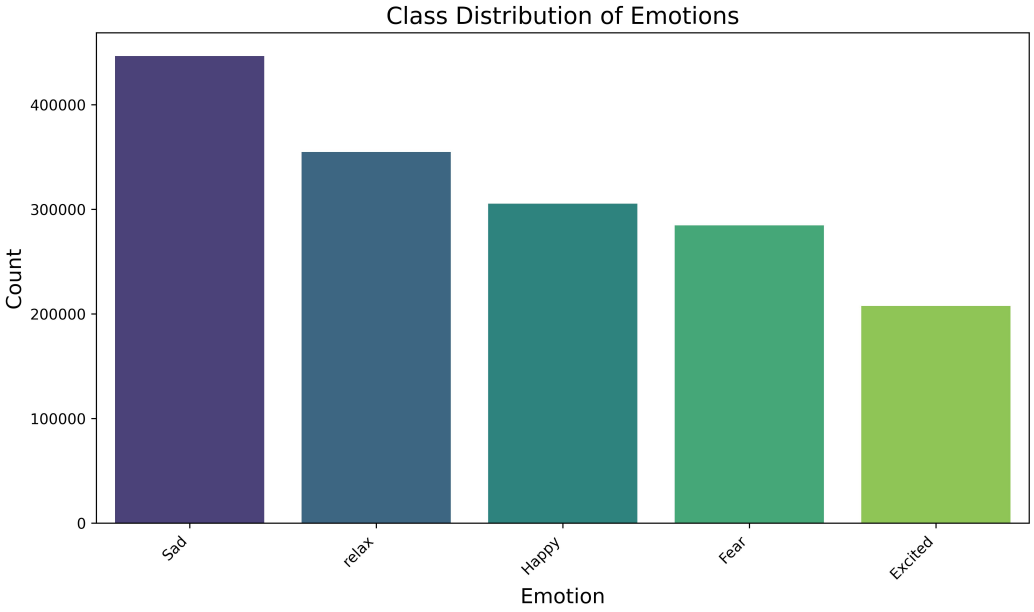


Figure 3.6: Distribution of the five core emotions across the stacked EEG dataset

3.2.3 Artifact Removal

Electroencephalography (EEG) signals are highly susceptible to artifacts from various sources, such as muscle movements and environmental electrical noise [QCY+24]. These artifacts can obscure the true brainwave patterns, making preprocessing through filtering a critical step for accurate signal analysis. Therefore, a bandpass filter is applied to the raw EEG data, restricting the signal to the 0.5-45 Hz range, where brain activity related to emotional responses typically occurs. This frequency band is selected because it encapsulates key brainwave patterns such as delta, theta, alpha, beta, and low gamma waves, which are associated with cognitive and emotional states [QCY+24].

The application of a bandpass filter helps to attenuate low-frequency drifts caused by movements and high-frequency noise from external sources, thus ensuring that the

relevant signal features remain intact for analysis. A Butterworth bandpass filter is chosen specifically due to its flat frequency response within the passband and its minimal phase distortion. These characteristics make it ideal for preserving the integrity of EEG signals without introducing significant delays or distortion [ICJ02]. The transfer function of the Butterworth filter can be described as:

$$H(s) = \frac{1}{\sqrt{1 + \left(\frac{s}{\omega_c}\right)^{2n}}},$$

where ω_c is the cutoff frequency and n is the order of the filter. Here, ω_c corresponds to the 0.5-45 Hz range, and the order $n = 4$ is chosen for smooth signal filtering.

To implement the filter, we use the standard bilinear transformation approach to convert the continuous-time filter into its discrete form, suitable for digital signal processing. The digital form of the filter is applied to each EEG channel individually:

$$y[n] = \sum_{i=0}^{N} b_i x[n-i] - \sum_{i=1}^{M} a_i y[n-i],$$

where $x[n]$ is the input signal, $y[n]$ is the filtered signal, and b_i and a_i are the filter coefficients obtained from the Butterworth filter design.

The filtering process enhances the generalizability of the study across participants, reducing inter-subject variability caused by noise and artifacts. With the artifacts removed, the cleaned EEG dataset provides a better representation of brainwave patterns associated with emotional states. This filtered data will serve as the foundation for machine learning-based emotion classification. To demonstrate the effectiveness of the filtering process, Figs. 3.7, 3.8, and 3.9 show the original and filtered signals from the P3, FT9, and FT10 channels, respectively. These figures illustrate the significant reduction in noise and the preservation of essential signal features.

3.3 Feature Extraction

Feature extraction is a crucial step in signal processing that involves transforming raw EEG data into a set of informative features for subsequent analysis, such as classification tasks. The primary goal is to reduce the complexity of the data while preserving the essential information related to brain activity, which in this study is connected to emotional states [QCY+24].

3.3.1 Windowing Technique

To capture temporal dynamics in the EEG signals, a sliding window technique is employed [WW+13]. The EEG data is divided into overlapping segments or windows, each spanning 2 seconds, corresponding to 256 samples at the 128 Hz sampling rate. This method allows to capture non-stationary behavior in the EEG signals, facilitating a more dynamic analysis of brainwave patterns across time.

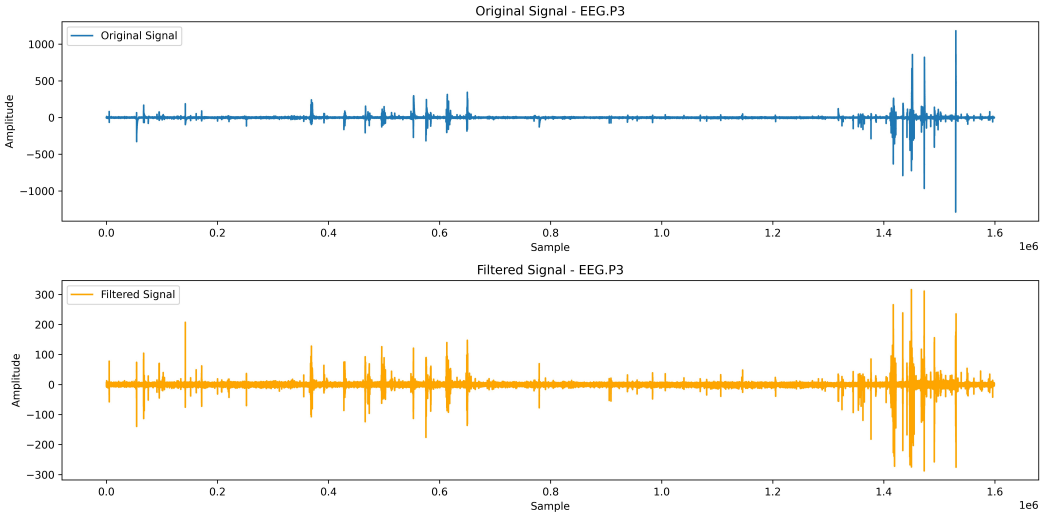


Figure 3.7: Original and filtered EEG signal from the P3 channel. The figure highlights the reduction of noise while preserving the essential characteristics of brainwave patterns associated with emotional states.

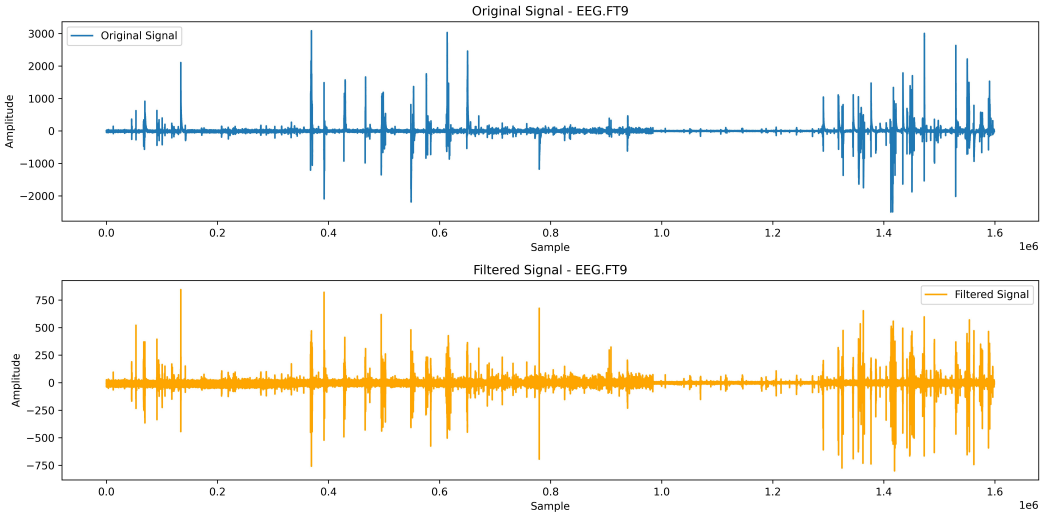


Figure 3.8: Original and filtered EEG signal from the FT9 channel. This visualization emphasizes on the effectiveness of the bandpass filter in enhancing the clarity of the EEG signals.

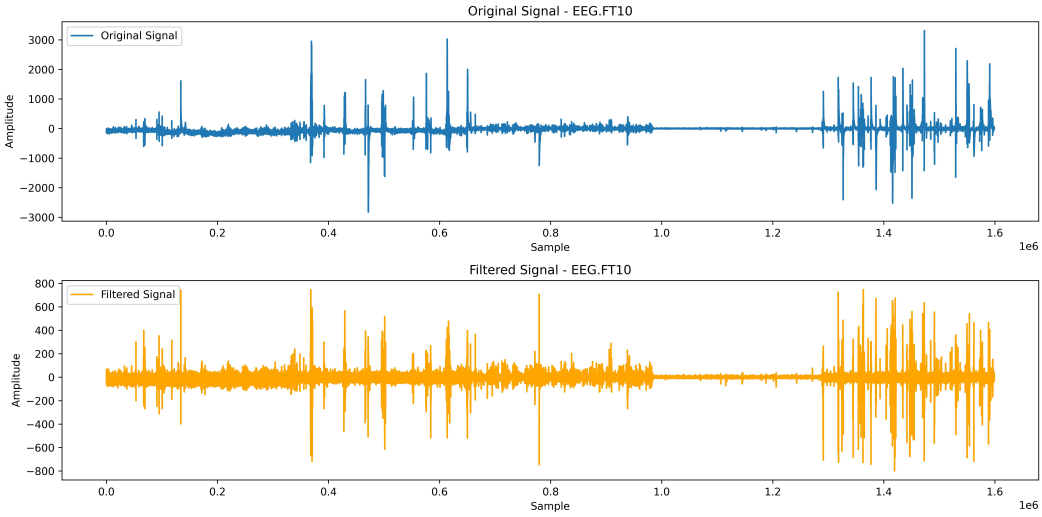


Figure 3.9: Original and filtered EEG signal from the FT10 channel, demonstrating the improvement in signal quality post-filtering.

Mathematically, the segmentation can be represented as:

$$x[n] = \{x_1, x_2, ..., x_{N_w}\},$$

where $x[n]$ represents the entire EEG signal, and x_i denotes a segment of the signal containing N_w samples within a 2-second window. The windowing step size is chosen to balance the trade-off between temporal resolution and computational efficiency, ensuring that sufficient signal variations are captured without overwhelming the model with excessive data.

3.3.2 Statistical Feature Calculation

For each window of EEG data, a diverse set of statistical features is computed to capture both time-domain and frequency-domain characteristics. These features includes measures of central tendency (such as mean and median), variability (e.g., standard deviation, variance), and frequency-specific metrics like power spectral density (PSD) and band power across various frequency bands (delta, theta, alpha, beta, and gamma). This comprehensive feature extraction approach is critical for uncovering the nuanced patterns associated with different emotional states in EEG signals.

In line with the strategy [AC16], this multi-dimensional feature set enhances the model's ability to detect and classify emotional states by enriching the representation of the EEG signals. The inclusion of both temporal and spectral features ensures a robust understanding of how emotions influence neural oscillations across different time scales and frequency bands.

Measures of Central Tendency

Measures of central tendency provide insight into the general location of the signal's amplitude distribution. The following equations are approximations, as the exact distribution of the signal is unknown.

- **Mean** (μ): The average value of the signal over the window, given by:

$$\mu = \frac{1}{N_w} \sum_{i=1}^{N_w} x_i,$$

  where x_i are the signal observations and N_w is the number of samples in the window.

- **Median**: The middle value of the signal, offering robustness against outliers or skewed distributions.

Measures of Variability

Measures of variability capture the spread and distributional properties of the signal, providing deeper insights into signal dynamics. These equations are also approximations due to the unknown underlying distributions.

- **Standard Deviation** (σ): Measures the spread of the signal values around the mean:

$$\sigma = \sqrt{\frac{1}{N_w} \sum_{i=1}^{N_w} (x_i - \mu)^2},$$

  where x_i are the signal values, μ is the mean, and N_w is the number of samples.

- **Variance**: The square of the standard deviation, indicating overall signal variability.

- **Root Mean Square (RMS)**: Provides a measure of the signal's magnitude:

$$\text{RMS} = \sqrt{\frac{1}{N_w} \sum_{i=1}^{N_w} x_i^2},$$

  where x_i are the signal values and N_w is the number of samples.

- **Skewness**: Assesses the asymmetry of the signal's amplitude distribution, indicating potential biases:

$$\text{Skewness} = \frac{\frac{1}{N_w} \sum_{i=1}^{N_w} (x_i - \mu)^3}{\sigma^3},$$

  where x_i are the signal values, μ is the mean, σ is the standard deviation, and N_w is the number of samples.

- **Kurtosis**: Evaluates the "tailedness" of the signal distribution, identifying the presence of outliers:

$$\text{Kurtosis} = \frac{\frac{1}{N_w}\sum_{i=1}^{N_w}(x_i - \mu)^4}{\sigma^4} - 3,$$

  where x_i are the signal values, μ is the mean, σ is the standard deviation, and N_w is the number of samples.

- **Interquartile Range (IQR)**: Measures the range within which the central 50% of data points lie, defined as:

$$\text{IQR} = Q_3 - Q_1,$$

  where Q_1 and Q_3 are the first and third quartiles, respectively.

- **Peak-to-Peak Amplitude**: The difference between the maximum and minimum signal values, reflecting the signal's overall range.

- **Zero-Crossing Rate (ZCR)**: The rate at which the signal crosses zero, providing an estimate of the signal's oscillatory behavior:

$$\text{ZCR} = \frac{1}{N_w}\sum_{i=1}^{N_w-1}\mathbf{1}_{x_i x_{i+1}<0},$$

  where $\mathbf{1}_{x_i x_{i+1}<0}$ is an indicator function that counts zero crossings, and N_w is the number of samples.

3.3.3 Frequency-Domain Analysis

While the previously mentioned features serve well in capturing the estimation of statistical data of the EEG signal, another set of techniques make use of the frequency domain representation like the power spectral density (PSD) and the discrete wavelet transform (DWT), both capturing key data for some specific frequency bands and at specific time windows. These techniques balance out between both the time and the frequency domain, trading off some clarity in one to achieve more locality in the other.

Power Spectral Density (PSD)

In addition to time-domain features, frequency-domain analysis is performed by computing the PSD of the signal. The Welch method is used to estimate the PSD, which smooths the signal by dividing it into overlapping segments and averaging their periods [LTX+16]. This technique reduces noise and improves the reliability of the PSD estimate. The PSD provides insight into the power distribution across different frequency bands:

$$\text{PSD} = \frac{1}{T} \left| \mathcal{F}\{x(t)\} \right|^2 ,$$

where T is the time duration of the signal and $\mathcal{F}$ denotes the Fourier transform.

For each window, the PSD is computed over the following EEG frequency bands [QCY+24]:

- **Delta (0.5-4 Hz)**
- **Theta (4-8 Hz)**
- **Alpha (8-12 Hz)**
- **Beta (12-30 Hz)**
- **Gamma (30-45 Hz)**

The band power for each frequency band is calculated by integrating the PSD within the corresponding frequency range:

$$P_{\text{band}} = \int_{f_{\text{low}}}^{f_{\text{high}}} \text{PSD}(f)\, df,$$

where P_{band} is the power in the specified frequency band, $\text{PSD}(f)$ is the Power Spectral Density at frequency f and f_{low} and f_{high} represent the lower and upper bounds of the frequency band, respectively (e.g., for the Delta band, $f_{\text{low}} = 0.5$ Hz and $f_{\text{high}} = 4$ Hz).

This analysis enables the extraction of features that capture the power of specific brainwave activities, which are often associated with various cognitive and emotional states.

Discrete Wavelet Transform (DWT)

The DWT is a powerful tool for analyzing signals in both time and frequency domains, especially suited for non-stationary signals like EEG [ZW20]. Unlike the continuous wavelet transform, which computes wavelet coefficients for all possible scales and translations, the DWT computes coefficients only at discrete intervals, making it computationally efficient [Sub19]. The DWT of a discrete signal $x[n]$, where n is the time index, is defined as:

$$W(j,k) = \sum_n x[n]\, \psi_{j,k}[n],$$

where $W(j,k)$ represents the wavelet coefficients at scale j and position k, $\psi_{j,k}[n]$ is the discrete wavelet function derived from the mother wavelet $\psi[n]$, and it is given by:

$$\psi_{j,k}[n] = \frac{1}{\sqrt{2^j}} \psi \left(\frac{n - 2^j k}{2^j} \right),$$

j is the scale parameter, controlling the frequency resolution, and k is the translation (position) parameter, controlling the time localization of the wavelet.

In this discrete form, j and k are integers representing the discrete scaling and shifting of the wavelet, which enables the transformation of the signal into different frequency bands.The DWT decomposes the signal into approximation and detail coefficients at various levels of resolution:

- Approximation coefficients capture the low-frequency components of the signal, corresponding to the global, smooth features.
- Detail coefficients capture the high-frequency components of the signal, corresponding to the fine details and rapid changes.

Each level of decomposition splits the signal into these two sets of coefficients, with the approximation coefficients further decomposed in the next level. This iterative process allows us to analyze the signal at multiple resolutions. In our study, we use the DWT to decompose the EEG signals into different frequency bands, corresponding to well-known EEG rhythms mentioned earlier. For the DWT, we use the Daubechies wavelet (db4) due to its similarity with EEG waveforms, capturing transient characteristics effectively [Sub07]. The decomposition is carried out over 5 levels, ensuring that all the relevant EEG frequency bands mentioned earlier are captured.

Feature Extraction Using DWT

After applying the DWT, we extract the following features from each EEG channel:

- **Energy of wavelet coefficients**: The energy of the approximation and detail coefficients at each level provides insights into the power distribution across different frequency bands. It is calculated as:

$$E_j = \sum_{k}^{K} |W(j,k)|^2,$$

  where K is the number of wavelet coefficients and $W(j,k)$ are the wavelet coefficients at level j.

- **Entropy of wavelet coefficients**: Entropy measures the complexity or randomness of the signal in each frequency band. It is calculated as:

$$H_j = -\sum_{k}^{K} p_{j,k} \log(p_{j,k}),$$

  where K is the number of wavelet coefficients and $p_{j,k}$ is the probability distribution of the wavelet coefficients at level j.

- **Standard deviation of wavelet coefficients**: The standard deviation of the wavelet coefficients reflects the variability of the signal in each frequency band, capturing the spread of the signal's energy.

$$\sigma_j = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(W(j,k) - \mu_j)^2},$$

where: $W(j,k)$ is the wavelet coefficient, μ_j is the mean of the wavelet coefficients, K is the total number of wavelet coefficients, and σ_j is the standard deviation of the wavelet coefficients, all at level j.

The sliding window technique effectively captures the temporal evolution of EEG signals, making it possible to observe short-term fluctuations that may be associated with emotional changes. By dividing the signal into smaller, manageable windows, the complexity of the data is reduced, allowing for a more focused analysis of signal dynamics.

The time-domain statistical features provide essential insights into the overall characteristics of the signal, while the frequency-domain features enable a more detailed analysis of brainwave activities across different frequency bands. The Welch method for estimating PSD is advantageous in reducing noise, thereby providing a more accurate and reliable representation of the power distribution across the EEG spectrum [QCY+24]. As for the DWT, it offers multi-resolution analysis by decomposing the signal into different frequency bands, thus DWT captures both fine and coarse features of the EEG signal. This way the DWT provides information about both when and at what frequency certain features of the signal occur, which is critical for capturing transient emotional states.

Together, these time-domain and frequency-domain features form a comprehensive feature set, offering a robust basis for the subsequent classification of emotional states using machine learning techniques.

3.3.4 Differential Entropy

Feature extraction is crucial because it helps the classifier to analyze the data more, which improves classification performance. Here, we make use of the differential entropy method. First, we must determine the signal's variance to compute differential entropy. We use a windowing approach to partition the EEG signals before computing differential entropy. The windowing technique approach is mentioned section 3.3.1, the windowing technique approach allows for computing the differential entropy across the segmented singles. The computed differential entropy is fed as input to the classifier to preform the classification. Differential Entropy feature extracted from Segmented EEG data provides stable and accurate information for emotion classification[ZZPL14].

DE extends the idea of Shannon entropy and is used to measure the complexity of a continuous random variable. DE as a feature was first introduced to EEG-based emotion recognition by Duan et al [DZL13].

$$h(Y) = -\int_{-\infty}^{\infty} \sqrt{\frac{1}{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}} \log\left(\sqrt{\frac{1}{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}\right) dy$$

$$= \frac{1}{2}\log(2\pi e\sigma^2)$$

Here, the DE is employed on a Windowed segmented signal across the EEG signal to reduce the dimensionality of the signal.DE feature extracted from EEG data provides stable and accurate information for emotion classification[ZL15].

3.4 Dimensionality Reduction

EEG signals, due to their high temporal resolution, often generate a large number of features when analyzed. While rich in information, this high-dimensional feature space can lead to challenges such as increased computational complexity and overfitting when used in machine learning models. To address these issues, dimensionality reduction techniques are employed, aiming to retain essential information while reducing the number of variables. This section discusses the use of Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) as key techniques for dimensionality reduction.

3.4.1 Principal Component Analysis

PCA is a statistical technique use to transform a set of correlated variables into a set of uncorrelated variables, known as principal components. These components are ordered by the amount of variance they capture from the original data, with the first principal component explaining the largest amount of variance, followed by each subsequent component. PCA achieves dimensionality reduction by projecting the data onto a lower-dimensional space, preserving as much of the data's variance as possible [QCY+24].

PCA is applied to the feature-extracted EEG dataset, which initially consists of 551 features, corresponding to various statistical and spectral measures derived from windowed EEG segments (section 3.3). The objective is to reduce the dimensionality of the feature space while retaining at least 90% of the variance present in the data.

Standardization of Features

Before applying PCA, it is crucial to standardize the features [CCC+03] as PCA is sensitive to the scaling of data. Features with larger ranges may dominate the calculation of principal components. Standardization transforms the features to have a mean of 0 and a standard deviation of 1, ensuring that each feature contributes equally to the PCA process.

PCA Implementation

PCA decomposes the covariance matrix of the data and identifies principal components as directions of maximum variance [DP20]. Mathematically, PCA solves the eigenvalue problem for the covariance matrix C of the standardized data:

$$C = \frac{1}{n-1} Z^T Z,$$

where n is the number of samples. The eigenvectors of the covariance matrix represent the directions of the principal components, while the eigenvalues correspond to the amount of variance explained by each component.

The cumulative variance explained by the principal components is then calculated to determine how many components are necessary to retain at least 90% of the total variance. The results obtained after running the algorithm of PCA in Python shows that 79 components are required to meet this threshold, representing a substantial reduction in dimensionality from 551 to 79 features as shown in the Fig. 3.10

Explained Variance and Component Selection

The cumulative variance explained by the principal components is plotted in Fig. 3.10. As seen in the figure, the first few components capture a significant proportion of the total variance, with 79 components capturing 90% of the variance. This threshold is indicated by a horizontal red line in the plot, and the corresponding number of components is marked by a vertical dashed red line.

The reduced feature set consisting of 79 principal components is expected to retain most of the useful information from the original dataset, significantly lowering the dimensionality and simplifying the subsequent machine learning tasks without sacrificing much predictive power.

Based on the above results, PCA thus serves as an effective tool for dimensionality reduction in EEG-based emotion recognition, helping to balance the trade-off between retaining useful information and reducing complexity for better model performance.

3.4.2 Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a supervised dimensionality reduction technique used to find a linear combination of features that best separates the classes in a dataset [ZZY+24]. Unlike PCA, which maximizes variance without considering class labels, LDA focuses on maximizing class separation, making it particularly useful for classification tasks with labeled data. In our project, LDA is applied to the scaled feature-extracted EEG recordings, reducing dimensions while maintaining class separability for emotion recognition.

LDA optimizes the ratio of the determinants of the between-class scatter matrix (S_b) to the within-class scatter matrix (S_w) in the feature space [PP08]. The between-class scatter matrix is defined as:

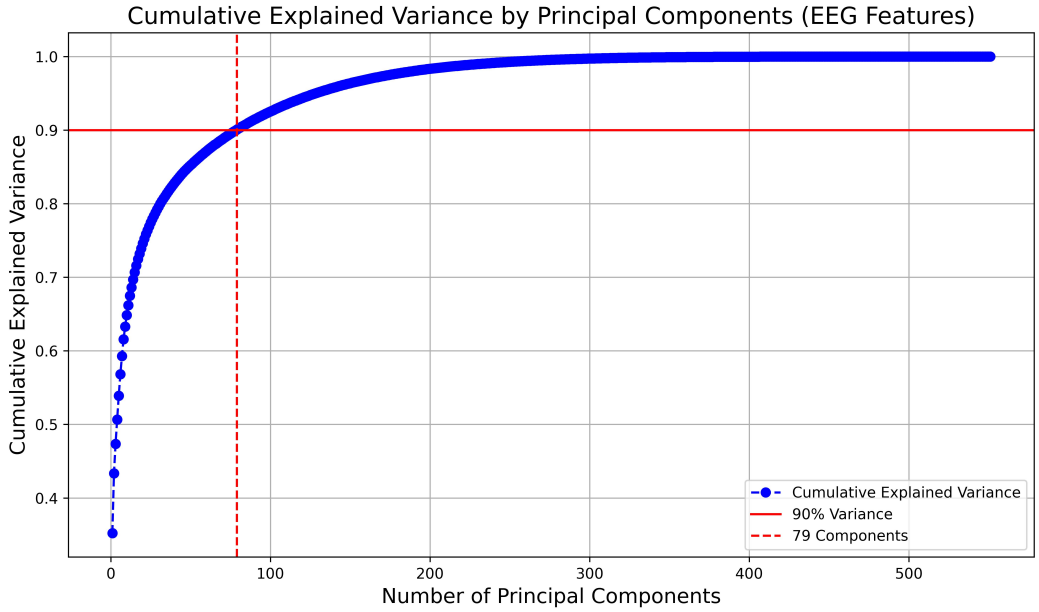


Figure 3.10: Cumulative explained variance by principal components. The red line indicates the 90% variance threshold, and the dashed vertical line shows the number of components required to reach this threshold.

$$S_b = \sum_{i=1}^{k} n_i(\mu_i - \mu)(\mu_i - \mu)^T,$$

where k is the number of classes, n_i is the number of samples in class i, μ_i is the mean vector for class i, and μ is the overall mean vector of the dataset. Conversely, the within-class scatter matrix is defined as:

$$S_w = \sum_{i=1}^{k} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T,$$

where x is a sample in class i and μ_i is the mean vector of class i.

Application of LDA

LDA is applied to the feature-extracted EEG readings, which comprise five classes of emotion: fear, happy, sad, excited, and relaxed. By projecting the high-dimensional feature space onto a lower-dimensional subspace, LDA effectively reduces the dataset's dimensions while preserving class separability.

The number of linear discriminants obtainable is constrained by the number of classes present, allowing for a maximum of $C-1$ linear discriminants [PP08], where C represents the total number of classes. In our case, with five classes, this results in $5-1=4$ linear

discriminants.

LDA Results and Visualization

To illustrate the separation of classes in the reduced feature space, we present the data both before and after applying LDA. Fig. 3.11 shows the original feature space, plotting the first two features of the EEG-recorded dataset. The five emotional states are represented by different colors. The considerable overlap observed among the classes highlights the limited separability of the original features, reinforcing the need for dimensionality reduction techniques like LDA to improve classification accuracy.

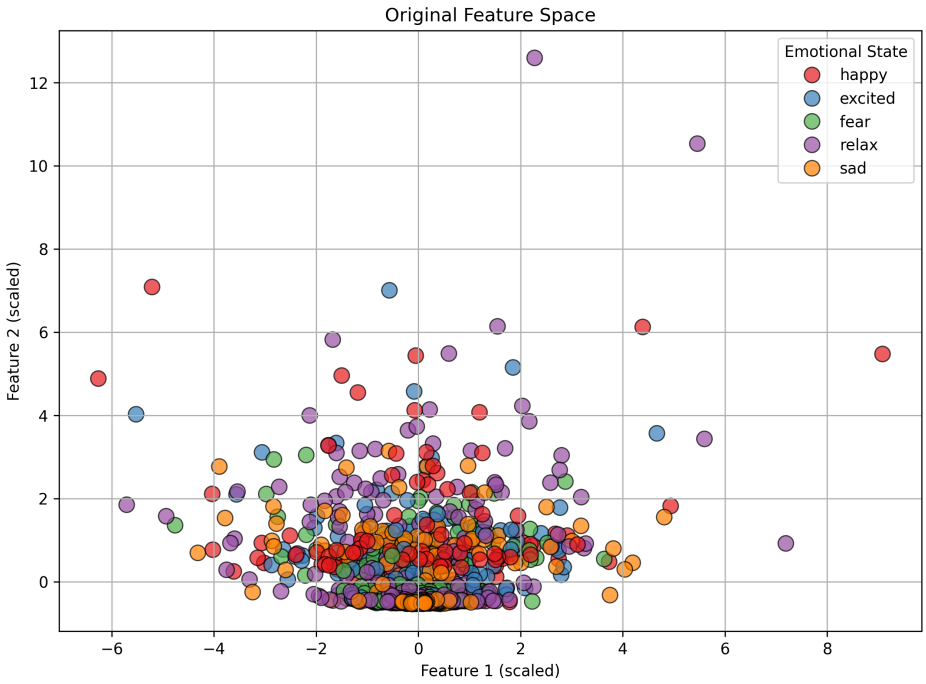


Figure 3.11: Original feature space using Feature 1 and Feature 2 (scaled), highlighting the five emotional states.

Conversely, in Fig. 3.12, the feature space is displayed after applying LDA. The x and y axes represent the first and second linear discriminants (LD1 and LD2), respectively. The plot reveals clear clusters corresponding to the five emotional states: fear, happy, sad, excited, and relaxed. This distinct separation of classes demonstrates the effectiveness of LDA in improving class differentiation, thus supporting enhanced classification accuracy in subsequent machine learning tasks.

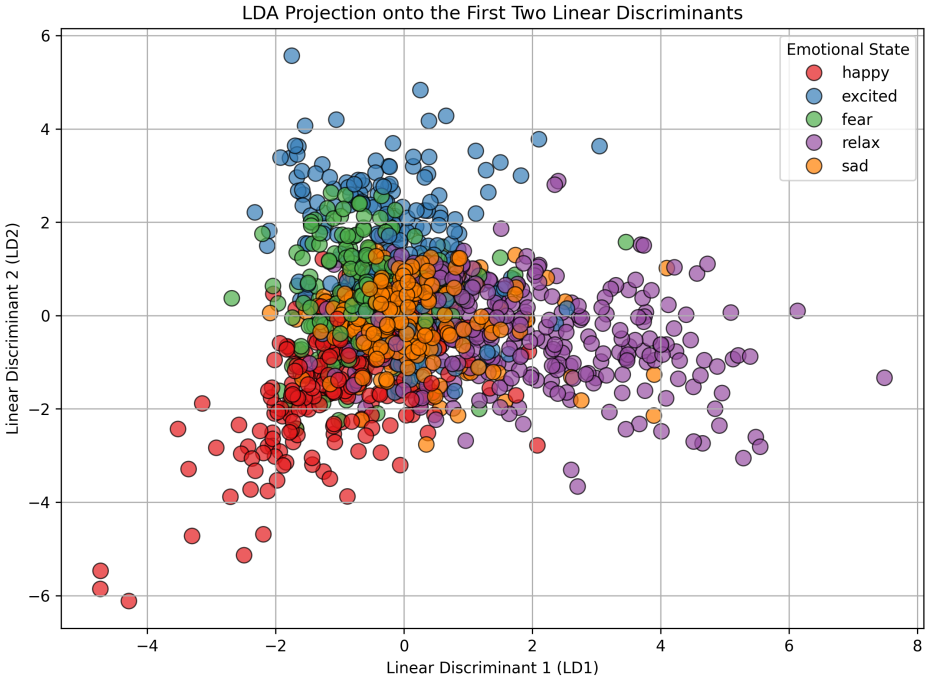


Figure 3.12: LDA-transformed feature space projected onto the first two linear discriminants (LD1 and LD2), showing distinct clusters for the five emotional states.

3.5 Classification

In this section, several classifiers were employed to perform a five-class classification of emotions based on EEG signals. The classification task aimed to identify the emotional state of the participants, categorized into five classes: Excited, Fear, Happy, Sad, and Relax. To assess the effectiveness of these models, a range of performance metrics such as accuracy, precision, recall, F1-score, and confusion matrices were used. The results were evaluated to determine the most effective classification approach.

3.5.1 Support Vector Machines (SVMs)

SVMs are chosen as one of the primary classifiers due to their ability to handle high-dimensional data effectively, which is particularly useful for EEG datasets with a large number of features. Specifically, an SVM with a Radial Basis Function (RBF) kernel is implemented. The RBF kernel is widely used for non-linear classification tasks, as it maps the input features into a higher-dimensional space where the data becomes linearly separable [AMMK16].

The primary goal of SVMs is to find the hyperplane that best separates the data

points into distinct classes. The optimal hyperplane is the one that maximizes the margin between the two classes. This margin is the distance between the hyperplane and the closest data points from each class, called support vectors [CST00].

The objective function in SVM aims to minimize the following cost function:

$$\text{minimize } \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i,$$

subject to the constraints:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \forall i,$$

where $\mathbf{w}$ represents the weights of the hyperplane, C is the regularization parameter controlling the trade-off between maximizing the margin and classification error, and ξ_i are slack variables that allow misclassification for non-linearly separable data. y_i are the actual class labels, and $\mathbf{x}_i$ are the feature vectors. By solving this optimization problem, the SVM finds the hyperplane that minimizes classification errors while maximizing the margin.

Data Augmentation and Class Balancing

The challenge we face in EEG-based emotion classification is the imbalance in class distribution, where certain emotions may be underrepresented in the dataset. To address this, data augmentation is performed to create a balanced dataset. The Synthetic Minority Over-sampling Technique (SMOTE) was employed, which generates synthetic samples for the minority classes by interpolating between existing samples. Additionally, the noise was added to the augmented data to improve the model's generalization capability [CBHK02].

This technique helps in balancing the dataset and ensuring that the classifier is not biased toward any particular class.

Training and Hyper parameters

The SVM model is trained using the augmented and standardized data set. The regularization parameter $C = 10$ is chosen by performing a grid search technique in Python to balance the trade-off between maximizing the margin and minimizing classification error. Additionally, class weights are computed and applied to penalize miss-classifications of minority classes more heavily, ensuring that the model focuses on improving performance across all classes.

The decision function of the SVM can be written as:

$$f(\mathbf{x}) = \sum_{i=1}^{n}\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b,$$

where α_i are the Lagrange multipliers, y_i are the class labels, $K(\mathbf{x}_i, \mathbf{x})$ is the RBF kernel function, defined as $K(\mathbf{x}_i, \mathbf{x}) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}\|^2)$ and b is the bias term.

The SVM's objective is to maximize the margin between the different emotion classes while minimizing the classification error.

Performance Evaluation

After training in Python, the SVM model is evaluated using the test dataset. The training accuracy reaches 91.02%, while the test accuracy is 79.47%. Although there is a slight drop in accuracy between the training and test sets, the model generalized reasonably well to unseen data.

To further analyze the model's performance, a detailed classification report is generated, providing key metrics such as precision, recall, F1-score, and support for each class [SL09]. These metrics are computed as follows:

- **Precision** is the ratio of correctly predicted positive observations to the total predicted positives:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}.$$

- **Recall** is the ratio of correctly predicted positive observations to all observations in the actual class:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}.$$

- **F1-Score** is the harmonic mean of precision and recall, giving a balanced measure of a classifier's performance:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

The classification report, presented in Table 3.1, shows balanced performance across the five emotion classes.

Emotion	**Precision**	**Recall**	**F1-Score**	**Support**
Excited	0.75	0.90	0.82	600
Fear	0.81	0.80	0.81	600
Happy	0.77	0.76	0.77	600
Sad	0.82	0.76	0.79	600
Relax	0.84	0.75	0.79	600
Test Accuracy	79.5%			

Table 3.1: Classification Report for SVM

Confusion Matrix Analysis

The confusion matrix [KP98], shown in Table 3.2, provides further insight into the model's performance. It displays the actual versus predicted class labels and highlights the areas where the classifier performed well and where it struggled. For instance, the excited emotion class shows a strong performance with 539 true positives, but there is some confusion with other emotions such as fear and happy.

Table 3.3 presents the normalized version of the confusion matrix. This matrix allows for a clearer comparison of performance across emotion classes by showing the relative percentage of correct and incorrect classifications.

	Excited	**Fear**	**Happy**	**Sad**	**Relax**
Excited	539	22	16	9	14
Fear	59	480	26	16	19
Happy	48	32	458	31	31
Sad	33	38	50	455	24
Relax	42	17	45	44	452

Table 3.2: Confusion Matrix for SVM

	Excited	**Fear**	**Happy**	**Sad**	**Relax**
Excited	0.83	0.03	0.02	0.01	0.02
Fear	0.09	0.74	0.04	0.02	0.03
Happy	0.07	0.05	0.71	0.05	0.05
Sad	0.05	0.06	0.08	0.71	0.04
Relax	0.06	0.03	0.07	0.07	0.70

Table 3.3: Normalized Confusion Matrix for SVM

Summary of Results

The SVM model with an RBF kernel achieved promising results, with a training accuracy of 91.02% and a test accuracy of 79.47%. The classification report and confusion matrix indicate balanced precision, recall, and F1-scores across most emotion classes, demonstrating that the SVM can effectively classify emotions from EEG data.

3.5.2 Deep Neural Networks (DNNs)

DNNs are also employed for emotion classification based on EEG data [LYH+22]. The dataset used for training is augmented to address class imbalances, ensuring a more robust learning process. The model's effectiveness is evaluated using accuracy metrics, a comprehensive classification report, and a confusion matrix [KP98].

DNNs are a class of artificial neural networks that consist of multiple layers of interconnected neurons, designed to capture complex non-linear patterns in data [GBC16]. DNNs typically include an input layer, several hidden layers, and an output layer.

DNNs learn the relationships in the data by adjusting the weights in each neuron through backpropagation. The loss function, typically Cross-Entropy for classification problems, is minimized using gradient descent. The Cross-Entropy loss function is defined as:

$$\text{Loss} = -\sum_{i=1}^{n} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right],$$

where y_i represents the true class labels, and $\hat{y}_i$ represents the predicted probabilities for the class. The model updates the weights through multiple iterations to minimize the loss and improve accuracy, capturing hierarchical and complex patterns in the data for high-performance classification.

To enhance the model's generalization capability and mitigate overfitting, the following strategies are implemented [IS15], [SHK+14] :

- **Batch Normalization**: This technique normalizes the output of the previous activation layer for each mini-batch. Mathematically, for a given layer output x, the normalized output $\hat{x}$ is computed as:

  $$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}},$$

  where μ is the batch mean, σ^2 is the batch variance, and ϵ is a small constant to prevent division by zero. This normalization helps stabilize the learning process and accelerates convergence.

- **Dropout** : A dropout rate of 30% is applied after each hidden layer. Dropout randomly sets a fraction of input units to zero at each update during training time, which prevents overfitting by reducing the model's reliance on specific neurons.

- **Activation Functions**: The Rectified Linear Unit (ReLU) activation function is used in hidden layers for its ability to introduce non-linearity while maintaining computational efficiency. The ReLU function is defined as:

  $$f(x) = \max(0, x),$$

  The final output layer employs the softmax activation function to convert logits into class probabilities, defined as:

  $$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}},$$

  where z_i are the logits for class i and K is the total number of classes.

- **Class Balancing**: To address class imbalances within the training dataset, class weights are manually adjusted. This adjustment ensures that the model treats all classes with equal importance during training. The class weight for class c can be computed as:

$$\text{class\_weight}[c] = \frac{N}{n_c},$$

where N is the total number of samples and n_c is the number of samples in class c.

Performance Evaluation

After training in Python, the hyperparameters are tuned by the validation data set, and the results are evaluated using the test data set. The training accuracy reaches 91.01%, while the validation accuracy is 82.46% and the test accuracy is 79.70%. The classification report provides detailed insights into the model's performance across different emotion classes. It includes metrics such as precision, recall, F1-score, and support for each class [SL09]. These metrics are already defined under performance evaluation (section 3.5.1). The classification report, presented in Table 3.4, shows balanced performance across the five emotion classes.

Emotion	Precision	Recall	F1-Score	Support
Excited	0.83	0.81	0.82	600
Fear	0.75	0.87	0.80	600
Happy	0.84	0.77	0.80	600
Sad	0.79	0.76	0.78	600
Relax	0.79	0.78	0.78	600
Validation Accuracy	82.5%			
Test Accuracy	80%			

Table 3.4: Classification Report for DNN

Confusion Matrix Analysis

The confusion matrix Table 3.5 provides a visual representation of the classification results, illustrating the true positive, false positive, false negative, and true negative rates for each class. Table 3.6 presents a normalized version of the confusion matrix. This matrix allows for a clearer comparison of performance across emotion classes by showing the relative percentage of correct and incorrect classifications.

Summary of Results

The DNN model achieves a training accuracy of 97.01%, a validation accuracy of 82.46%, and a test accuracy of 79.70%. The classification report indicates balanced precision

	Excited	Fear	Happy	Sad	Relax
Excited	487	46	12	23	32
Fear	22	521	14	17	26
Happy	22	49	462	38	29
Sad	24	48	36	456	36
Relax	31	33	29	42	465

Table 3.5: Confusion Matrix for DNN

	Excited	**Fear**	**Happy**	**Sad**	**Relax**
Excited	0.81	0.08	0.02	0.04	0.05
Fear	0.03	0.87	0.02	0.03	0.05
Happy	0.03	0.10	0.77	0.06	0.05
Sad	0.04	0.09	0.07	0.75	0.05
Relax	0.06	0.06	0.05	0.07	0.76

Table 3.6: Normalized Confusion Matrix for DNN

and recall across the emotion classes, suggesting that the model can effectively classify emotions from EEG data.

In conclusion, the DNN classifier demonstrates a promising approach for emotion recognition using EEG data, with potential avenues for further optimization and enhancement.

3.5.3 k-Nearest Neighbours (kNN)

The k-Nearest Neighbours (kNN) algorithm, known for its simplicity and effectiveness, is widely used in various classification tasks [SFJI20, Mur11]. It is a non-parametric method that operates on the principle that data points from the same class tend to exist in close proximity within the feature space. It classifies a test sample based on the majority class among its 'k' nearest neighbours (where 'k' represents the number of neighbours), as determined by a distance metric [GWB+03].

The distance metric chosen for this implementation is the Euclidean distance, computed by:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}\,.$$

The Euclidean distance $d(\mathbf{x}, \mathbf{y})$ is defined as the distance between the test sample $\mathbf{x}$ and a training sample $\mathbf{y}$, where x_i and y_i represent the coordinates of the points $\mathbf{x}$ and $\mathbf{y}$ in the i-th dimension, and n denotes the number of dimensions in the feature space, corresponding to the four linear discriminants obtained after LDA transformation (i.e., $n = 4$).

Model Training and Hyperparameter Tuning

To train and optimize the k-Nearest Neighbors (kNN) classifier, two cross-validation methods are employed, specifically k-fold and Leave-One-Out Cross-Validation, which are widely recognized for their effectiveness in evaluating classification algorithms [Won15].

i. Stratified K-Fold Cross-Validation: We use Stratified K-Fold cross-validation with 10 splits to select the optimal number of neighbors (k) for the kNN classifier. This method ensures that each fold preserves the proportion of emotion classes, providing a reliable estimate of model performance [SDJ22]. After evaluating multiple values of k, the best result is obtained with $k = 29$, yielding a **mean accuracy of 58.80%**.

ii. Leave-One-Out Cross-Validation: To further assess the model's performance, we apply Leave-One-Out (LOO) cross-validation. In this method, the model trains on all participants except one, and the held-out participant is used for testing. This approach offers a robust estimate of the model's generalizability across different participants [Won15]. Using the best k=29 from Stratified K-Fold, the model achieves a **mean accuracy of 55.90%**.

In the LOO evaluation, different values of k are tested, with the best accuracies ranging from 55.6% to 56.1% for k values between 19 and 29, with the highest accuracy achieved at $k = 19$. In the stratified K-Fold CV, $k = 29$ provides the best accuracy of 58.80%. Given its competitive result in LOO and consistency across both validation methods, the hyperparameter k is tuned to 29, making it a reliable choice for the kNN classifier, as it effectively balances performance across different evaluation techniques.

Performance Evaluation

Following the model training, the kNN classifier's performance is evaluated using various metrics, including accuracy, precision, recall, and F1-score [SL09]. These metrics are essential for understanding how well the model classifies the emotions present in the EEG data and are already defined under performance evaluation of section 3.5.1. The overall test accuracy achieved by the kNN model is **62%**.

- **Classification Report:** The classification report provides detailed insights into the performance of the kNN classifier across different emotion classes. The precision, recall, F1-score, and support metrics are summarized in Table 3.7, highlighting the effectiveness of the kNN classifier.

- **Confusion Matrix:** The confusion matrix, presented in Fig. 3.13, provides a visual representation of the classification results. It illustrates the true positive, false positive, false negative, and true negative rates for each emotion class, allowing for a comprehensive evaluation of the classifier's performance.

Emotion Class	Precision	Recall	F1-Score	Support
excited	0.69	0.52	0.59	81
fear	0.51	0.66	0.58	59
happy	0.68	0.78	0.73	73
relax	0.77	0.57	0.66	77
sad	0.52	0.61	0.56	76
Accuracy	0.62			

Table 3.7: Classification report metrics for kNN (k=29)

3.5.4 Convolutional Neural Networks (CNNs)

CNN Model Architecture:

The idea behind CNNs resembles traditional artificial neural networks (ANNs), which consist of neurons that self-optimize through learning. CNNs are powerful performers on large sequential data represented by matrices, such as images broken down to their pixel values.[TM20]. The block diagram CNN model architecture is shown in Fig 3.14.

1. Input Layer

The input shape used in the CNN is (128, 29, 1), where 128 indicates the time steps and 29 represents the number of EEG channels. This configuration allows the CNN layers to capture the spatial information of EEG signals[IDT+23].

2. Model Layers

The CNN model uses three convolutional layers with filter sizes of (64,128 and 256). This setup helps to extract hierarchical spatial features from the input EEG data.[IDT+23].

3. Activation Functions

The ReLU (Rectified Linear Unit) activation function is applied after each layer to introduce non-linearity and enhance the model's learning capability for CNN [IDT+23].

4. Dropout

A dropout rate of 25% is applied after each convolutional layer to prevent overfitting, encouraging the model to learn more robust features. This regularization method is crucial for improving generalization, as shown in the CNN model used by Iyer et al. (2023) [IDT+23].

6. Dense Layer

The dense layers use ReLU activation to introduce non-linearity and improve the model's capacity to learn complex patterns. The CNN architecture leverages fully connected

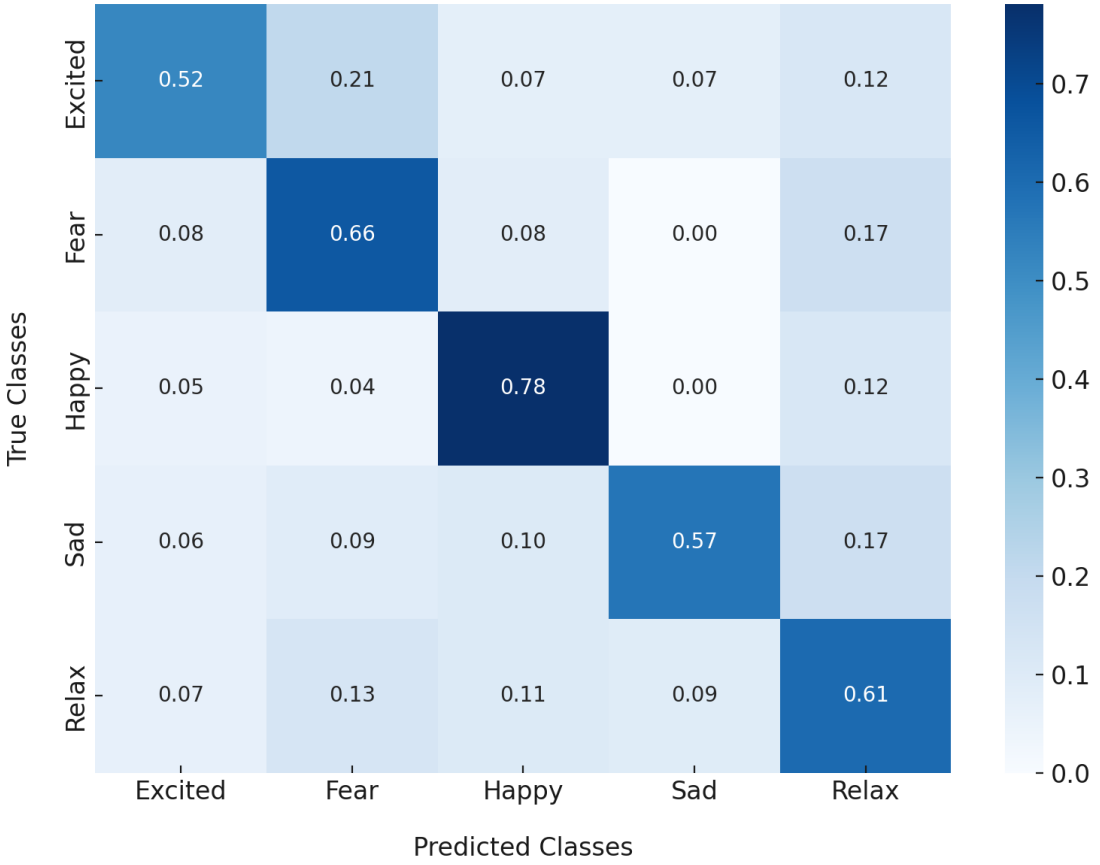


Figure 3.13: Normalized heatmap of the Confusion Matrix for kNN.

layers with 512, 256, and 64 neurons, as described by Iyer et al. (2023) [IDT+23].

7. Output Layer

The softmax activation function in the output layer generates a probability distribution over the emotion classes, making it suitable for multi-class classification tasks. This implementation is also found in the CNN-based model by Iyer et al. (2023) [IDT+23].

8. Batch Size and Epochs

The model is trained with a batch size of 32 and for 50 epochs to ensure a balance between training time and stability, following the training strategy used by Iyer et al. (2023) [IDT+23].

9. Optimizer and Loss Function

The Adam optimizer and categorical cross-entropy loss function are used for their effectiveness in multi-class classification tasks, as detailed by Iyer et al. (2023) [IDT+23].

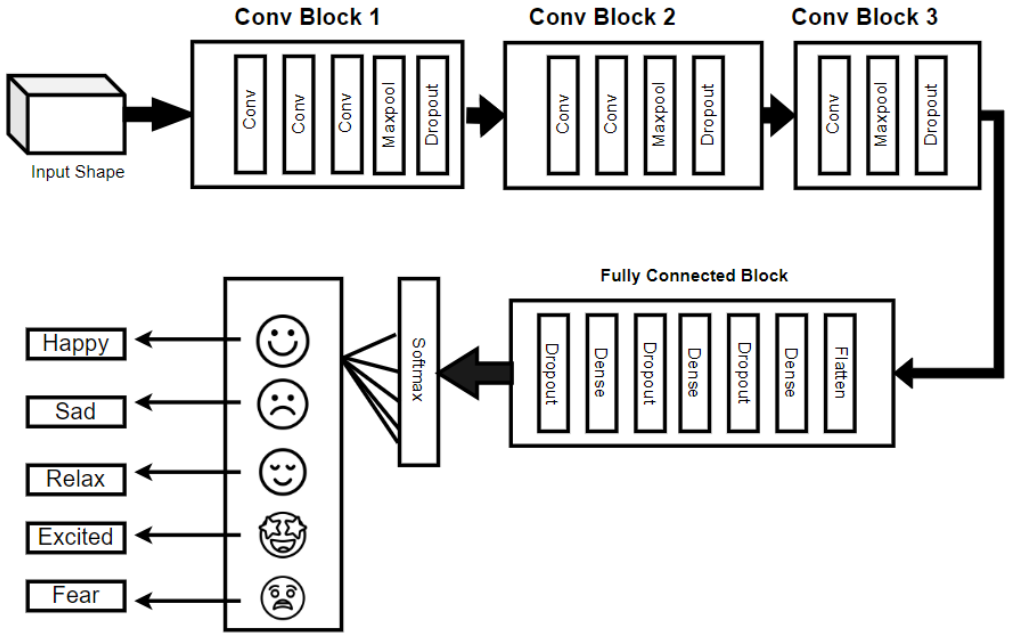


Figure 3.14: CNN-based model architecture.

In this project, the Classifier was estimated using two different approaches. The first approach used feature extraction as differential entropy, where extracted features fit into the classification model, and the other were classified without feature extraction and fit with raw data. In this context, raw data describes unprocessed EEG signals taken straight from the user and then put into the classifiers for analysis.

Classification Report:

The CNN Model Performance for each of the different emotions was represented in the classification. For every class, the classification reports include metrics like support, F1-score, and recall. The details about metrics can be obtained from Equations mentioned in 3.5.1. Here, Class 0: excited, Class 1: fearful, Class 2: happy, Class 3: relaxed, and Class 4: sad. Fig. 3.8 is the classification report based on Feature Extraction and Fig. 3.9 is the classification report based on raw data.

Confusion Matrix

The confusion matrix is a performance evaluation tool. It discusses predicted labels in comparison to actual labels. The confusion matrix is used in multi-class classification.

Class	Precision	Recall	F1-Score	Support
0	0.65	0.69	0.67	62
1	0.61	0.85	0.71	72
2	0.82	0.71	0.76	69
3	0.88	0.86	0.87	84
4	0.64	0.49	0.55	90
Training Accuracy 0.7135		Validation Accuracy 0.6990		

Table 3.8: CNN model classification report based on feature extraction.

Class	Precision	Recall	F1-Score	Support
0	0.75	0.76	0.75	7015
1	0.81	0.78	0.80	9384
2	0.74	0.69	0.72	7554
3	0.80	0.89	0.84	11612
4	0.97	0.92	0.95	12675
Training Accuracy 0.8272		Validation Accuracy 0.7780		

Table 3.9: CNN model classification report based on raw data.

The Confusion matrix provides True positives, true negatives, false positives, and false negatives for each class. The Fig. 3.15 is the confusion matrix with feature extraction and Fig. 3.16 is the confusion matrix with raw data.

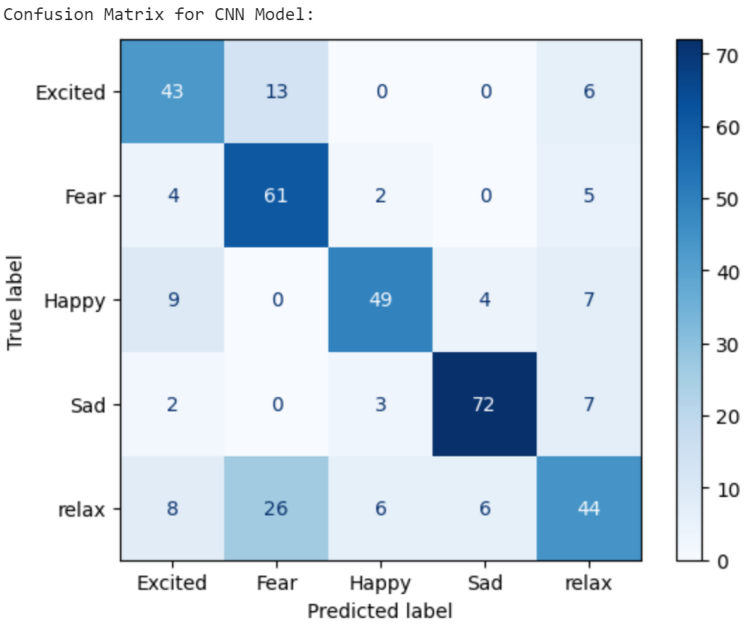


Figure 3.15: Confusion matrix of CNN model based on feature extraction.

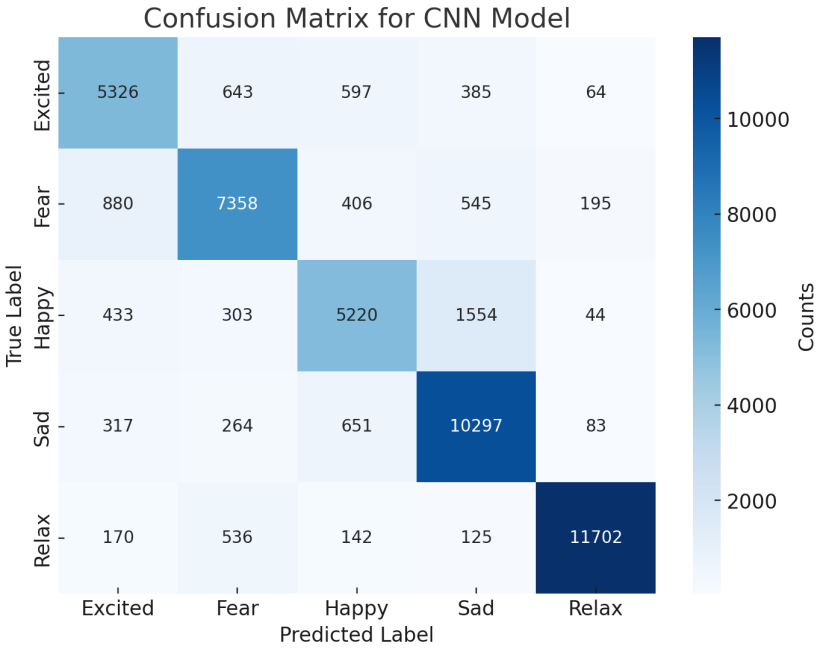


Figure 3.16: Confusion matrix of CNN model based on raw data.

3.5.5 Long Short-Term Memory (LSTM):

LSTM Model Architecture

LSTM networks are an advanced form of recurrent neural networks (RNNs) that can learn long-term dependencies. They are defined by weight matrices from both the input and prior state for every gate [GSK+17]. The standard LSTM formula with a logistic function (σ) is applied to the gates. The block diagram of the LSTM model architecture is shown in Fig. 3.17.

1. **Input Layer:**
   The input shape used in the LSTM model is (128, 29), where 128 indicates the time steps, and 29 represents the number of EEG channels. This configuration allows the LSTM layers to capture the temporal dynamics of EEG signals [IDT+23].

2. **Convolutional Layers:**
   The model employs four layers with memory sizes of 128 and 256 neurons, respectively. This setup captures both short-term and long-term dependencies in EEG data [IDT+23].

3. **Activation Functions:**
   The Tanh activation function is applied after each LSTM layer to introduce nonlinearity and enhance the learning of complex temporal features [IDT+23].

4. **Dropout:**
   A dropout rate of 30% is applied after each LSTM layer to prevent overfitting. This regularization technique encourages the model to learn more robust features and improve generalization, as shown in the LSTM model used by Iyer et al. (2023) [IDT+23].

5. **Fully Connected Dense Layer:**
   After the LSTM layers, three dense layers with 512, 256, and 64 neurons are included, each using ReLU activation. ReLU introduces non-linearity, which enhances the model's capacity to learn complex patterns from the features extracted by the LSTM layers, as described by Iyer et al. (2023) [IDT+23].

6. **Output Layer:**
   The softmax activation function is used in the output layer to produce a probability distribution over the emotion classes, suitable for multi-class classification tasks. This approach is also found in the LSTM-based model by Iyer et al. (2023) [IDT+23].

7. **Batch Size and Epochs:**
   The model is trained with a batch size of 32 for 50 epochs, ensuring a balance between training time and model stability. This training strategy follows that used by Iyer et al. (2023) [IDT+23].

8. **Optimizer and Loss Function:**
   The Adam optimizer and categorical cross-entropy loss function are utilized for their effectiveness in multi-class classification tasks, as highlighted by Iyer et al. (2023) [IDT+23].

Classification Report

The LSTM Model Performance for each of the different emotions is represented in the classification report. For every class, the classification report includes metrics like support, F1-score, and recall. The details about these metrics can be obtained from equations mentioned in 3.5.1. The classes are defined as follows: Class 0: excited, Class 1: fear, Class 2: happy, Class 3: relax, and Class 4: sad.

The classification report based on Feature Extraction is shown in Fig. 3.10, and the report based on raw data is shown in Fig. 3.11.

Confusion Matrix

The confusion matrix is a performance evaluation tool. It discusses predicted labels in comparison to actual labels. The confusion matrix is used in multi-class classification. The Confusion matrix provides True positives, true negatives, false positives, and false negatives for each class. The Fig. 3.18 is the confusion matrix with feature extraction and Fig. 3.19 is the confusion matrix with raw data.

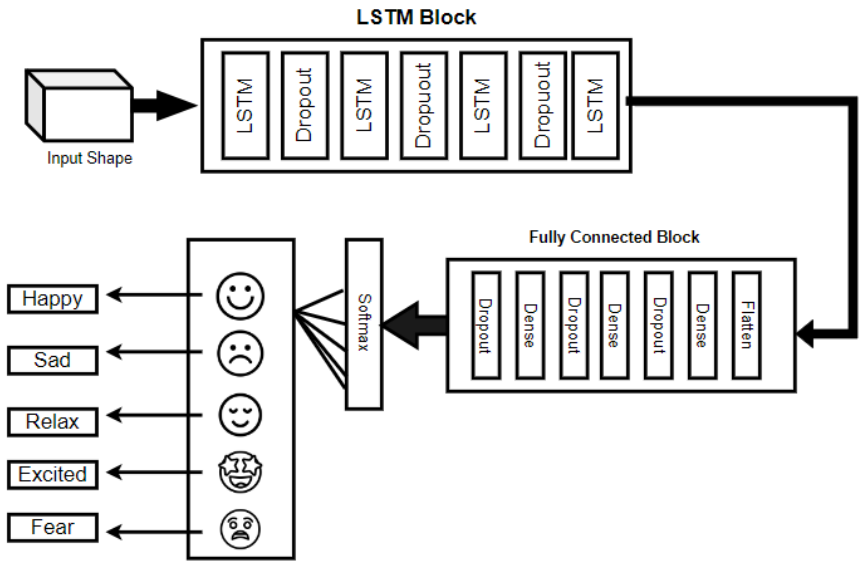
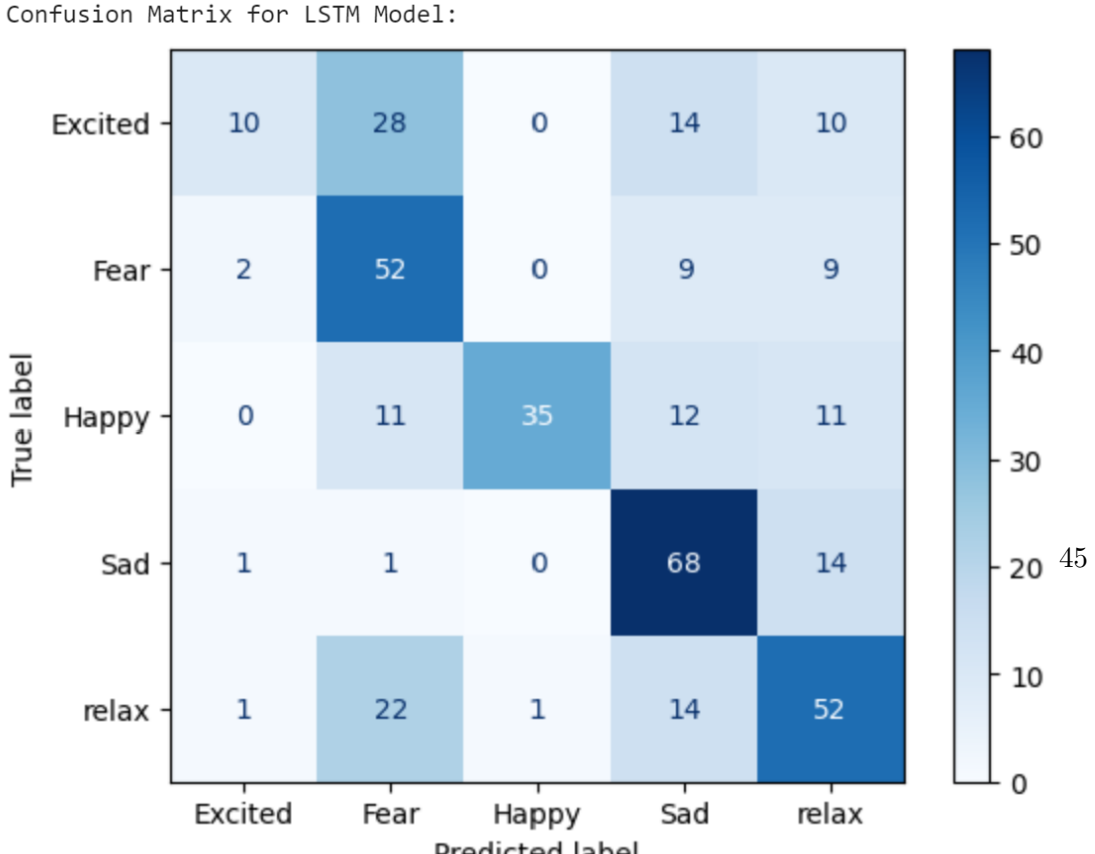


Figure 3.17: LSTM-based model architecture.

Class	Precision	Recall	F1-Score	Support
0	0.71	0.16	0.26	62
1	0.46	0.72	0.56	72
2	0.97	0.51	0.67	69
3	0.58	0.81	0.68	84
4	0.54	0.58	0.56	90
Training Accuracy: 0.6056		Validation Accuracy:0.5705		

Table 3.10: LSTM model classification report based on feature extraction.



Class	Precision	Recall	F1-Score	Support
0	0.77	0.79	0.78	7015
1	0.85	0.81	0.83	9384
2	0.74	0.76	0.75	7554
3	0.82	0.86	0.84	11612
4	0.95	0.92	0.93	12675
Training Accuracy 0.8390		Validation Accuracy:0.8065		

Table 3.11: LSTM model classification report based on raw data.

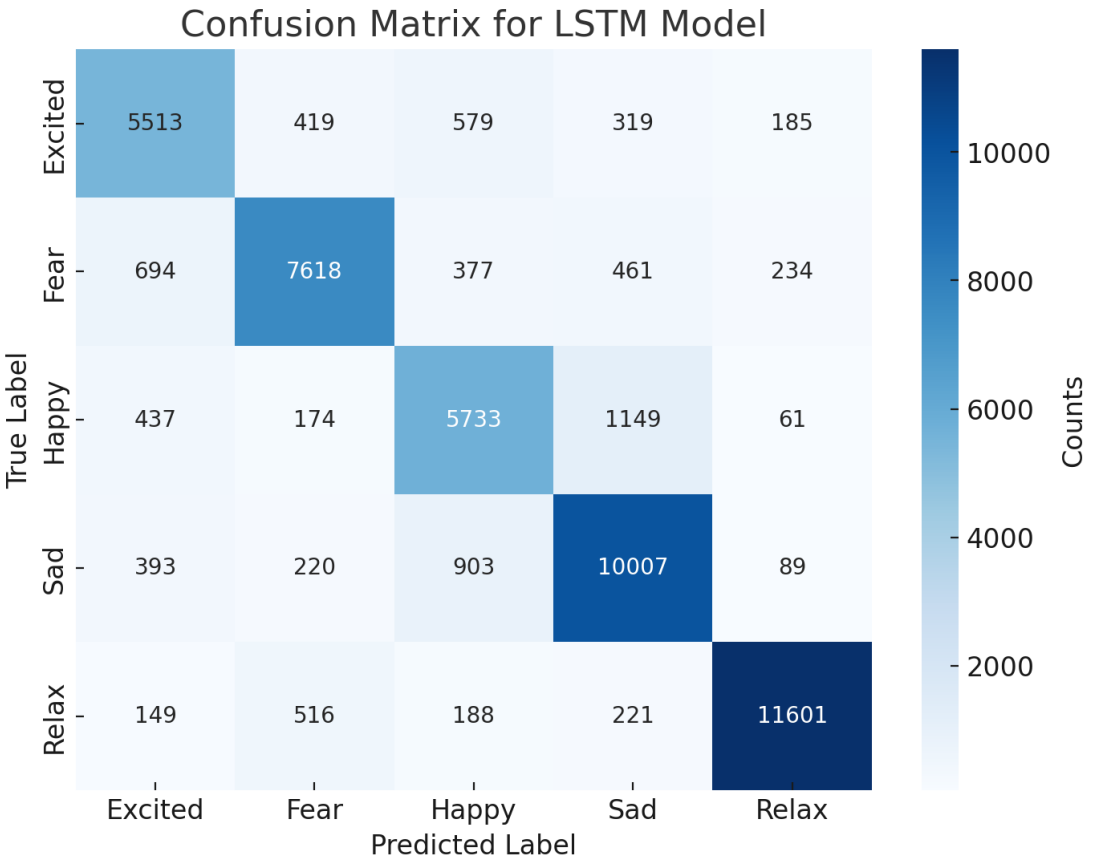


Figure 3.19: Confusion matrix of LSTM model based on raw data.

3.5.6 Hybrid Model

The hybrid model is employed as a combination of CNN+LSTM as proposed by Iyer et al. (2023) [IDT+23]. The block diagram of the LSTM model architecture is shown in Fig. 3.20.

Hybrid Model Architecture

1. **Input Layer:**
   The input shape used in the hybrid model is (128, 29, 1), where 128 indicates the time steps, and 29 represents the number of EEG channels. This configuration allows the LSTM layers to capture the temporal dynamics of EEG signals [IDT+23].

2. **Convolutional Layers:**
   The model employs four convolutional layers with memory sizes of 128 and 256 neurons, respectively. This setup captures both short-term and long-term dependencies in EEG data [IDT+23].

3. **Activation Functions:**
   The Tanh activation function is applied after each LSTM layer to introduce non-linearity and enhance the learning of complex temporal features [IDT+23].

4. **Dropout:**
   A 25% dropout rate is applied after each CNN layer to prevent overfitting, while a 30% dropout rate is applied to the LSTM layer, encouraging the model to learn more robust features. This regularization method is crucial for improving generalization [IDT+23].

5. **Fully Connected Dense Layer:**
   Three fully connected layers follow the LSTM layers, each using ReLU activation. ReLU introduces non-linearity, which enhances the model's capacity to learn complex patterns from the spatial and temporal features extracted by the layers [IDT+23].

6. **Output Layer:**
   The softmax activation function is used in the output layer to produce a probability distribution over the emotion classes, suitable for multi-class classification tasks [IDT+23].

7. **Optimizer and Loss Function:**
   The Adam optimizer and categorical cross-entropy loss function are utilized for their effectiveness in multi-class classification tasks, as highlighted by Iyer et al. (2023) [IDT+23].

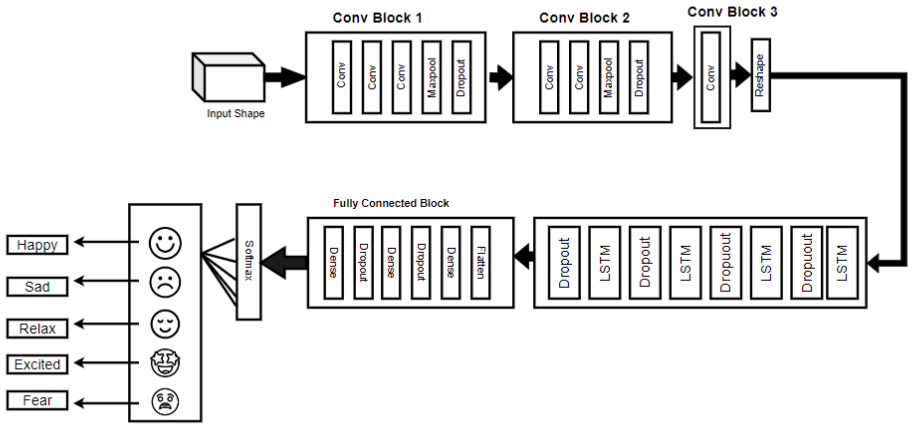


Figure 3.20: Hybrid Model–based model architecture.

Classification Report:

The Hybrid Model(CNN+LSTM) Performance for each of the different emotions was represented in the classification. For every class, the classification reports include metrics like support, F1-score, and recall. The details about metrics can be obtained from equations mentioned in 3.5.1. Here, Class 0: excited, Class 1: fear, Class 2: happy, Class 3: relax, and Class 4: sad. Fig. 3.12 is the classification report based on Feature Extraction and Fig. 3.13 is the classification report based on raw data.

Class	Precision	Recall	F1-Score	Support
0	0.86	0.71	0.78	62
1	0.63	0.93	0.75	72
2	0.83	0.84	0.83	69
3	0.90	0.87	0.88	84
4	0.64	0.49	0.55	90
Training Accuracy: 0.7586		Validation Accuracy:0.7276		

Table 3.12: Hybrid model classification report based on feature extraction.

Confusion Matrix

The confusion matrix is a performance evaluation tool. It discusses predicted labels in comparison to actual labels. The confusion matrix is used in multi-class classification.

Class	Precision	Recall	F1-Score	Support
0	0.72	0.67	0.69	7015
1	0.70	0.75	0.73	9384
2	0.74	0.59	0.66	7554
3	0.76	0.85	0.80	11612
4	0.91	0.89	0.90	12675
Training Accuracy 0.7769		Validation Accuracy:0.7376		

Table 3.13: Hybrid model classification report based on raw data.

The Confusion matrix provides True positives, true negatives, false positives, and false negatives for each class. Fig. 3.21 is the confusion matrix with feature extraction and Fig. 3.22 is the confusion matrix with raw data.

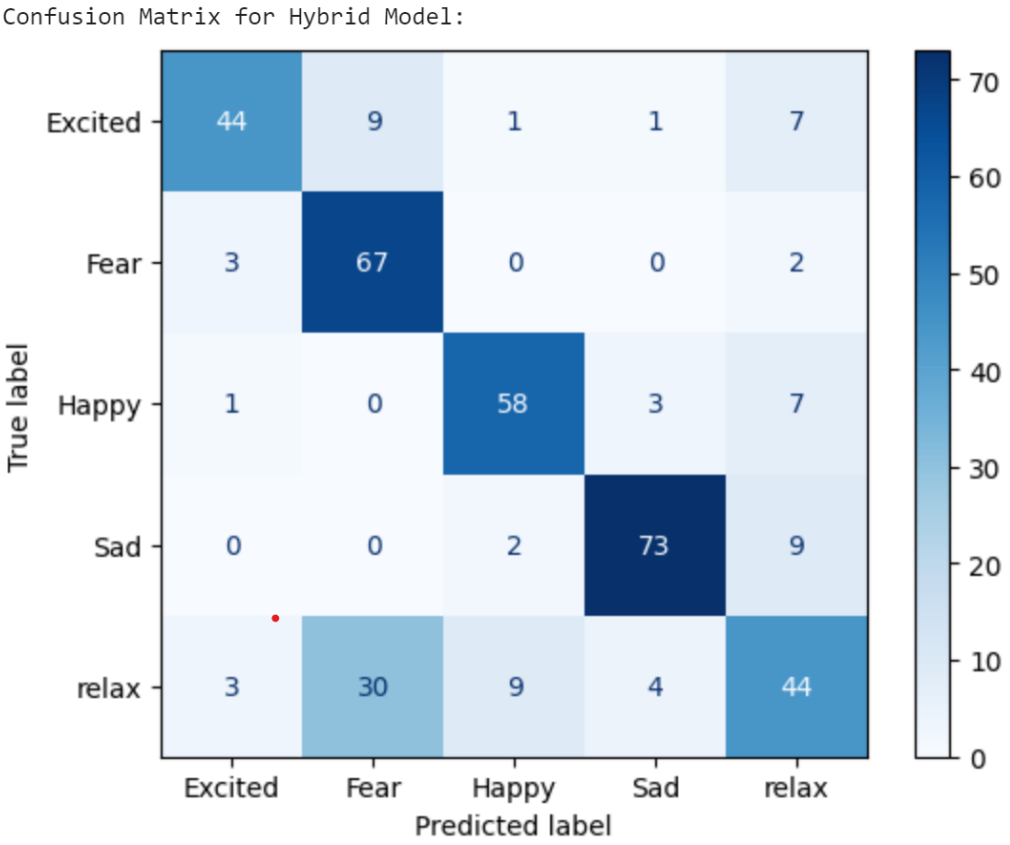


Figure 3.21: Confusion matrix of Hybrid model based on feature extraction.

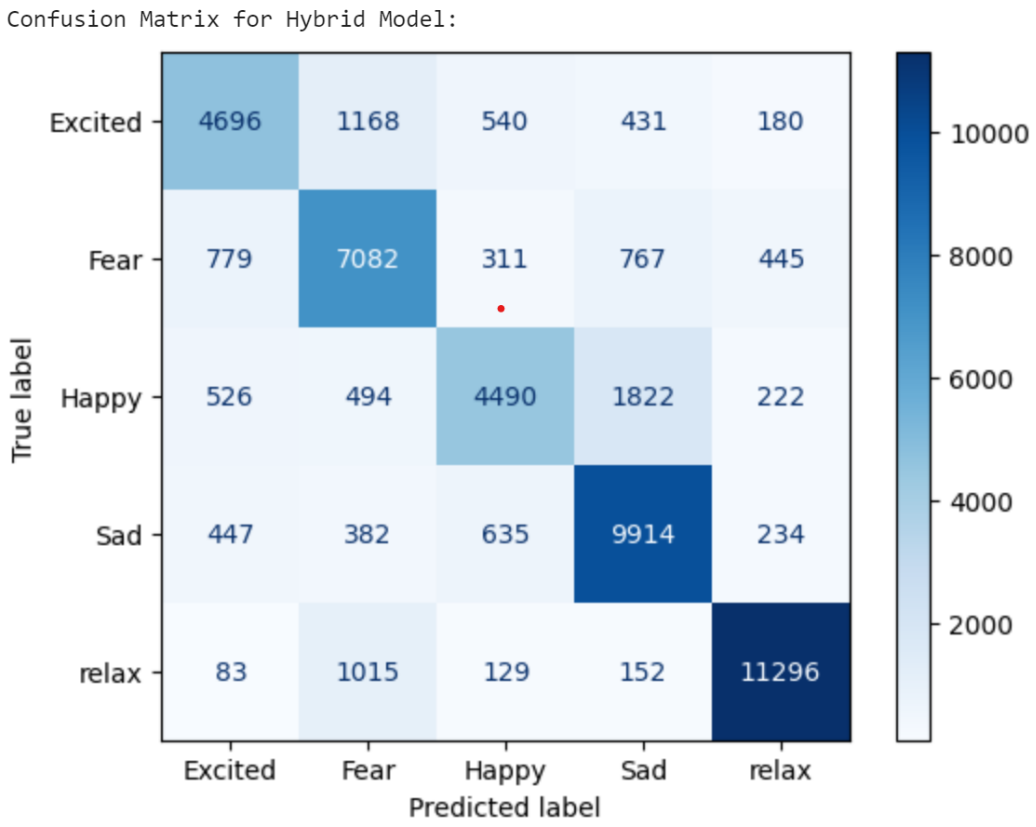


Figure 3.22: Confusion matrix of Hybrid model based on raw data.

3.5.7 Ensemble Approach

The ensemble approach combines the accuracy of the models (e.g., CNN, LSTM, and Hybrid Model accuracy) to improve overall accuracy and emotion classification.

Classification Report: The Ensemble approach (CNN+LSTM+Hybrid) Performance for each of the different emotions was represented in the classification. For every class, the classification reports include metrics like support, F1-score, and recall. The details about metrics can be obtained from equations mentioned in 3.5.1. Here, Class 0: excited, Class 1:fear, Class 2:happy, Class 3: relax, and Class 4: sad. Fig. 3.14 is the classification report based on Feature Extraction and Fig. 3.15 is the classification report based on raw data.

Confusion Matrix

The confusion matrix is a performance evaluation tool. It discusses predicted labels in comparison to actual labels. The confusion matrix is used in multi-class classification. The Confusion matrix provides True positives, true negatives, false positives, and false negatives for each class. Fig. 3.23 is the confusion matrix with feature extraction and Fig.3.24 is the confusion matrix with raw data.

Class	Precision	Recall	F1-Score	Support
0	0.71	0.66	0.68	62
1	0.61	0.86	0.72	72
2	0.83	0.71	0.77	69
3	0.90	0.87	0.88	84
4	0.62	0.53	0.57	90
Accuracy			0.7241	

Table 3.14: Ensemble model classification report based on feature extraction.

Class	Precision	Recall	F1-Score	Support
0	0.79	0.79	0.79	7015
1	0.84	0.83	0.84	9384
2	0.79	0.73	0.76	7554
3	0.83	0.90	0.86	11612
4	0.96	0.93	0.95	12675
Accuracy	0.8520			

Table 3.15: Ensemble model classification report based on raw data.

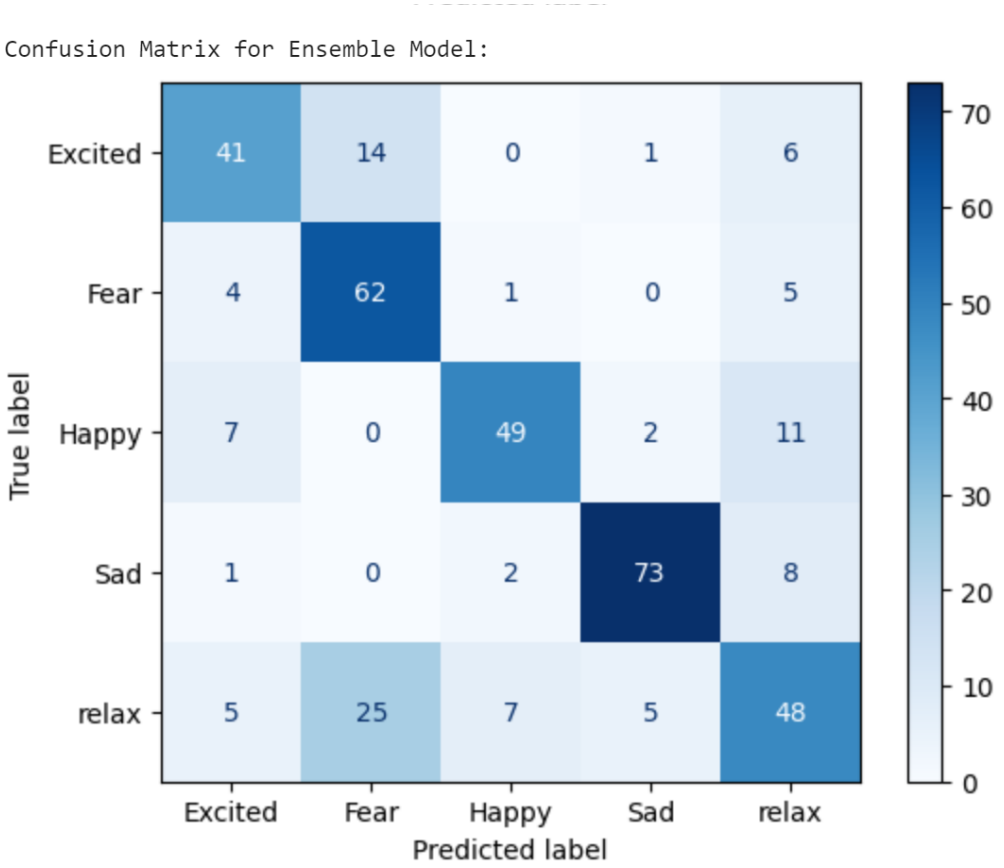


Figure 3.23: Confusion matrix of Ensemble model based on feature extraction.

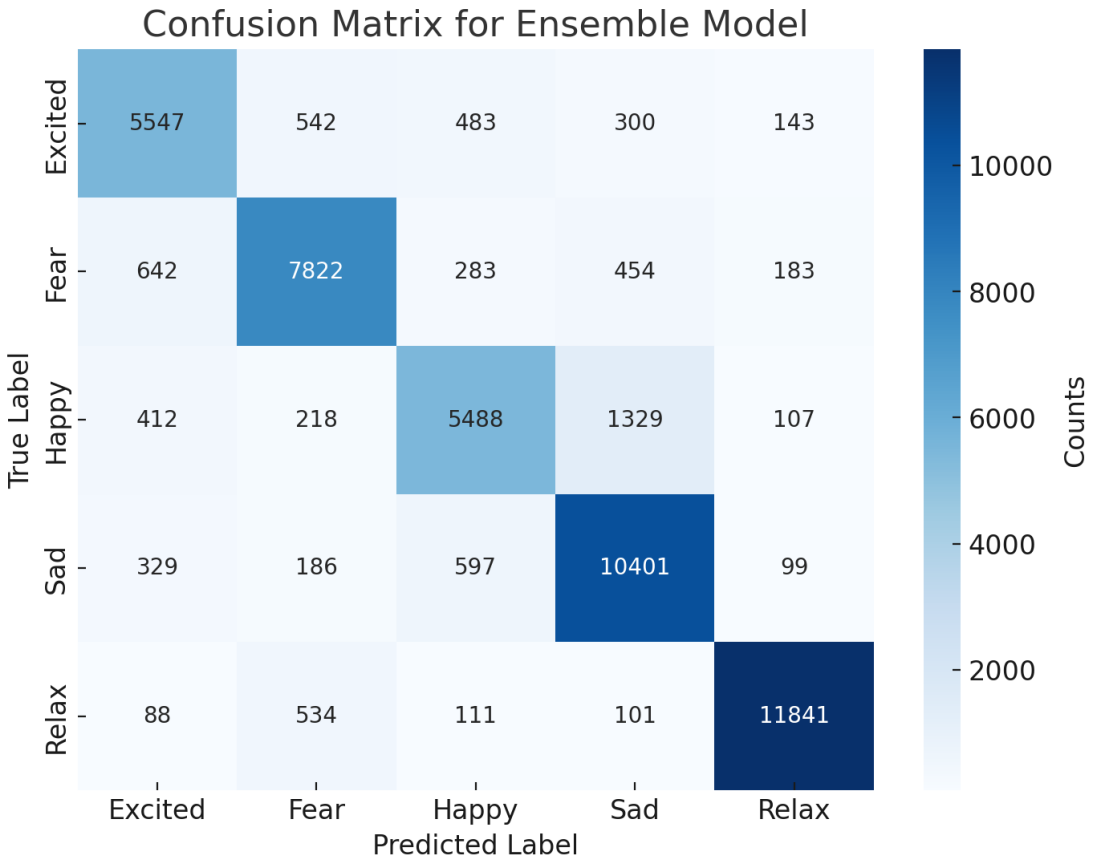


Figure 3.24: Confusion matrix of Ensemble model based on raw data.

3.6 Evaluation

In this section, we compare the performance of different classifiers used for emotion classification based on EEG data, as shown in Table 3.16. The evaluation focuses on the preprocessing techniques applied, the feature transformation methods utilized, the specific features used, and the resulting test accuracies for each classifier. The classifiers are ranked in descending order of test accuracy to highlight the best-performing models.

Chance Level Accuracy:

In a multi-classification task, chance level accuracy represents the probability of correctly classifying an instance by random guessing. This probability is inversely proportional to the number of classes involved in the multi-classification task [BN06].

In this project, the number of classes utilized is Five Class. So, the chance level accuracy for each model is $1/5 = 0.20$ or 20%.

The accuracy of the CNN model is 82%, the LSTM model is 83%, the hybrid model

Classifier	**Feature Extraction**	**Dimensionality Reduction**	**Test Accuracy (%)**
Ensemble (CNN + LSTM + Hybrid)	-	-	85
LSTM	-	-	83
CNN	-	-	82
DNN (Deep Neural Network)	Statistical features, PSD	PCA	80
SVM (Support Vector Machine)	Statistical features, PSD	PCA	79.5
Hybrid (CNN + LSTM)	-	-	77
Hybrid (CNN + LSTM)	Differential Entropy (DE)	-	75
Ensemble (CNN + LSTM + Hybrid)	Differential Entropy (DE)	-	72
CNN	Differential Entropy (DE)	-	71
k-NN (k-Nearest Neighbors)	Statistical features, PSD	LDA	62
LSTM	Differential Entropy (DE)	-	57

Table 3.16: Comparison of Classifiers for Emotion Classification

is 77%, and the Ensemble model is 885%. The Chance level accuracy for each model is 20%. This shows that the models performed well above random guessing.

4 Wristband

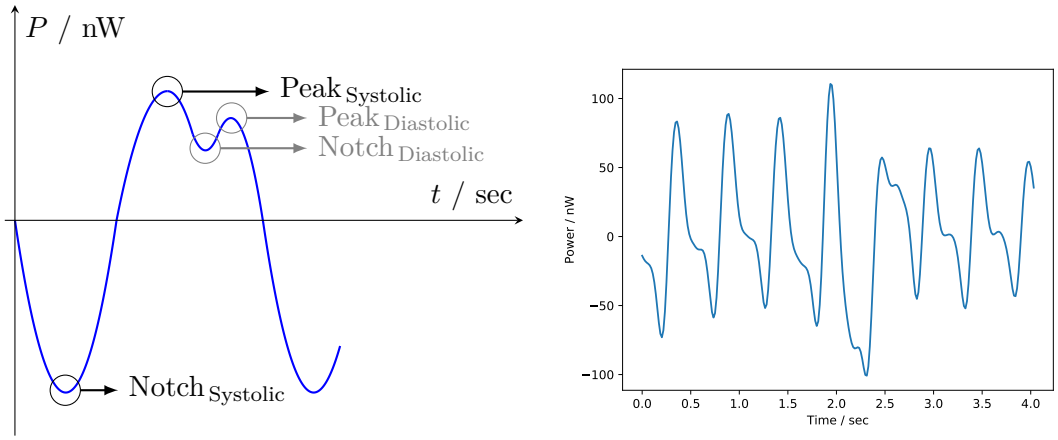
In this chapter the data processing regarding wristband data will be discussed. First in section 4.1 the raw data gathered by the three relevant sensors of the wristband will be presented. After that in section 4.2 data preparation and segmentation will be done using an automated pipeline approach followed by section 4.3 which describes features extracted from gathered segments. As the feature extraction yields a high dimensional feature set, section 4.4 focuses on reducing the amount of features to a reasonable set. Making use of the features selected, classification techniques, which will be presented in section 4.5, can be utilized. Finally section 4.6 discusses the findings of this chapter.

4.1 Sensory Data

The E4-wristband comes with three relevant sensors. These allow to have data about a subjects blood volume pulse, sweat activity and wrist temperature. Furthermore, conducted experiments do not contain a physical active part which ensures not only the main contribution of measured changes to be from purposely induced stimuli but also to have a good quality contact between the subject and the used sensory for the entire duration of said experiments. This improves overall data quality but even with ideal data a potential problem regarding the rate of change of each quantified metric stays. For example the heart rate, which is part of the BVP information, can change significantly in a matter of seconds, while the temperature of a subject takes up to multiple minutes to change by a tenth of a degree.

4.1.1 Blood Volume Pulse

The PPG sensor also called BVP sensor is a non intrusive way to measure the blood volume at the wrist where the sensor is fixed. It measures the amount of reflected light which is inversely proportional to the amount of blood at the wrist. Not only does the internal E4 algorithm remove the measured average to ensure a zero mean value, but it also inverts the resulting curve. The moment of the highest blood volume measured means the most light got absorbed resulting in the lowest reflected power. Inverting the curve results in upper peaks describing peak blood volumes. This ideally yields an aortic pressure curve as shown in figure 4.1a. The amplitude is mainly dependent on the distance to the heart but also on the thickness and color of the skin. However, the form of the curve itself is usually the same. This means that even at rather high distances to the heart characteristic points of the aortic pressure curve can still be observed and analyzed. The most important points of interest, like the systolic notch, are shown in figure 4.1a. The systolic notch is the minimal point of pressure of each beat. This point



(a) BVP model with points of interest.

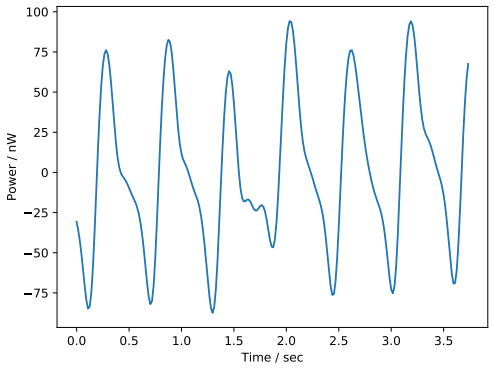
(b) Measured BVP signal.

Figure 4.1: BVP signals comparing ideal and measured. Ideal is model derived by [Edi17][Emp20a]

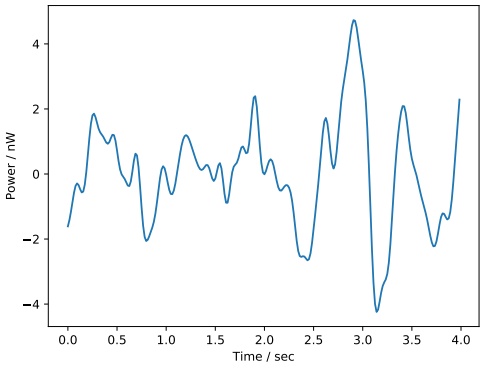
marks where the aortic valve opens or basically where the electrical impulse leading the heart to contract is present. Although this point is slightly after the ECG observable R-peak, it is often referred to as the equivalent point in time. The second important point is the systolic peak, which is the maximum of each beat. This point marks where the heart starts to relax. Another important point is the diastolic notch, which is the local minimum of each beat. The diastolic notch marks where the aortic valve closes as the pressure inside the heart drops below the outside blood pressure. It marks the end of the systole and thus the start of the diastole. The last point of interest is the diastolic peak, which is the local maximum of each beat. This point marks where the heart relaxation stops or basically blood starts to fill the heart again.

It is easily visible that only the systolic peak and notch are clearly identifiable when comparing figure 4.1a and figure 4.1b, while the diastolic peak and notch are guessable but often unclear. There are multiple reasons for that, with the sensor quality itself being the most obvious one. Another one can be the internal algorithm of the E4 device itself visible in figure 4.1b between second two and three. The diastolic peak at 2.4 s just as the systolic peak at 2.5 s is way too low while the systolic peak at 2 s and the diastolic peak at 2.6 s are too high. An artefact caused by movement usually distorts the measured result in a way where the characteristics are not clear anymore. However, in this case for example not only are all the important points still visible but also a zero mean value is kept across the chosen example. It is unclear how exactly Empatica keeps their promise of a zero mean value, thus there is a probability for the E4s algorithm scaling some samples to keep this promise. Assuming this is the case, one of the most important properties of the points of interest, which is their position in time, stays untouched. Thus the worst case would be artefacts that do not lead to a loss of information regarding IBI extraction described in section 4.2.3.

Other artefacts are due to movement or the wristband not worn tightly enough. Figure



(a) BVP signal with a simple movement artefact.



(b) Bad BVP data due to the wristband not worn tightly nor stationary.

Figure 4.2: BVP signals comparing artefacts. Both are actual parts of collected data.

4.2a shows a simple movement artefact, which probably has been caused by moving a finger. This is an example of an uncritical artefact where the systolic notch is still clearly visible just as the systolic peak which has been slightly corrected upwards. Figure 4.2b however shows unusable data. In this recording the wristband has not been worn tightly resulting in a really small amplitude. Additionally looking at the data from the acceleration sensor using Empatica's website, the wristband has been moving around the entire time during the recording. Especially movement makes BVP data useless quite quickly which is why in every experiment the subject shall keep the wristband as stationary as possible.

The obtained BVP data reflects the blood pressure dependent on time having multiple characteristic points regarding heart activity. Not only can it be used in its raw form but it allows to extract other metrics as well such as:

- *Inter-Beat-Interval*
- *Heart-Rate*
- *Heart-Rate-Variability*

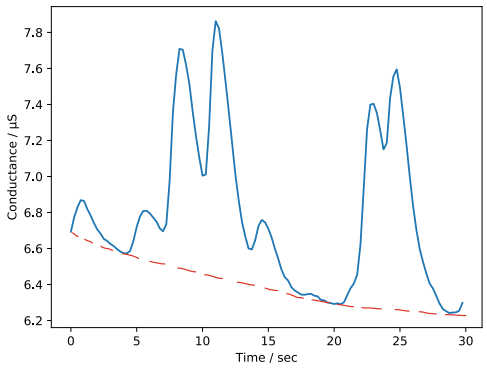
The inter-beat-interval (IBI) reflects the time between two R-peaks, meaning the time between two start points of heart contraction. These are obtained by extracting the time between two systolic notches. The heart-rate (HR) is also known as the pulse which can be felt by pressing slightly at one's arteries. HR can be obtained by using IBI data to directly calculate the amount of beats per unit of time, usually per minute. Last but not least is the heart-rate-variability (HRV) which describes the changes between inter-beat-intervals. It is inversely proportional to the subject's stress [oC21] [Mal96].

4.1.2 Electrodermal Activity

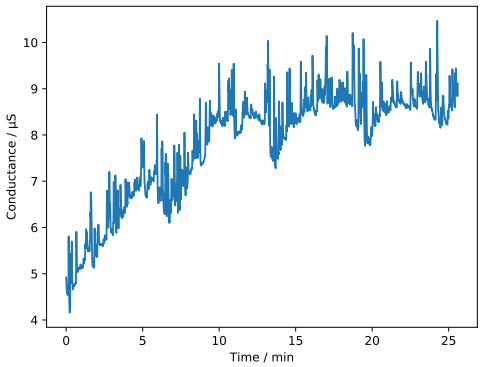
The EDA sensor uses a low frequency current to measure the skin resistance at the wrist. The resulting voltage across the electrodes is measured and thus a resistivity or conductance can be calculated. It is used to estimate the sweat activity of a user. Unlike BVP data from which different metrics can be derived, EDA data is usually just split into its two key components:

- *Galvanic Skin Conductance* (GSC)
- *Galvanic Skin Response* (GSR)

Galvanic skin conductance also referred to as *tonic skin conductance level* or just *tonic component* describes the general skin conductance level including low frequency components. It is hand drawn into figure 4.3a as the dashed red function showing a decrease by 400 nS over 30 s of time. The remaining part is the galvanic skin response also referred to as *phasic skin conductance response* or just *phasic component* [BWJR13][Emp21]. It consists of fast changing reaction peaks due to stimuli associated with rather strong emotional arousal or high attention [LSMG09]. Typically those peaks have a steep rise and an exponential decline back to the basic conductance level after the peak itself [LFF12]. The first but also the third smaller peak in figure 4.3a show this properly. Here it is also apparent that those peaks can interfere with each other, shown by the big double peaks. Figure 4.3b shows the general trend of skin conductance over an entire session. Apart from the flimsy peaks and clearly visible trend contributed by both components, the slow rate of change of the tonic component becomes apparent. The existence of reaction peaks is rather independent from the overall skin conductance level and changes in a



(a) EDA signal of a single segment of 30 seconds. The red function is hand drawn and represents the basic skin conductance level on which the conductance response adds on to.



(b) EDA signal of a 26 minute session.

Figure 4.3: EDA signals taken from actual measurements.

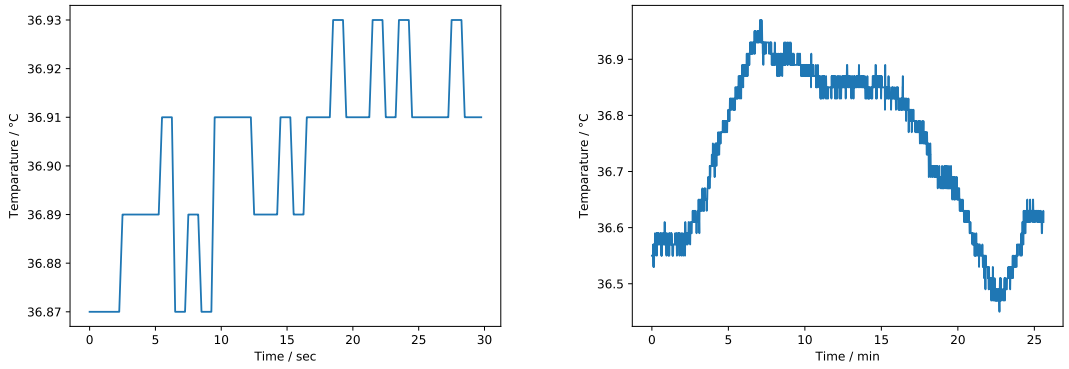
matter of seconds just as the heart rate for example. The tonic component however changes by about $0.8\,{}^{\mu S}/_{min}$ in the shown segment and by about $0.4\,{}^{\mu S}/_{min}$ in the first 10 minutes of the shown session. Assuming the ideal tonic component of the specific subject between two induced emotions lays 1 µS apart, a relaxation period greater than the used 60 s is needed to ensure the possibility of the correct skin conductance level for said emotion. Thus the beginning part of displayed videos or relax phases are used as establishment time in which the tonic component changes the most and are later cut away.

4.1.3 Temperature

The temperature sensor of the E4 device is placed right next to the BVP sensor at the bottom of the main body. It is implemented by using a thermophile sensor which changes its output voltage accordingly to detected infrared light. A single segment and an entire session are shown in figure 4.4. Here it becomes apparent that most of the sensor's usable range ($-40\,°C \ldots 115\,°C$) is unfit for measuring human body temperature as only values from $35\,°C \ldots 38\,°C$ are of interest. Furthermore using figure 4.4b the entire session lays between 37 °C and 36.4 °C meaning that according to 4.1 less than 32 different values, or a resolution smaller than 5 bit, is used.

$$\frac{\triangle T}{\text{size}_{\text{step}}} = \frac{37 - 36.4\,°C}{0.02\,{}^{°C}/_{\text{step}}} = 30\,\text{steps} \leq 32\,\text{step} \equiv 5\,\text{bits} \tag{4.1}$$

Segments often contain only 3 to 4 different values for temperature due to the bad usable resolution for values of interest. Furthermore using the session displayed in figure 4.4b the rate of change between the biggest and lowest value achieved is comparable to the rate of change of the tonic skin conductance level discussed previously. Additionally here



(a) Temperature signal of a single 30 second segment.

(b) Temperature signal of a session of about 26 minutes.

Figure 4.4: Temperature signals taken from actual measurements.

even opening a window in the room used for experiments can influence the temperature significantly, making a controlled environment a key priority during experiments.

4.2 Data Acquisition Pipeline

As the implemented approaches regarding classification can not work properly on lengthy raw data, it is common to cut it into many smaller pieces first. These small pieces are called segments which are made of equal length. From those features, reassembling properties of the segments, can be extracted to continue as it will be shown in the next section 4.3. Just as the GUI from section 2.2.3 is an automated approach to gather as much data as possible, processing it into a specific form suitable for classification has also been deemed to be automated. Therefore, a pipeline has been realized whose general structure is shown in figure 4.5. Currently, the data acquisition pipeline provides segments using the raw data at its native sample rate. To support as many classification techniques as possible, changing the sample-rate to an equal but arbitrary value is planned but not yet implemented. Another functionality which can be included in the future is the feature extraction which currently is implemented separately in multiple different files. Combining as many used functionality into an automated approach makes testing with different parameters easier as selecting different sets of metrics, different parameters for those metrics, different segment lengths or even different sets of subjects can be predefined into a single configuration file. This way even using a series of different configuration files can be done automatically without manually selecting multiple scripts one by one. Currently, the pipeline can be copy pasted into a user folder, where the session.csv and at least one recorded session is present. From there it can be directly executed. Results only depend on the parameters at the head of the *pipeline.py* file. This means as long as the general environment, ensured by the previously mentioned GUI, is present the pipeline will work out of the box on any machine having the necessary *Python* modules installed.

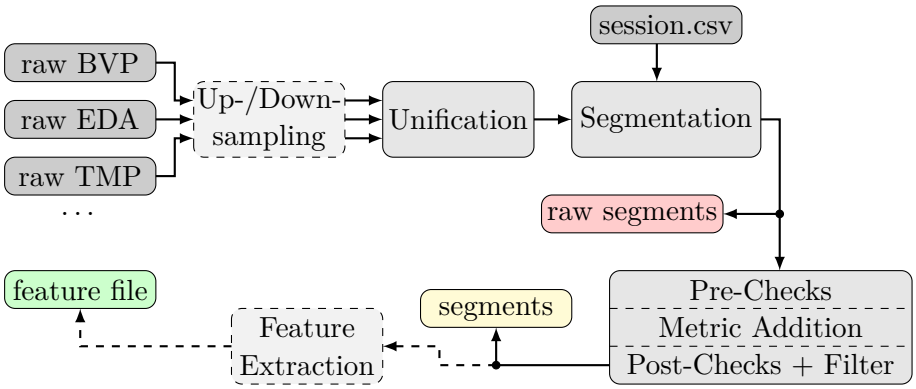


Figure 4.5: Simplified data acquisition pipeline. Dashed elements are not included yet.

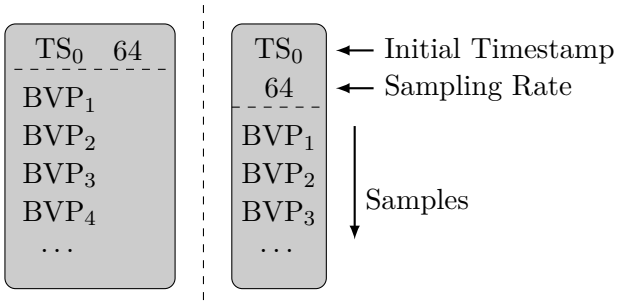


Figure 4.6: Raw BVP file format. Left is the real time recorded format, right is Empaticas online format. The pipeline can use both formats.

4.2.1 Pre-processing

Before the raw sensory data captured by the wristband can be used for segmentation, it needs to be prepared. Its initial saving format is shown in figure 4.6. In the proposed pipeline, preparation consists of two steps. The first one is the option to convert the sampling rate of each raw input stream of data. This has not been required so far and consequently has not been implemented yet. This means that for every step after data collection the native sample rate of $64\,{}^{\mathrm{Sa}}/_{\mathrm{s}}$ for BVP data and $4\,{}^{\mathrm{Sa}}/_{\mathrm{s}}$ for EDA and temperature data has been used. Figure 4.6 shows the saving format for a raw BVP file. Consequently, the second step of preparation, called unification, can not just combine those streams without gaps. Furthermore, the initial raw data streams are collected sequentially by the E4 device meaning no timestamp of a sample perfectly matches another timestamp of any data stream.

To circumvent this issue the exact timely information gets loosened in a format shown in figure 4.7. The unification combines each raw data stream in chunks of one second, meaning the later segmentation can cut any multiple of one second as segment length. Timestamps ranging from *hh:mm:ss.000000* to *hh:mm:ss.999999* are considered part of the same second. The milli- and micro-second information is only used for timely order inside of the chunk and discarded afterwards. This way only one timestamp with a precision of a second has to be saved for each chunk as each chunk can be the beginning of a segment later on. Inside of a chunk all data streams fill their respective column from the top down till there are no values left for this second of the specific stream. This means that only the first row can be considered from roughly the same point in time. As for example EDA and BVP have different sample rates the 2^{nd} EDA value is about from the same point in time as the 17^{th} BVP value. The amount of rows of the combined file is defined by the highest sampling rate of the individual raw files. Gaps from data streams of lower sample count are filled with dashes till the end of each chunk. This approach of saving has been chosen for two reasons. The exact point in time of each sample is not relevant for our application of emotion recognition, since we assume that the changes in the modalities in response to different emotions are slower than

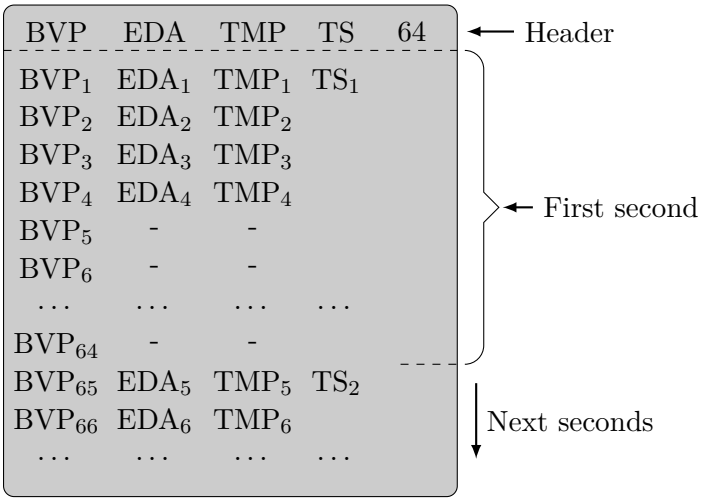


Figure 4.7: Beginning of a unity file. Only necessary format elements are shown.

one second. Consequently, saving each individual point in time would be unnecessarily complex. As long as the timely order of each stream is undisturbed no useful information is lost. The second reason is pure simplicity of the writer and reader of files. Even data of unevenly sampled information, which is not the case with raw data but will be the case with later added metrics such as IBI, can be saved in a fairly simple matter. The reader of a file does not have to search the entire chunk as the first dash of a column marks the end of information of the column in this chunk.

The unified format takes as many raw streams as there are specified in the configuration of the pipeline. New metrics can easily be added into a new column as long as the timestamp column and sample rate header are at the outer right part of the file. This way other devices than the currently used E4 wristband are supported by the pipeline. As one core idea of this project is to combine the output of multiple devices and Empatica's official support of their wristband ends before the end of this project anyway, using a flexible approach has been an initial design choice. For each session an individual unified file will be created as there can not be any gaps between seconds. This way jumping between individual sessions is easy as they are saved with the first timestamp in their name, such as *unity_raw_1720794911.csv* in a user folder.

4.2.2 Segmentation

The next step of the data acquisition pipeline is the segmentation of data. This functional block takes all unified files alongside the session file as inputs to cut the desired segments. Cutting happens based on the start and stop timestamps saved in a user's session file for each video or relax phase. Ideally, data for every desired part in a session file is present. Newer, older or missing sessions yield a notification but do not hinder execution. Cutting happens from the oldest video or relax phase in the session file to the newest.

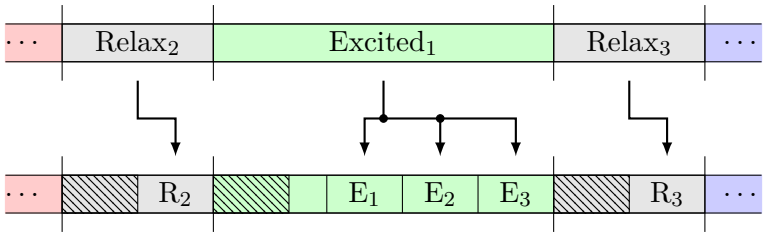


Figure 4.8: Example segmentation using 30 s cutaway and 30 s segments.

As mentioned in chapter 2.2.2 and shown in section 4.1 data at the beginning of a video might be influenced by the previously induced stimuli. To eliminate this effect as much as possible the beginning of each phase or video is cut away. Furthermore, as the video length minus cutaway almost never is a multiple of the desired segment length, a small overhead is left. This overhead is too small to cut another segment and as earlier parts of a video might contain unwanted previous stimuli, it is used as an extension to the cutaway part. This process is visualized in figure 4.8.

Here 30 s cutaway and 30 s segments are used to segment 60 s relax phases and a 135 s video. The relax phase duration is a multiple of 30 s and thus it does not have an overhead part nor multiple segments. Applying cutaway to the video leaves 105 seconds of useful data. It will be segmented beginning from the end towards the start based on the assumption that later video parts are only influenced by the wanted stimuli. This means E_3 is the first cut segment leaving 75 seconds of useful data continuing with E_2 leaving 45 seconds and so on. This resumes until the remaining duration is smaller than the needed segmentation length. Doing so 15 s of samples are left as wasted data resulting in 45 seconds of effective cutaway and 3 segments of 30 s length each for this video. Although E_3 has been the first cut segment it doesn't get the index 1 to keep naming in a timely ordered matter. Finally, the segmentation functionality yields raw segments named *happy_1.csv* or *sad_5.csv*. Their format is equivalent to the unity files format only differing in length being a specific integer of seconds.

4.2.3 Checks And Additional Metrics

The final currently implemented element in the pipeline is used to add additional metrics based on the raw data streams to the segments. To do so this block consists of multiple parts, namely prechecks ensuring the segments can be used, the actual addition of metrics and finally postchecks which mainly consist of a filter mechanic. This procedure is used on all raw segments individually converting them to 'normal' segments.

Prechecks are checks ensuring that the following functionalities, mainly the addition of other metrics, can be executed reliably. Currently, three kinds of checks are implemented. The first one ensures that the segment length, based on the amount of rows, matches the expected amount. After that, the presence of each expected sensor is looked up and finally, each column is checked for missing data. If just one of these checks is

failed, the specific segment is assumed to be a bad segment and will be removed.

To prevent removing segments causing gaps in the naming scheme counting up the indices and thus assuring an implicit knowledge of all segment's names, removing also includes renaming. Assuming five *sad* segments are present, the first segment to be checked will be *sad_5.csv* followed by the fourth and so on. In case of the segment with the highest index failing the tests, removing does not require any renaming as it has no successor. However, if for example number two fails the checks its removal leaves a gap which will be filled by number five renamed to *sad_2.csv*. Always the segment with the highest index of a specific class will be used as filler as it has passed the checks previously. Consequently, this way does not keep a timely order of said segments as a later one might get a lower index than earlier segments. However, this does not yield any problems for later processing as the specific order or time of the individual segments towards each other is irrelevant. Previously kept timely matter regarding their naming scheme has been ensured to keep debugging as easy as possible.

After guaranteeing that all segments present are usable based on there core characteristics, more columns containing additional metrics can be added. For that at least four different options are implemented or planned, consisting of:

- Inter Beat Interval (IBI)
- Heart Rate (HR)
- Heart Rate Variability (HRV)
- Seperation of EDA into GSC and GSR

Currently IBI and HR are implemented. HRV will be implemented in the upcoming semester and the separation of EDA currently is done in the feature extraction part using a third-party tool. It further splits up the GSR component and might be used for comparison of a self developed option if necessary.

Inter beat interval also called IBI describes the interval between two heartbeats. Usually, it is measured using an electrocardiography (ECG) device as the timely difference between two R-peaks. R-peaks describe the electrical activity of the heart when contracting thus they refer to a beat. Using BVP data the systolic notches can be used instead. They are slightly offset to the actual R-peaks but the offset is basically a constant. However, due to artefacts those systolic notches aren't always as easy to identify as figure 4.2a might imply. For that a self developed algorithm filtering all viable beat candidates, considered systolic notches, to calculate IBI is used. This algorithm works in two phases. First, it extracts all notches and peaks it can find in the segment. In the first phase it filters these by the following assumptions:

1. Notches or peaks considered have to be at least 21 samples apart from each other.
2. The first value considered is a notch, the last a peak.
3. Notches and peaks always appear in pairs following each other.

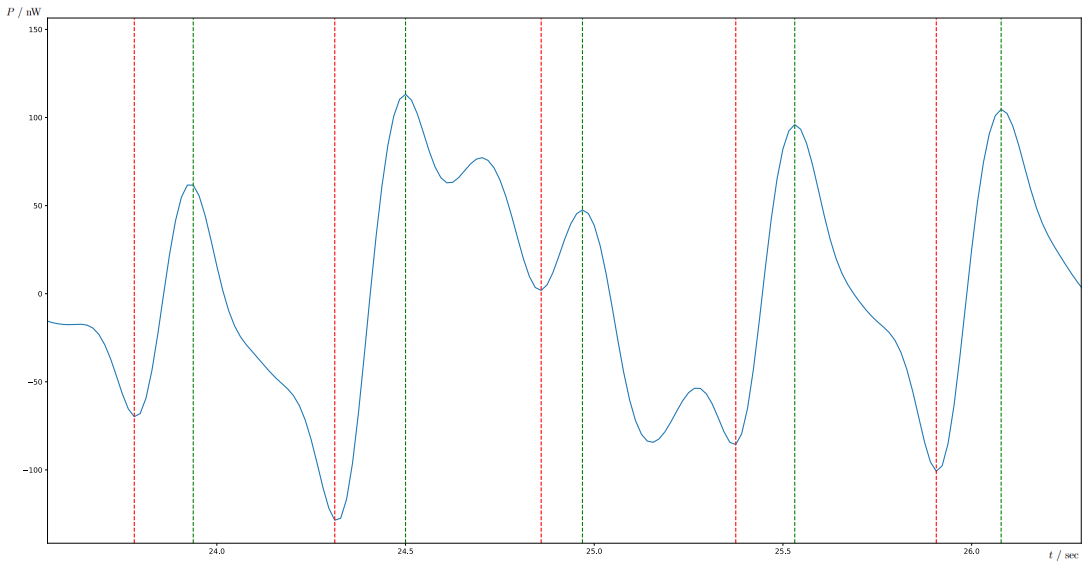


Figure 4.9: Phase one filtering used on BVP data containing an artefact. Remaining notches or peaks, marked with dashed lines, only exist in pairs.

The first assumption is based on the intention of accepting a subject's instantaneous heart rate of up to 180 $^{\text{Beats}}/_{\text{min}}$. It is ensured by using a window of size 21 moved over the BVP data of a segment. Only the notch with the highest absolute value inside the window is kept. This step is repeated for the peaks as well. The second assumption eliminates unnecessary entries based on the fact that a leading peak without a notch is useless and a notch without a following peak is hard to judge. This is literally done by deleting every leading peak or ending notch from the stored candidates. The last assumption relies on the fact that a systolic notch and a systolic peak as a pair reflect the heart contraction activity. Every valid notch is followed by a peak and after each pair follows another pair. This means that there can not be multiple valid notches or peaks following each other. In case of multiple peaks following each other, the later peak has no own preceding notch and thus is eliminated. In case of multiple notches, only the latest one closest to a peak is considered and the earlier notches without their own peak are deleted. An example can be seen in figure 4.9. The first assumption eliminated the peak between the 2$^{\text{nd}}$ and 3$^{\text{rd}}$ pair and the notch between the 3$^{\text{rd}}$ and 4$^{\text{th}}$ pair. The third assumption eliminated the single notch or peak candidates that were left in this case. Only notch-peak pairs remain at this point thus a naive implementation could use the gathered notch candidates straight away. However, the algorithm at hand filters further with a second phase.

In the second phase the characteristics of each notch-peak pair, shown in figure 4.10, is analyzed and compared to the previous pair. The first candidate is set as a starting value. From this onward the next candidate is assumed to have similar characteristics as the previous one. The algorithm checks for similarity by using a threshold around the last candidate's characteristics. If the current checked candidate's notch value for example is close to the last one it will be inside the threshold window and thus the new threshold will be based on it. If it fails to be inside this threshold window, it will be marked for failing said test. The threshold window will be increased by 10% for each candidate failing to be within it. This is done for all three tests per candidate where currently failing just one test is considered enough for elimination.

However, there are multiple potential issues with the approach as it is implemented right now. The first one is its reliability on the first chosen candidate without further consideration. If its notch value is much bigger than later notch values the following candidates will be eliminated till the threshold covers them which can take up to 15 candidates in an extreme case. Although these cases are rare as only one of those segments exists currently, it is an issue leading to lose potentially good segments. Another issue is the fact that the tests of the second phase are considered equal in weighting. Due to the limited sensor quality and or the E4 algorithm potentially scaling values the notch and peak values can be varying quite a bit for just one candidate. This might lead to this candidate or the following one being outside of the threshold and thus being eliminated. Only the heart compression time is unaffected by this as the timely position of the notch and peak is basically never affected. Additionally, candidates such as the 3rd pair in figure 4.9 are eliminated as artefacts, although its position compared to the other candidates seems reasonable. Tackling these issues by for example using a static threshold based on the top 10 candidates or using the current approach (with different weighting) to have multiple candidate groups to use for further runs, which can be compared to each other, is under testing.

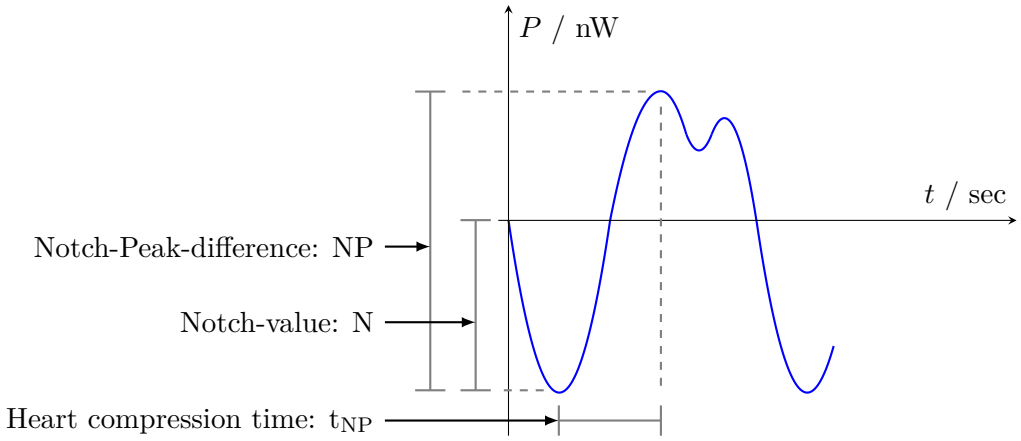


Figure 4.10: Used BVP characteristics to determine IBI candidates.

Whatever exact approach will be used later, in an ideal case it leads to many candidates representing a heartbeat each. As described earlier only the notch is used to calculate IBIs. The middle of said interval is used to address its position in time. For example an IBI based on the notches at 2.5 s and 3.3 s would be addressed at 2.9 s and thus added to chunk number 3. This leads to uneven filling of the chunks of a segment which can not be prevented but is accepted by the saving format itself.

Heart rate is another metric which can be added based on the previously gathered IBI information. Usually, it is done as the instantaneous heart rate ($\mathrm{HR}_{\mathrm{inst}}$) but will be done as a pseudo instantaneous heart rate ($\mathrm{HR}_{\mathrm{inst,p}}$) or even windowed heart rate ($\mathrm{HR}_{\mathrm{win}}$) to keep one result per second. The instantaneous heart rate is defined as:

$$\mathrm{HR}_{\mathrm{inst}} = \frac{{}^{\mathrm{sec}}/_{\mathrm{min}}}{{}^{\mathrm{sec}}/_{\mathrm{Beat}}} = \frac{60}{\mathrm{IBI}}\,{}^{\mathrm{Beats}}/_{\mathrm{min}} \tag{4.2}$$

It directly converts IBI values into HR values. However, this would keep the same uneven filling of a segment's chunks as IBI data does. To prevent this and consequently achieving 1 heart rate value per second, a pseudo instantaneous heart rate is used for each chunk:

$$\mathrm{HR}_{\mathrm{inst,p}} = \frac{60}{\frac{1}{L} * \sum_{l=1}^{L} \mathrm{IBI}_l}\,{}^{\mathrm{Beats}}/_{\mathrm{min}} = \frac{60}{\overline{\mathrm{IBI}}}\,{}^{\mathrm{Beats}}/_{\mathrm{min}} \tag{4.3}$$

Here the arithmetic mean is used for all IBI values inside of a second. This yields the desired result under the assumption that every second has at least one IBI value. Real results do not fulfill this assumption as the underlying data might be of bad quality. To tackle this issue a windowed heart rate can be used:

$$\mathrm{HR}_{\mathrm{win}} = \frac{60}{\frac{\sum_{w=0}^{2W}\sum_{l=1}^{L_w}\mathrm{IBI}_{w,l}}{\sum_{w=0}^{2W} L_w}}\,{}^{\mathrm{Beats}}/_{\mathrm{min}} = \frac{60}{\frac{1}{L'} * \sum_{m=1}^{L'} \mathrm{IBI}_m}\,{}^{\mathrm{Beats}}/_{\mathrm{min}} = \frac{60}{\overline{\mathrm{IBI}'}}\,{}^{\mathrm{Beats}}/_{\mathrm{min}} \tag{4.4}$$

This approach uses not only the chunk for which a result is calculated but also its surrounding neighbours equally to both sides as an extension. Using a window size of $W = 0$ means no neighbour is used. The windowed approach of equation 4.4 then is identical to the pseudo instantaneous approach from before seen in equation 4.3. Good data should not have multiple consecutive empty chunks thus a window size of $W = 1$ is used by default. This means only direct neighbours of a chunk are used. Using windows however smooths data out meaning information is lost. Therefore, the windowed approach is a compromise of losing only as much information as needed to get an acceptable segment past the final filter. Better approaches which can be implemented in the future would be to only use the windowed approach when needed, interpolating missing HR values or to guess an evenly sampled IBI function from which HR values can be directly calculated. The later options might allow for an even higher rate than just $1\,{}^{\mathrm{Sa}}/_{\mathrm{sec}}$.

Postchecks currently are just implemented as a filter. Instead of checking the integrity of each segment again, like it has been done in the prechecks, currently it is

assumed that the integrity is given. The idea of the postchecks is to use the previously gathered IBI and HR information to check whether each segment is good or not. This is done by counting how many IBI and HR values are not inside of a pre defined window. This counter then is divided by the total amount of values for IBI or HR respectively. If the relative amount of bad data for each IBI or HR is bigger than the acceptance threshold, a segment is eliminated as a bad segment. Eliminating a bad segment is done just as in the prechecks with gap prevention using renaming. Concrete default values for the filter are:

- IBI window: $0.5\,\mathrm{s} \ldots 1\,\mathrm{s}$
- HR window: $60\,{}^{\mathrm{Beats}}/_{\mathrm{min}} \ldots 120\,{}^{\mathrm{Beats}}/_{\mathrm{min}}$
- Default acceptance threshold: $10\,\%$

In properly done sessions where a subject's behaviour stays as close to ideal as possible, almost all segments are kept. The latest session for each project group member has an acceptance rate of $96\,\% \ldots 100\,\%$ yielding between 50 and 60 30 s-segments each. These segments are still saved in the same format as raw segments and ideally contain more columns. They can be used for feature extraction described in the following section 4.3 which currently is not part of the pipeline.

4.3 Feature Extraction

Feature extraction is the process of transforming raw high-dimensional information into a lower-dimensional representation. In our case, it involves extracting representations from data that is well suited for the task of classifying emotions. The features extracted can broadly be classified into two categories, time-domain and frequency-domain. Time domain features involve features extracted from raw time-series signals, whereas, frequency domain features involve the features extracted after transforming the signals into the frequency domain using a Fourier transform. This section presents the preprocessing steps applied and continues with the time-domain and frequency-domain features extracted.

4.3.1 Preprocessing

Before the features can be extracted from the signals, the signals are preprocessed to remove noise and additionally normalized.

BVP Signal

The BVP signal is initially filtered using a Butterworth bandpass filter to eliminate low-frequency components as well as high-frequency components that are not relevant. A 5th-order Butterworth bandpass filter is chosen to operate between the frequency ranges of 0.4 and 10 Hz. This is done to ensure the resulting signal only contains the important

aspects while balancing the complexity of the filter. As shown in figures 4.11a and 4.11b, the low-frequency components dominate the PSD periodogram for unfiltered BVP data, but are completely removed by the bandpass filter, resulting in a clear response that can be used for feature extraction. The filtered signal is then normalized to ensure that all segments of the BVP signal are scaled. The normalization scheme used is the min-max normalization scheme shwon in figure 4.5 [ZWYG18].

$$BVP_{\text{normalized}} = \left(\frac{BVP - BVP_{\min}}{BVP_{\max} - BVP_{\min}} \right) \times 100 \tag{4.5}$$

EDA Signal

Curretnly, regarding electrodermal activity a thrid-party algorithm is used to perform the split into its individual components. More specifically, the convex optimization algorithm *cvxEDA* is used [GVl+16]. It splits the EDA data into three components, namely:

- **Tonic**, which is thee baseline/slow-varying component,
- **Phasic**, which is the rapidly varying component driven by some minor stimuli
- **Sudomotor nerve activity** (smna), which is the spiking activity, producing sweat upon exposure to major stimuli

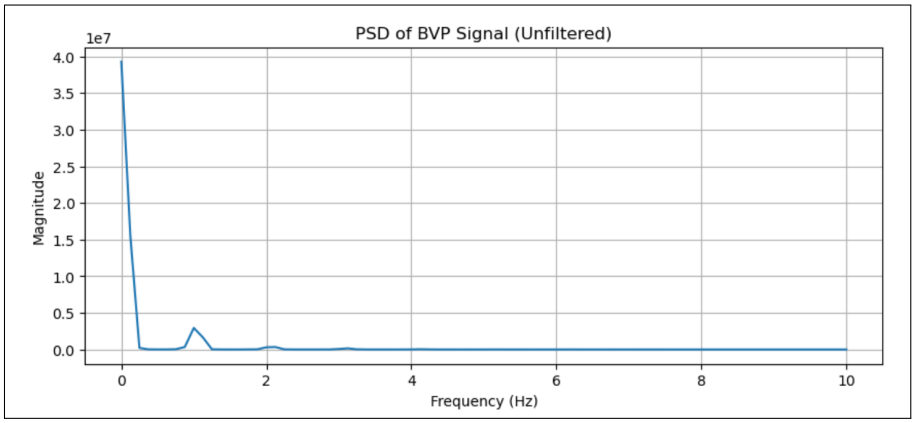
4.3.2 Time-Domain Features

Time-domain feature extraction is the process of extracting raw information about the signal's characteristics regarding short or long term trends. In this case, short term trends of all signals are been analyzed to extract important information. For the purpose of feature extraction, a sliding window algorithm is used to extract our data, dividing the signal into a pre-determined number of chunks and further processing each chunk separately. Aiming to capture the short-term varying behavior of all signals, a window length of 64 samples is taken.

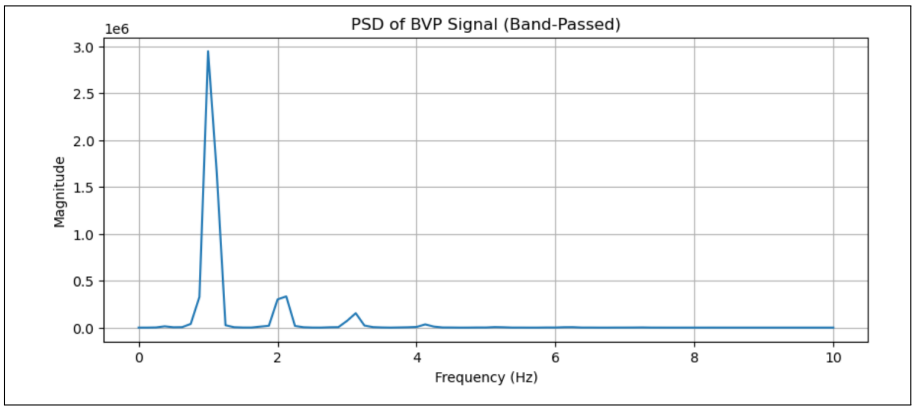
List of Features

Currently, 14 different statistical features are been used, inspired by [BFF+20]. They will be applied on each of the following metrics.

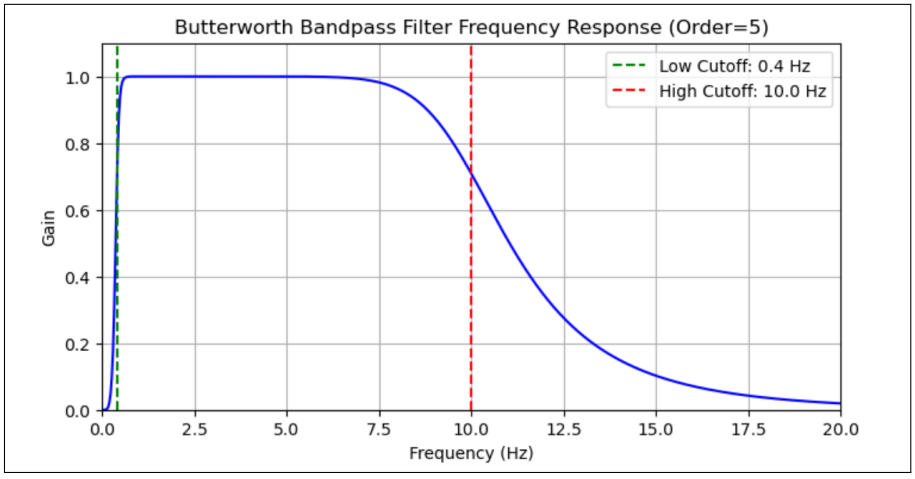
- BVP
- Temperature at the wrist
- EDA tonic component
- EDA phasic component
- EDA smna component



(a) PSD of unfiltered BVP



(b) PSD of filtered BVP



(c) 5th order Butterworth Bandpass filter

Figure 4.11: Effect of filtering on Power Spectral Density of the BVP and the bandpass filter used.

Assuming m is the number of data points of a signal in a used window and x_i is an individual sample, the statistical features are computed as shown in the following equations.

- Mean: Measures the central tendency of data.

$$\mu_x = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{4.6}$$

- Median: Measures the value which is present exactly at the mid-point of data.

- Variance: The square of the average distances between each data value and the mean.

$$s^2 = \frac{\sum_{i=1}^{m}(x_i - \mu_x)^2}{m} = \frac{\sum_{i=1}^{m} x_i^2 - \mu_x^2}{m} \tag{4.7}$$

- Mean Difference: Measures the mean of the differences between data points in a signal.

$$\Delta\mu_x = \frac{1}{m-1}\sum_{i=1}^{m-1} x_{i+1} - x_i \tag{4.8}$$

- Mean Absolute Difference: Measures the mean of absolute differences between data points in a signal.

$$|\Delta\mu_x| = \frac{1}{m-1}\sum_{i=1}^{m-1} |x_{i+1} - x_i| \tag{4.9}$$

- Mean Absolute Deviation: The average distance between each data value and the mean value.

$$\text{Mean Absolute Deviation} = \frac{1}{m}\sum_{i=1}^{m} |x_i - \mu_x| \tag{4.10}$$

- Median of Differences: Measures the median of the differences between data points in a signal.

$$\Delta\text{median}_x = \text{median}\left(x_2 - x_1, \ldots, x_m - x_{m-1}\right) \tag{4.11}$$

- Median Absolute Difference: Measures the median of the absolute differences between data points in a signal.

$$|\Delta\text{median}_x| = \text{median}\left(|x_2 - x_1|, \ldots, |x_m - x_{m-1}|\right) \tag{4.12}$$

- Median Absolute Deviation: Mean of absolute distance between each data value and the median.

$$\text{Median Absolute Deviation} = \frac{1}{m}\sum_{i=1}^{m} |x_i - \text{median}_x| \tag{4.13}$$

- Minimum: Measures the minimum value for the given data.

$$\text{Minimum} = \min(x_1, x_2, \ldots, x_m) \tag{4.14}$$

- Maximum: Measures the maximum value for the given data.

$$\text{Maximum} = \max(x_1, x_2, \ldots, x_m) \tag{4.15}$$

- Skewness: Measures the lack of symmetry in a given data distribution.

  The Fischer-Pearson coefficient:

$$g_1 = \frac{\sum_{i=1}^{m} (x_i - \mu_x)^3 / m}{s^3} \tag{4.16}$$

  If *bias* is set to *false*, an adjusted Fischer-Pearson coefficient is calculated as follows:

$$G_1 = \frac{\sqrt{m(m-1)}}{m-2} \frac{\sum_{i=1}^{m} (x_i - \mu_x)/m}{s^3} \tag{4.17}$$

  Where s is the standard deviation of the data

- Kurtosis: It is a measure of how heavily the tails of a given distribution differ from the tails of a normal distribution.

$$\text{kurtosis} = \frac{\sum_{i=1}^{m} (x_i - \mu_x)^4 / m}{s^4} \tag{4.18}$$

- Inter-Quartile Range: It is the difference between the 75^{th} and 25th percentile of data.

$$\text{IQR} = \text{Q}_3 - \text{Q}_1 \tag{4.19}$$

4.3.3 Frequency Domain Features

To obtain frequency domain features such as power mean values of low-, medium- or high-frequency bands, the time domain signal must be transformed to the frequency domain using the discrete Short-Time-Fourier-Transform (STFT) given by the equation 4.20. The discrete STFT is chosen to ensure that the time component is not lost. To compute the discrete STFT of the signal, it is first divided into a fixed window length section consisting of several samples. A windowing function is applied to this portion of the signal followed by a Discrete Fourier Transform (DFT) to obtain the frequency response.

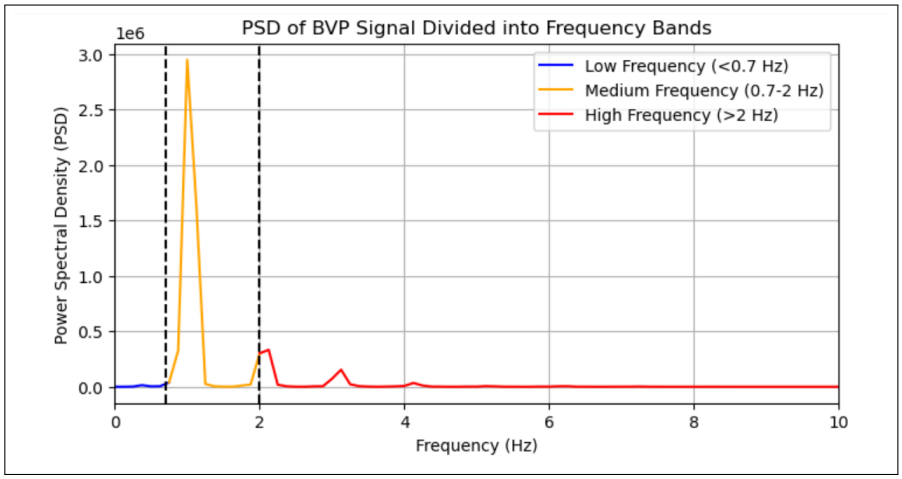


Figure 4.12: Frequency band split

$$\mathrm{STFT}\{x(t)\}(\tau,\omega) = X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)\, w(t-\tau)\, e^{-i\omega t}\, dt \tag{4.20}$$

where:
$x(t)$:= The input signal
$w(t-\tau)$:= The window function
$X(\tau,\omega)$:= The STFT of the signal

Power Mean Value of Sub-bands

In our case frequency domain features are obtained only for the BVP signal. Here a window of size 1024 samples corresponding to 16 seconds is used to segment the BVP signal. A rectangular window is chosen as the window function since spectral leakages are not a concern and having an attenuating window like a Hann-window would alter the power characteristics of the signal. An FFT is computed over the segment of the signal and the PSD is derived from the FFT using the periodogram approach for PSD estimation.

The spectrum is then divided into three bands, namely low-frequency (<0.7 Hz), medium-frequency (0.7 to 2 Hz), and high-frequency (>2Hz) as shown in figure 4.12. The power mean value is computed for each of these sub-bands and the fundamental frequency of each band is noted. The power mean value and fundamental frequency mentioned by [ZWYG18] are crucial features because the frequency of the BVP fluctuates based on the individual's emotional state, making it a key indicator of a person's current emotional state.

4.4 Dimensionality Reduction

There are a large number of features that can be extracted from the BVP, EDA, and temperature signals. In our case, we end up with 98 features resulting in a 98-dimension input vector for each segment. Having such a high dimensional input can result in the *curse of dimensionality* during the classification stage, necessitating an extensive training dataset. Furthermore, to fit a high dimensional input space, extremely complex models will be needed which can lead to other issues such as over-fitting. Hence, there is a need to reduce the dimensionality of the data while at the same time not losing too much information, that will be needed for the classifier. This section presents the explored techniques aimed at reducing the dimensionality of data.

4.4.1 Data Standardization

Before implementing a dimensionality reduction approach such as PCA or LDA, two necessary transformations are required to be performed on our data, namely:

- Mean-Centering
- Standardization to maximum variance

Mean Centering:
Let us assume we have N rows and K dimensions for our data. Initially, we compute the mean, i.e. the average of our data. This K-dimensional mean vector is then subtracted from each N row, to get the mean-centered data.

Unit-Scaling:
In Principal Component Analysis, one of our objectives is to preserve the variance of the original data distribution. PCA is very sensitive to variances across all dimensions, with the dimensions having more variances than others having a proportionate influence on the final model. Thus, we standardize the variance across the data-set by scaling the variance of all dimensions to unit variance.

4.4.2 Principal Component Analysis

Principal Component Analysis (PCA) is one of the most commonly used unsupervised learning techniques today. The feature space we obtain at the end of the previous process is processed using PCA to reduce the dimensions of the data so that it can be trained on traditional classifiers like SVM and KNN. Utilizing our newly transformed data, we will use the *maximum variance* approach stated in [JC16]. We eventually want to project our original data onto a multi-dimensional vector space to maximize the variance. However, for the case of simplification, we initially consider the space on which the data is supposed to be mapped, i.e. $\ell = 1$. Let us assume that X is the original multi-dimensional data, on which PCA is supposed to be carried out. Then, for every i^{th} row of vector X, the

projection of that row onto a one-dimensional space, or a unit vector u_1 is:

$$\text{Proj}_{u_1}(x_i) = \mathbf{u}_1^T x_i u_1 \mathbf{u}_1^T x_i u_1 \mathbf{u}_1^T x_i u_1 \mathbf{u}_1^T x_i u_1, \quad (4.21)$$

Using this expression, we can find the projection of the mean of X on vector u_1 as:

$$u_1^T \bar{X} u_1 \tag{4.22}$$

We know that the variance of a projection is the difference between each X vector and the mean vector of X. Hence, it can be mathematically expressed as:

$$\frac{1}{N} \sum_{n=1}^{N} \left(u_1^T x_n - u_1^T \bar{X} \right)^2 = \frac{1}{N} \sum_{n=1}^{N} \left(u_1^T \left(x_n - \bar{X} \right) \right)^2 \tag{4.23}$$

This expression can be re-written as:

$$\frac{1}{N} \sum_{n=1}^{N} u_1^T \left(x_n - \bar{X} \right) \left(x_n - \bar{X} \right)^T u_1 = u_1^T \mathrm{S} u_1 \tag{4.24}$$

Where S is the closed form of the covariance matrix, and $u_1^T \mathrm{S} u_1$ is the variance to be maximized. The above derivation provides an optimization problem wherein we use a Lagrangian to maximize the variance value. Since we are performing the projection on a 1-dimensional space, our maximization problem is subject to the following constraint:

$$u_1^T u_1 = 1 \tag{4.25}$$

The Lagrangian can be expressed as:

$$u_1^T \mathrm{S} u_1 + \lambda \left(1 - u_1^T u_1 \right) \tag{4.26}$$

Taking the partial derivative wrt. unit vector u_1:

$$\lambda u_1 - \lambda 2 u_1 = 0 \tag{4.27}$$

This gives us the following relationship between u_1 and the covariance S as follows:

$$\mathrm{S} u_1 = \lambda u_1 \tag{4.28}$$

This is a crucial result which states that the direction of u_1 is given by the eigenvalue λ of covariance S. Further simplifying the result:

$$u_1^T \mathrm{S} u_1 = \lambda \tag{4.29}$$

Hence, in order to maximize the term on the left in the above equation, we pick the largest eigenvalue λ for the covariance S, and then choose the eigenvector u_1 corresponding to that eigenvalue. In this way we choose the optimum direction to map the variance $u_1^T \Sigma u_1$ onto our 1-dimensional space.

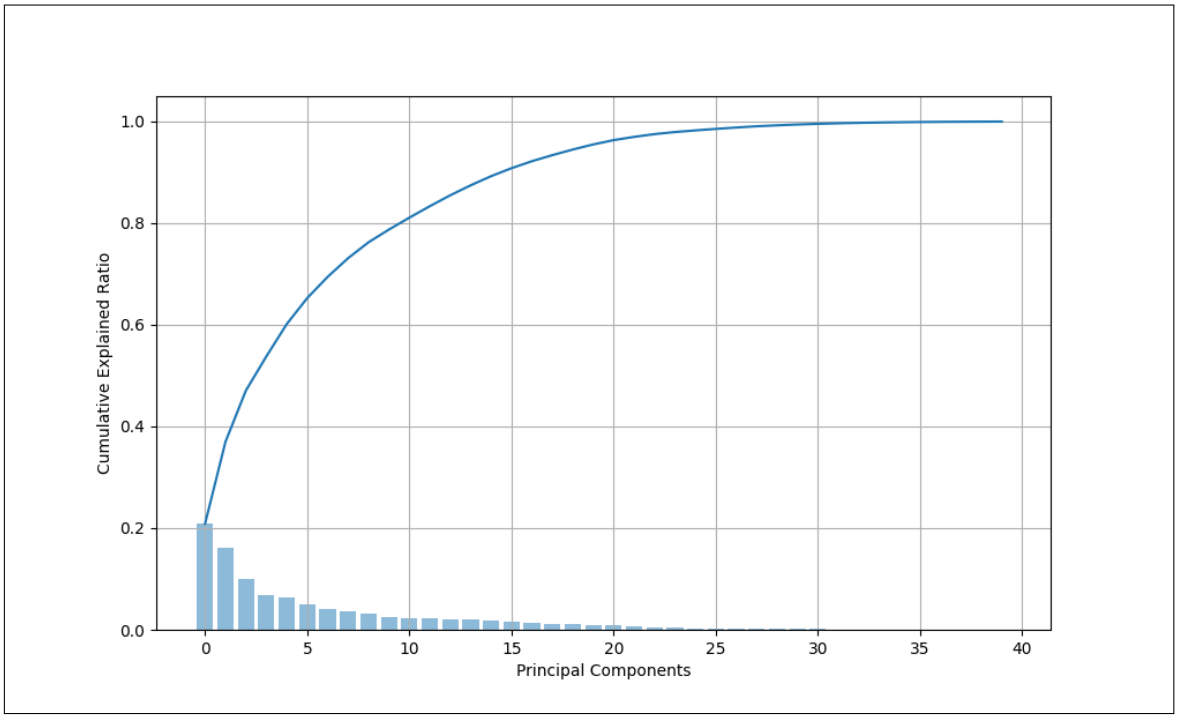


Figure 4.13: Elbow-curve plot showing the individual variances being explained by each component

In order to deal with a case such as ours, where the data is to be mapped onto a multi-dimensional vector space(ℓ = n > 1), we make use of mathematical induction. Instead of picking the highest eigenvalue, we choose the n highest eigenvalues of S, and their respective eigenvectors. Having performed the feature extraction in the previous step of this pipeline, we obtain a feature vector having 98 dimensions. Our objective is to reduce the number of dimensions to a feasible number that can allow us to train a K-Nearest Neighbours classifier on this data. Keeping this goal in mind, we perform the variance maximization process stated above on our feature data. In order to determine the exact eigenvector for each Principal Component, we project our original data onto a 98-dimensional vector space, i.e we retain the original number of dimensions. This enables us to plot the eigenvector of each Principal Component, against the variance explained by all components cumulatively. It can be inferred from the figure 4.13, that nearly 100 percent of the variance from the original data is explained by 40 components only. Some further examination into the percentage of variances explained by the first few components is shown in table 4.1.

The principal component analysis performed so far aids us in filtering out a large number of redundant components or dimensions from our data. From table 4.1, we can infer that the data beyond the first 14 components can be neglected, since nearly 80 percent of the total variance is explained by the former set of components. We

Eigenvalue	Explained Variance
λ_1	0.210
λ_2	0.161
λ_3	0.100
λ_4	0.067
λ_5	0.050
λ_6	0.036
λ_7	0.031
λ_8	0.031
λ_9	0.025
λ_{10}	0.023
λ_{11}	0.022
λ_{12}	0.021
λ_{13}	0.019
λ_{14}	0.017

Table 4.1: Explained Variances for PCA

will utilize this information, paired with another investigative method, to find out our optimum number of components, which we can use to build our classifier.

Evaluation of Components based on classifier accuracy

According to [Lux22], the number of components in PCA has little correlation to the classifier accuracy. Hence, having reduced the number of components from 98 to 14, we look to further narrow down this number, to an ideal range that provides us with maximum accuracy. For this purpose, we run a *Grid Search* to find the ideal number of components to be used. This is a step from the hyperparameter tuning process, which involves including the PCA object in a pipeline consisting of a class oversampler and the classifier object. While implementing a PCA + KNN pipeline on the data, we also require hyperparameters for our classifier model. Upon performing a Grid Search using a 5-fold Cross-Validation, we obtain a total of 1750 iterations, yielding 6 as the ideal number of PCA components. For this search, we use classifier accuracy as the benchmark for judging the hyperparameter performance.

4.4.3 Fisher's Linear Discriminant Analysis

A technique that is often used for dimensionality reduction is Fisher's Linear Discriminant Analysis (LDA). LDA is similar to PCA in that sense that it tries to find projection vectors to project the data into a lower dimensional space than its original. However, unlike PCA which is unsupervised and does not consider the class labels, LDA is a supervised algorithm that evaluates the class labels and aims to maximize the separation between the classes.

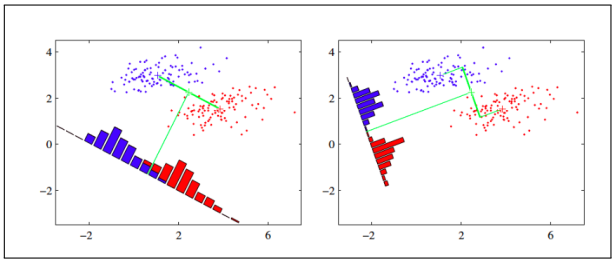


Figure 4.14: Working of Fisher's LDA inspired by [CL14].

As shown in figure 4.14 LDA finds the projection vectors that maximize the between-class scatter while at the same time minimizing the within-class scatter, thereby maximizing the separation between all the classes. The primary steps involved in finding the LDA projection vectors will now be illustrated. The initial step involves finding the mean vectors of all the classes that are present in the d-dimensional training dataset; a random sample($\mathbf{x_n}$) from the dataset is presented below.

$$\mathbf{x_n} = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{d-1} \end{pmatrix} \tag{4.30}$$

The d-dimensional means of all the K classes are then computed by grouping the training examples by class. This results in K, d-dimensional means.

$$\mathbf{m_k} = \begin{pmatrix} \mu_0 \\ \mu_1 \\ \vdots \\ \mu_{d-1} \end{pmatrix} \quad \text{where } k = 0, 1, \ldots, K-1 \tag{4.31}$$

Next two special matrices are computed, namely the *within-class scatter matrix* $\mathbf{S_W}$ and the *between-class scatter matrix* $\mathbf{S_B}$. $\mathbf{S_W}$ measures the amount of variance present within each class distribution, whereas, $\mathbf{S_B}$ measures the amount of variance between the classes, or rather how spread out the means of each class are from each other. The within-class scatter matrix $\mathbf{S_W}$ is given by:

$$\mathbf{S_W} = \sum_{k=0}^{K-1} \mathbf{S_k} \tag{4.32}$$

Where $\mathbf{S_k}$ represents the within class scatter for class k given by,

$$\mathbf{S_k} = \sum_{\mathbf{x} \in D_k} (\mathbf{x} - \mathbf{m_k})(\mathbf{x} - \mathbf{m_k})^T \tag{4.33}$$

Where $\mathbf{m_k}$ represents the mean of class k computed earlier and $\mathbf{x}$ is a sample belonging

Eigenvalue	Explained Variance
λ_1	0.454
λ_2	0.255
λ_3	0.168
λ_4	0.123

Table 4.2: Explained Variances for LDA.

to the dataset D_k of class k. The between-class scatter matrix $\mathbf{S_B}$ is given by:

$$\mathbf{S_B} = \sum_{k=0}^{K-1} N_k(\mathbf{m_k} - \mathbf{m_G})(\mathbf{m_k} - \mathbf{m_G})^T \tag{4.34}$$

Where $\mathbf{m_k}$ is the same as earlier, $\mathbf{m_G}$ is the global mean and N_k is the number of samples present in class k.

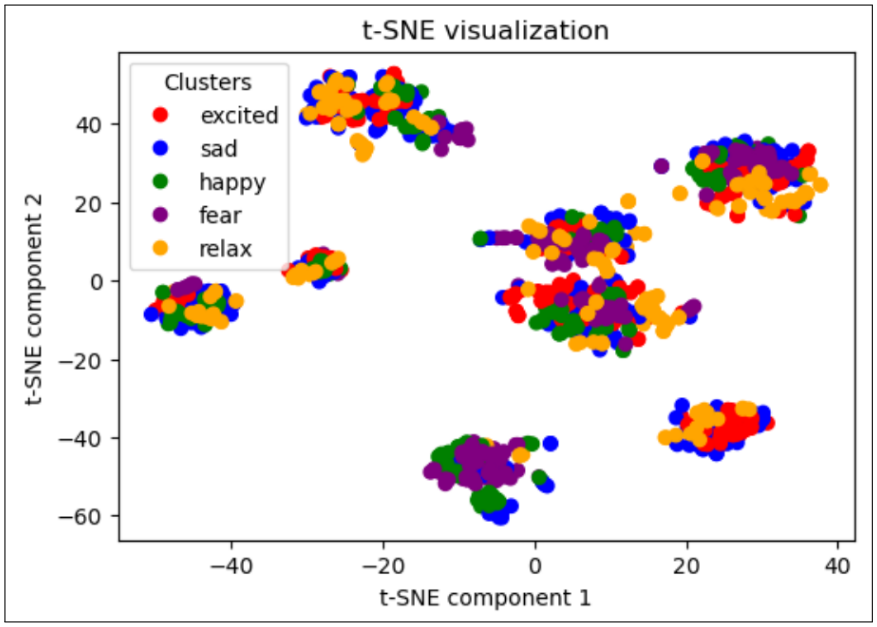
The subsequent step is to solve the generalized Eigenvalue problem for the matrix $\mathbf{S_W^{-1}S_B}$. The ratio of between-class scatter to within-class scatter represents the degree of separability between the classes and thus is the quantity that needs to be maximized. The resolution of this problem results in the Eigenvectors and Eigenvalues where the Eigenvectors with the largest Eigenvalues represent the direction of maximum separability. The number of eigenvectors that can be used in the transformation vector $\mathbf{Z}$ is given by $\min(d, k-1)$ where d represents the number of features and k represents the number of classes. The dimensionality reduction can then be performed by multiplying the input vector $\mathbf{x_n}$ with the transformation matrix $\mathbf{Z}$.

$$\mathbf{X}_{\text{LDA}} = \mathbf{Z}\mathbf{x_n} \tag{4.35}$$

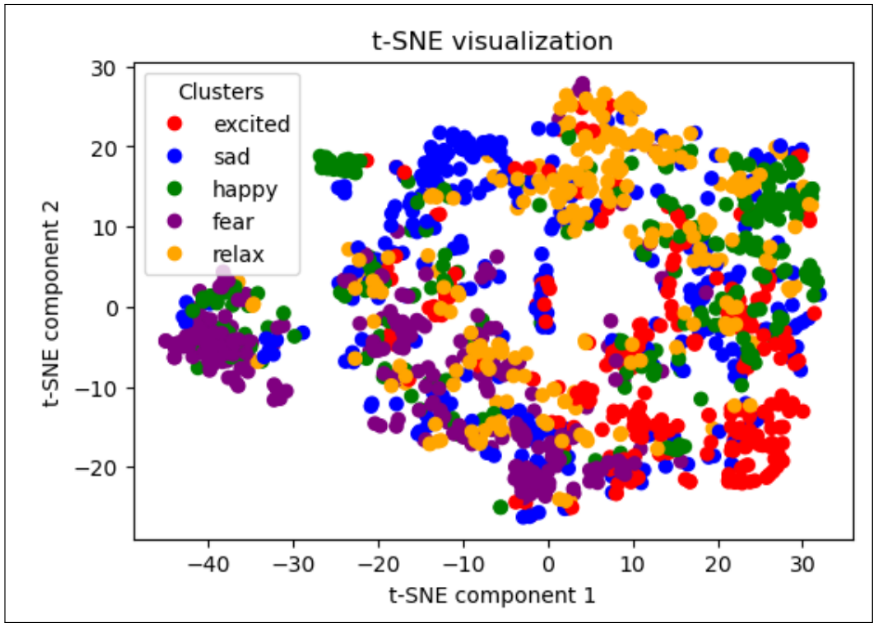
In this instance, LDA was used to reduce the high-dimensional data to four dimensions. This reduced input was then used for the classification stage. The explained variances for all the LDA components for the dataset comprising both time and frequency domain features are described in the table 4.2. Shown in figure 4.15 is the t-SNE plot of the dataset before and after dimensionality reduction using LDA.

4.4.4 T-distributed Stochastic Neighbor Embedding

T-distributed stochastic neighbor embedding (t-SNE) is a dimensionality reduction technique that falls under the category of manifold learning or nonlinear dimensionality reduction. t-SNE is primarily used for obtaining a visualization of high-dimensional data in a low-dimensional 2-D or 3-D space. It is useful for knowing through visualization whether higher dimensional data is separable or not and also to find meaningful structures in the data. It works well in complement to other linear techniques like PCA or LDA, to further refine the number of components needed, helping to balance the complexity versus separability trade-off. t-SNE was developed by Laurens van der Maaten and Geoffrey Hinton as an improvement over the Stochastic Neighbor Embedding (SNE),



(a) Original Data



(b) After Dimensionality reduction

Figure 4.15: t-SNE visualization of the data before and after dimensionality reduction with LDA.

developed by Geoffrey Hinton himself and Sam Roweis(2002) [HR02].

Initially, the high-dimensional Euclidean distance between the data points is converted to a conditional probability distribution that represents similarities. The similarity of datapoint $\mathbf{x_j}$ to $\mathbf{x_i}$ is the conditional probability $p_{j|i}$, that $\mathbf{x_i}$ would choose $\mathbf{x_j}$ as its neighbor, given that neighbors are chosen according to a Gaussian probability distribution centered at $\mathbf{x_i}$. For nearby points the similarity is large. Mathematically $p_{j|i}$ is given by:

$$p_{j|i} = \frac{\exp\left(-\frac{\|\mathbf{x_i}-\mathbf{x_j}\|^2}{2\sigma_i^2}\right)}{\sum_{k\neq i}\exp\left(-\frac{\|\mathbf{x_i}-\mathbf{x_k}\|^2}{2\sigma_i^2}\right)} \tag{4.36}$$

To ensure an easy optimization, the probabilities are symmetrized to ensure an easier optimization problem by forming a joint distribution p_{ij} given by:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \tag{4.37}$$

Now considering the choice of variance σ_i for the Gaussian distribution that is centered at point $\mathbf{x_i}$, there is no one value of σ_i that would be appropriate for all data points in the dataset. The regions with a sparse number of data points would require a higher σ_i, whereas dense regions would require a smaller σ_i. Hence the value of σ_i is chosen using a binary search, such that it yields a distribution P_i with a certain predefined *perplexity*. The *perplexity* can be thought of as a smooth measure of the effective number of neighbors and is given by,

$$\text{Perp}(P_i) = 2^{H(P_i)} \tag{4.38}$$

where $H(P_i)$ is the Shannon entropy of P_i measured in bits.

$$H(P_i) = -\sum_j p_{j|i}\log_2(p_{j|i}) \tag{4.39}$$

To represent the distances between data points in a low-dimensional space, a Student t-distribution with one degree of freedom is used. This is done to overcome the crowding problem that would be an issue if a Gaussian were used. The crowding problem refers to the issue of available space going down as we reduce the number of dimensions. Consider the situation where a set of points are uniformly distributed around a point i in a 10-dimensional space. Now if we were to map all of these points to a lower dimensional space, say 2 dimensions, the area available to accommodate moderately distant points in the 2-D space would not be sufficient to accommodate all the nearby points in the 10-D space. Thus if we would like to accurately map all points, the moderately distanced points would be pushed far away from the point i in the 2-D space. By using the Student t-distribution in the lower-dimensional space we somewhat overcome this problem by having heavy tails allowing for a more spread out distribution of points as compared to

the Gaussian. The joint probability distribution q_{ij} is given by:

$$q_{ij} = \frac{(1 + \|\mathbf{y_i} - \mathbf{y_j}\|^2)^{-1}}{\sum_{k \neq l}(1 + \|\mathbf{y_k} - \mathbf{y_l}\|^2)^{-1}} \tag{4.40}$$

To achieve a closer alignment between the two distributions, an optimization problem is defined by using the Kullback-Leibler Divergence between the joint distributions as a loss function, defined by:

$$C = D_{KL}(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{4.41}$$

To optimize the loss function, the gradient descent algorithm is used to incrementally optimize the positions of the data points in the lower dimensional map to better reflect the joint distribution derived from the higher dimensional space. The computed gradient of the loss function is given by:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y_i} - \mathbf{y_j})(1 + \|\mathbf{y_i} - \mathbf{y_j}\|^2)^{-1} \tag{4.42}$$

The low-dimension positions can then be updated by using the gradient descent update rule given below,

$$\Upsilon(t) = \Upsilon(t-1) + \eta \frac{\delta C}{\delta \Upsilon} + \alpha(t)\left(\Upsilon(t-1) - \Upsilon(t-2)\right) \tag{4.43}$$

To achieve a more optimal convergence of gradient descent to the global minima, two strategies are typically employed. The first strategy is called *early compression* which involves forcing the map points to be close together initially. When all distances are minimal it is easier for data points to move across the map. This is achieved by appending an additional L2-penalty to the cost function, which is proportional to the sum of squared distances of the map points from the origin. The second strategy is called *early exaggeration* and involves taking large steps initially and smaller steps later on. This helps to ensure that clusters form more tightly, allowing for more room for the movement of data points. This is normally implemented by multiplying the p_{ij}'s by a number greater than one.

4.5 Classification

Classification refers to the process of assigning a data point to one of the predefined class labels; in our case, this relates to assigning data points derived from physiological signals of the wristband and assigning them to one of the predefined labels that are *Excited, Fear, Happy, Relax* and *Sad*. This section talks about the various techniques used to perform multi-class classification of the data and also about the strategies used for balancing the classes to facilitate the learning process of the classification algorithms. Mainly classical machine learning algorithms, such as K-Nearest-Neighbors and Support

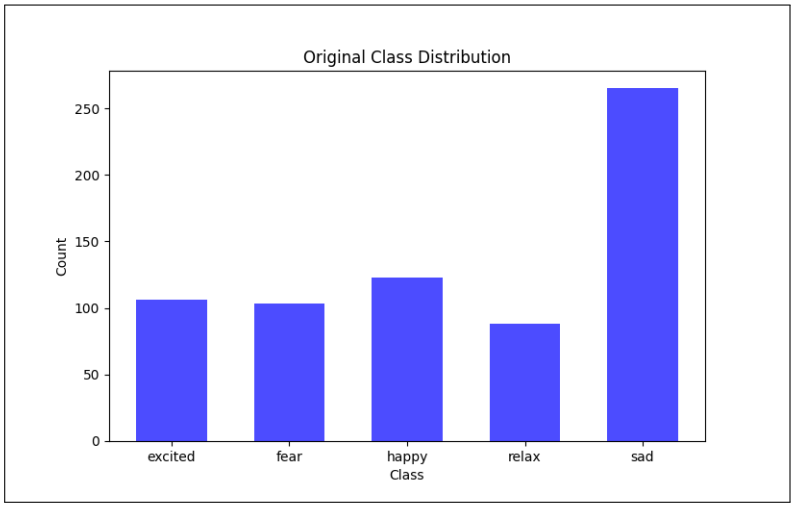


Figure 4.16: Class distribution of the data

Vector Machines, were explored for classification of data. This leaves room for future work related to the exploration of Deep Learning techniques that are currently considered state-of-the-art.

4.5.1 Class Balancing

The feature data, after having undergone dimensionality reduction, is now explored and processed further to be used for classification. The objective is to perform a 5-class classification, with the classes being *Excited*, *Fear*, *Happy*, *Relaxed* and *Sad*. However, before proceeding towards the classification process, the class distribution across the data is evaluated, and certain data augmentation methods are used to artificially introduce class balance. Figure 4.16 shows the tallies for each class in the dataset, and figure 4.17 displays the data distribution considering two arbitrary features. The class counts of the data are shown in table 4.3.

Since a class-imbalanced dataset can negatively influence the model metrics, we use artificial methods to create a balance between the samples of all 5 classes. We make use of the *Synthetic Minority Over-Sampling Technique(SMOTE)*. *SMOTE* is used in this case for synthetically oversampling the minority classes and achieving class balance. It uses a Nearest Neighbours approach to add more samples to the minority class. We select a minority sample from this distribution. After identifying its nearest neighbors,

Class	Excited	Fear	Happy	Relaxed	Sad
Samples	106	103	123	88	265

Table 4.3: Number of samples in each class before using SMOTE

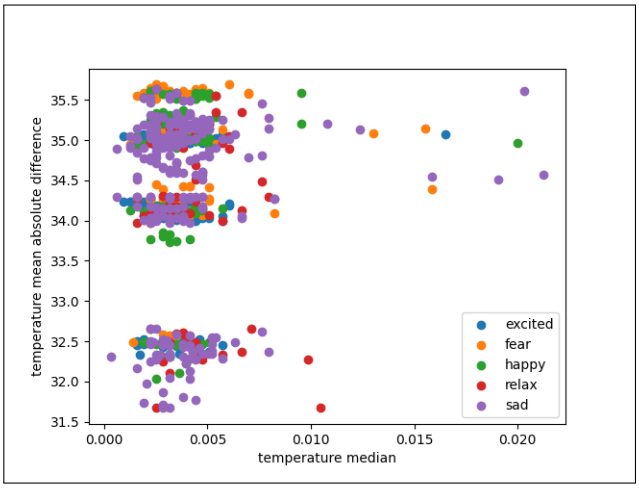


Figure 4.17: Class distribution of the data before SMOTE.

new samples are generated between the selected point and its neighbors. SMOTE allows us to choose the nearest neighbor's value for over-sampling, as well as the sampling strategy to be used. Based on the class count of the original distribution, we will over-sample the 4 minority classes to 80 percent of the sad emotion class. At the same time, we will perform a random under-sampling of the sad emotion class to 120 percent of the minority classes, bringing the class count down to 254. The updated numbers are displayed in table 4.4

From figure 4.18, we can observe that new samples have been generated synthetically to increase the minority class samples, particularly evident through the appearance of a large cluster of *relaxed* emotion samples in the plot. We then verify the class distribution by plotting the frequencies using a barplot to check the class counts. From the figure 4.19, we can confirm that the class distribution is more evenly balanced.

Class	Excited	Fear	Happy	Relaxed	Sad
Samples	212	212	212	212	254

Table 4.4: Number of samples in each class after using SMOTE

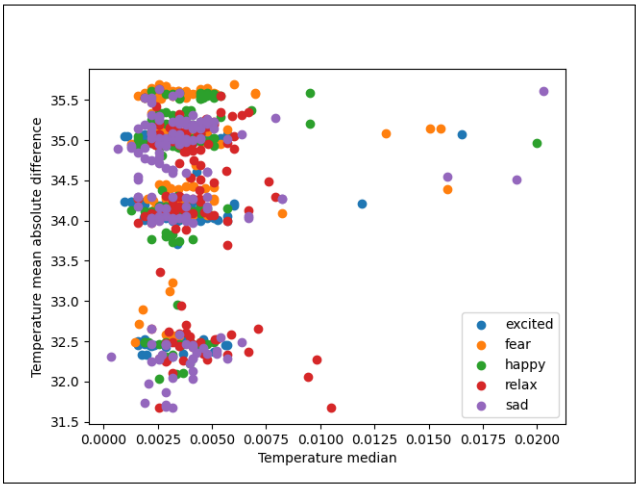


Figure 4.18: Class distribution of the data after SMOTE

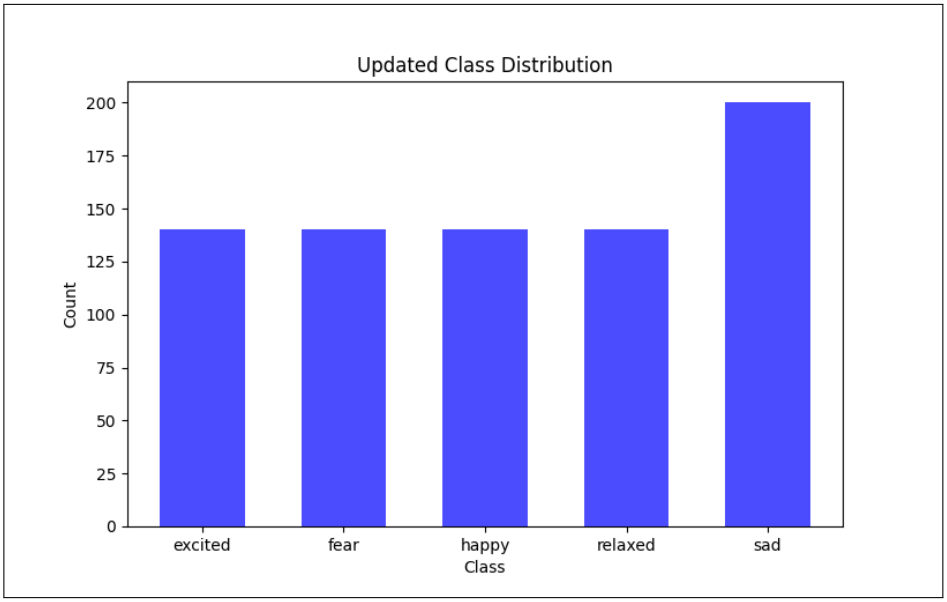


Figure 4.19: Class distribution of the data

4.5.2 Support Vector Machines

Support Vector Machine(SVM) is a supervised learning algorithm that aims to find the separating hyperplane that maximizes the separation of data points to their respective classes. An SVM maximizes the separation using the concept of the *margin*(figure 4.20), which is defined as the smallest distance between the decision boundary and any of the data points. Meanwhile, the data points that serve to define the margin are known as *support vectors*. To derive the SVM classifier a linear model is first considered in the

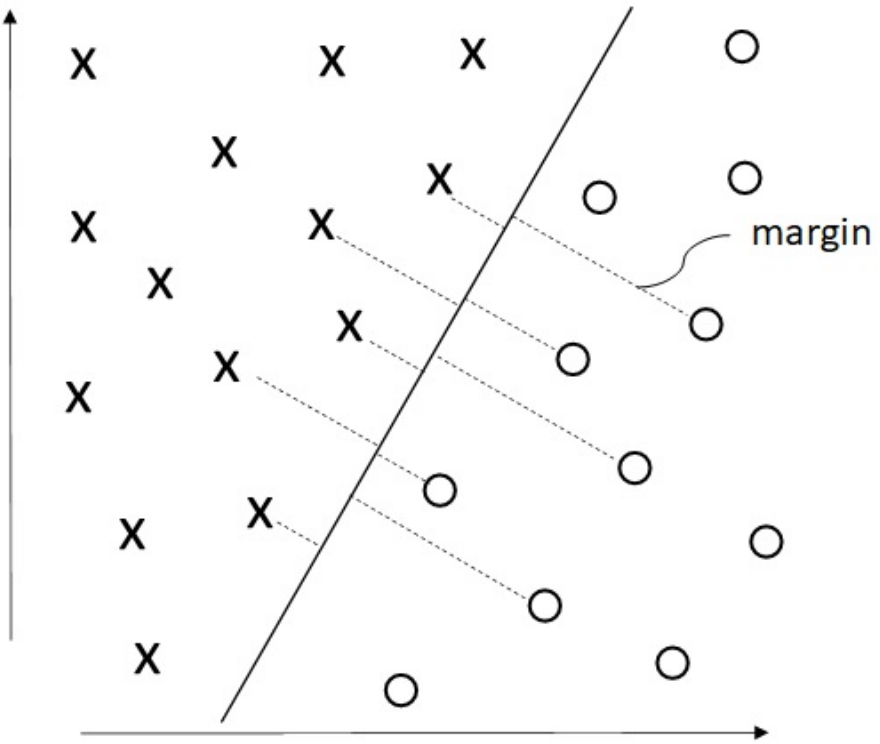


Figure 4.20: Support Vectors and Margins [Mat20].

context of a two-class classification problem.

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b \tag{4.44}$$

Here $\mathbf{x}$ is the input vector, $\mathbf{w}$ is the weight vector, $\phi(\mathbf{x})$ represents a feature transformation of the input and b is the bias term. The vector x is chosen from the data-set comprising N input vectors $x_1, \ldots, x_N$ with t_i corresponding to the target value for the input. The data is assumed to be linearly separable, and since a binary classification problem is considered $t \in \{-1, 1\}$. To find the margin, the perpendicular distance is considered between a point x_n and the separating hyperplane $y(x) = 0$ where $y(x)$ is defined by 4.44. The distance is given by [Bis06]:

$$\frac{t_n y(\mathbf{x}_n)}{\|\mathbf{w}\|} = \frac{t_n(\mathbf{w}^T \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|} \tag{4.45}$$

Now to derive the optimization problem we can take advantage of the fact that the distance between a point and the separating hyperplane(equation 4.45) is invariant to the scaling of $\mathbf{w}$ and b. Hence we can set the constraint mentioned in equation 4.46 for all margin data points, which simplifies the optimization problem.

$$t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) = 1 \tag{4.46}$$

Hence all data points now must satisfy the constraint 4.47

$$t_n(\mathbf{w}^T\phi(\mathbf{x}_n) + b) \geq 1 \tag{4.47}$$

The optimization problem now simplifies to requiring the maximization of $\|\mathbf{w^{-1}}\|$ or rather minimization of $\|\mathbf{w}\|^2$. Hence the optimization is,

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{4.48}$$

Subject to the constraint given by (4.47). This optimization problem can be solved by using *Lagrange Multipliers* $a_n \geq 0$, with one multiplier a_n for each of the constraints. This results in the *Lagrangian function*,

$$L(\mathbf{w}, b, a_n) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n \left\{ t_n(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n) + b) - 1 \right\} \tag{4.49}$$

Now by taking the derivative of $L(\mathbf{w}, b, a_n)$ with respect to $\mathbf{w}$ and b and setting the derivatives to 0, we obtain two conditions.

$$\mathbf{w} = \sum_{n=1}^{N} a_n t_n \boldsymbol{\phi}(\mathbf{x}_n) \tag{4.50}$$

$$0 = \sum_{n=1}^{N} a_n t_n \tag{4.51}$$

By now eliminating $\mathbf{w}$ and b from $L(\mathbf{w}, b, \mathbf{a})$ using these conditions we obtain the *dual representation* of the maximum margin problem in which we maximize,

$$\tilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(\mathbf{x}_n, \mathbf{x}_m) \tag{4.52}$$

Here the kernel function is defined by $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T\phi(\mathbf{x}')$. This can now be solved by using *quadratic programming*. The $y(\mathbf{x})$ can then be defined by,

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, \mathbf{x}_n) + b \tag{4.53}$$

The Kernel Trick

In many cases, the data points are not linearly separable in the current feature space. However, SVM being linear classifiers can only classify linearly separable data. One way of handling this problem is to transform the data into a higher dimensional space, where the data becomes linearly separable. To do so we consider the *kernel function* $k(\mathbf{x}, \mathbf{x}')$,

so far we have only considered a linear kernel that does not transform the data defined by $\phi(\mathbf{x})^T\phi(\mathbf{x}')$. However, several different kernels can transform the data into higher dimensions where the data might be linearly separable. In our case, we consider the *radial basis kernel*, which is given by,

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2\right) \tag{4.54}$$

Here γ is a hyperparameter that controls the spread of the Gaussian [sl24].

Soft-Margin SVM

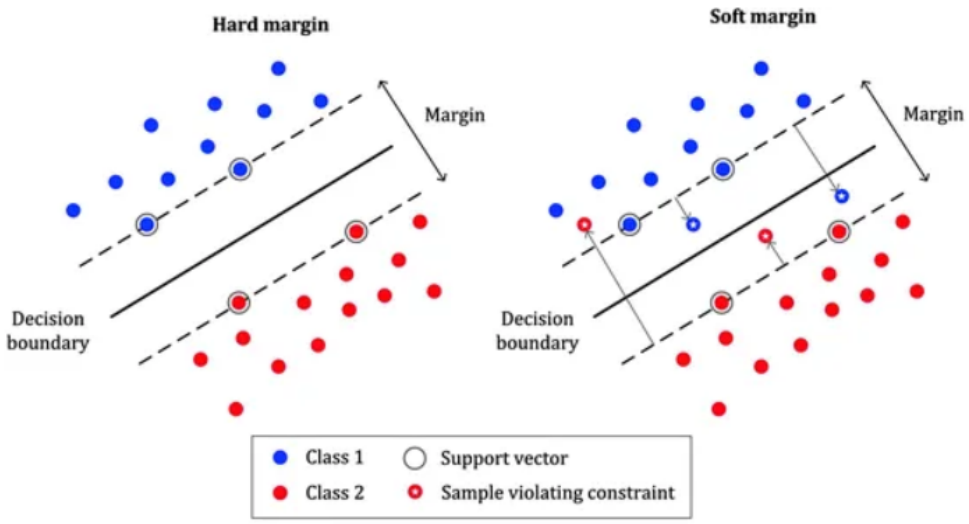


Figure 4.21: Soft-Margin SVM for Handling data that is not linearly separable [MLM19].

There is an assumption thus far that the data is linearly separable, either in its current state or after being transformed to a higher dimensional space. Nevertheless, in the majority of instances, even after transforming the data into a higher-dimensional space, there will be some data points that cannot be accommodated by a hard-margin SVM. In this case, the only option is to relax the condition required for a hard margin, to accommodate a few misclassifications figure 4.21. To do so we introduce a slack variable ξ_n to softly penalize data points that lie within the margin or are classified incorrectly. The minimization problem now becomes,

$$C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|\mathbf{w}\|^2 \tag{4.55}$$

Here $C > 0$ represents a hyper-parameter, which controls the trade-off between the slack variable penalty and the margin. This term can then be optimized similarly to earlier.

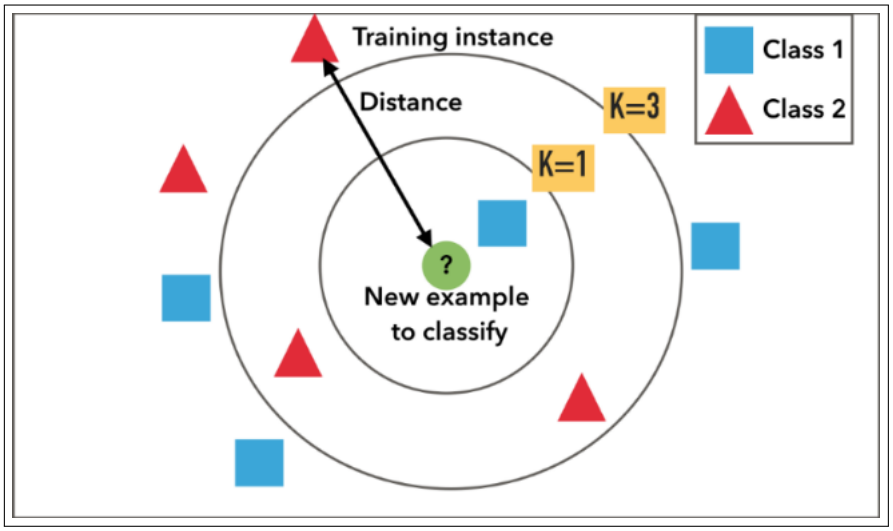


Figure 4.22: KNN visualized, taking reference from [HIB23]

4.5.3 K-Nearest Neighbours

The K-Nearest Neighbors algorithm (KNN) is a very popular supervised classification algorithm. It is a lazy learning algorithm, which implies that every time we intend to make predictions, it utilizes the entire training set to make its computations. We use KNN to classify 5 different emotions: *excited*, *fear*, *happy*, *relax* and *sad*. We then evaluate these predictions using different metrics and determine the effectiveness of the algorithm on our collected data.

KNN considers each query point and identifies the K neighbors close to it. It then uses the majority class from those K neighbors to classify the query point as belonging to that particular class. The value of K is an important hyperparameter for the algorithm. KNN involves the computation of distance between two samples, and there are numerous distance metrics that we can choose from. For implementing KNN on our data, we pick the *Euclidean* distance metric. Figure 4.22 displays this. The expression for the Euclidean distance between points x and y is expressed as follows:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.56}$$

We implement the K-Nearest Neighbors algorithm on the PCA-transformed data. To automate the different stages explained before, including class oversampling, we build a pipeline consisting of the various stages described above and train it on our data to build our model. The pipeline also consists of the hyperparameter tuning described in the next section.

4.6 Evaluation

This chapter outlines the modifications implemented to enhance the precision of the classifiers (SVM and KNN), including hyperparameter tuning and the exclusion of specific feature sets. It also documents the outcomes achieved and identifies avenues for further improvement. It is divided into four parts, with the first two sections discussing the hyperparameter tuning done for both the classifiers(SVM and KNN) and the final two sections concerning the results obtained using SVMs and KNNs.

4.6.1 Hyperparameter tuning for SVMs

In our case, given that we had *five* classes, the LDA yielded *four* components after dimensionality reduction. Furthermore, the SVM with the RBF kernel was considered, allowing two hyperparameters to be tuned to identify the optimal classifier, namely γ and C. Before tuning the hyperparameters, the dataset was divided into *training* and *test* datasets with a 80 : 20 split respectively. To identify the optimal values, *k-fold cross-validation* was used with $k = 5$ to divide the training dataset into five sections, with training in four sections and validation in the fifth section. This process yielded the optimal values of $C = 25$ and $\gamma = 0.01$ for the case where both time and frequency domain features were considered.

4.6.2 Hyperparameter tuning for KNNs

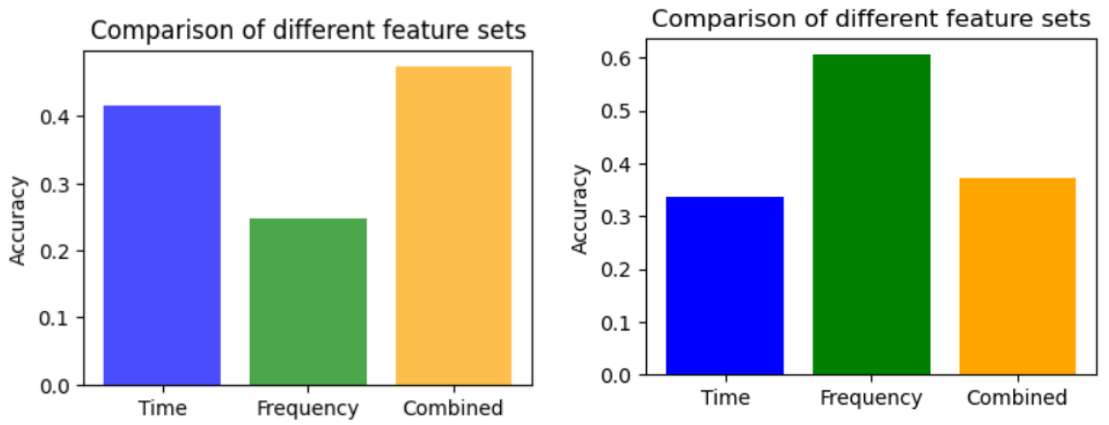
Before implementing KNNs on our data, we split our data into an 80 : 20 ratio, with the larger portion of the data being termed the training set and the remaining portion being termed the test set. To find the ideal value of K, we perform a Grid Search, selecting all possible values of K between 1 and 50 and fitting each model on the training portion of the data. We purposely keep the test set aside as a holdout data-set, to evaluate the accuracy of our final model. For the Grid Search, we plan to evaluate two separate hyper-parameters:

- The number of Principal Components to be used (Range: 1 to 14)
- The ideal K value (Range: 1 to 50)

While specifying the range of K values, we only select odd values in this range. Since we require a clear majority voting while calculating the K nearest neighbors to determine the majority class, we avoid even values in this range. For finding the ideal hyperparameter combination, the Grid Search, using a *5-fold Cross-Validation*, runs a total of $14 * 25 * 5 = 1750$ iterations. The ideal hyperparameter combination we obtain is:

- The number of Principal components to be used: 6
- The ideal K value: 5

Hence, we get our final model, which has 6 Principal Components and K equal to 5.



(a) KNN (b) SVM

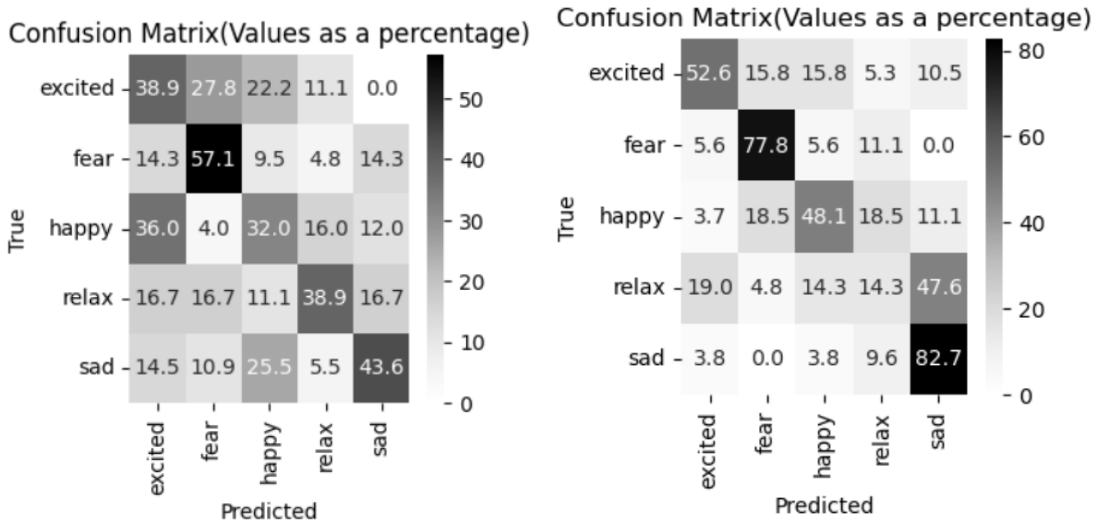
Figure 4.23: Accuracies of KNN and SVM using time, frequency and combined datasets.

4.6.3 Results for SVMs and KNNs

The classification problem comprised five emotional states namely *Sad, Relax, Happy, Fear and Excited*. In a five-class classification problem, the baseline accuracy would be 20%. However, in our case due to the imbalanced class distribution, highly skewed towards the *Sad* class, an alternative approach is required for determining the baseline accuracy. In this case, we employed the *Zero Rate Classifier* approach [Lee21], which simply means finding the accuracy of a classifier that always selects the majority class. In this case, selecting the *Sad* class consistently yielded an accuracy of 37.96% which was used as the baseline.

The accuracy for both the SVM and the KNN using different feature sets is given in fig:4.23. Dimensionality reduction was performed using LDA and PCA for the SVM and KNN classifiers, respectively, for the time-domain and combined feature sets. The frequency-domain feature sets only contain six features and therefore do not require a dimensionality reduction. The highest accuracy for the SVM was achieved by utilizing the frequency domain features, which resulted in an accuracy of 60.58%. Conversely, the KNN achieved the best accuracy on the combined feature set, achieving an accuracy of 45.55%. The low accuracy of the SVM for the combined data set may be attributed to the LDA struggling with the high dimensionality of data in comparison to the number of classes.

The confusion matrices for the SVM and KNN classifiers for the feature set that yielded the optimal accuracy are presented in fig:4.24. An examination of the confusion matrix of the KNN reveals that several data points belonging to the sad class are misclassified as primarily happy. This may be attributed to the issues with the stimuli used to induce sadness, failing to do so. The confusion matrix of the SVM, however, indicates that the SVM has difficulties classifying data points belonging to the relaxed class. This could be attributed to the user's emotional state still being affected by the video played earlier. Consequently, the relaxation time window could probably be extended, or it could also



(a) KNN (b) SVM

Figure 4.24: Confusion Matrices of KNN and SVM.

be the case that the classifier is skewed towards the sad class, which could be inferred from several misclassifications being sad.

5 Conclusion

5.1 Summary

In this project, a functional approach for sensing emotions has been developed in the form of an asynchronous procedure, which means that prerecorded data is used. Its intended specialty is to access as many devices as possible as long as they are all recording the same session simultaneously. Currently, an older wristband and an electroencephalography device are being relied on. As the intended purpose is to sense emotions, music videos have been selected as conducted experiments. Recording data happens based on an automated approach, which handles the recording itself, the specific video selection, and data saving. Only the setup and shutdown procedure, just as a new user logging in or giving individual feedback, are kept manually. This allows for collecting as much data as possible, saved in an automated matter, which later can be filtered by individual subject characteristics for specific tests. At the time of this report, data recorded by the two selected devices is not analyzed in a combined way, but instead individually. The intention behind that is to familiarise with the individual specialties of the used devices and recorded data. Therefore the project group developing the initial task has been split into two subgroups of four for the EEG device and three for the wristband.

The EEG component of this project has demonstrated the critical role that brain activity plays in emotion recognition. The analysis of EEG signals has provided valuable insights into how different emotional states are reflected in brainwave patterns. The successful preprocessing and feature extraction of EEG data, including artifact removal, windowing, and frequency-domain analysis, have laid a solid foundation for accurate emotion classification. Dimensionality reduction techniques such as PCA and LDA have effectively reduced the complexity of the EEG dataset, allowing machine learning classifiers to focus on the most relevant features. The implementation of classifiers like SVM and neural networks has shown promising results. The highest accuracy of 85% was achieved using an Ensemble model combining CNN, LSTM, and their hybrid variants. The SVM classifier, using statistical features and PCA, achieved a test accuracy of 79.5%, however, simpler models like k-NN lagged behind, with a lower accuracy of 62%.

In the case of the wristband subgroup initially, the used sensory is been explored. Based on that an automated pipeline approach has been developed, allowing for automated testing of any following-up functionality. It saves the acquired raw data in a combined easy-to-use format, segments it into parts of interest based on the individual session's characteristics, and checks for integrity. Additionally, it can add more metrics based on the raw data of each segment and also filter individual segments based on

predefined traits. Segmentation value lengths, metrics, or filtering are dynamically adjustable, ensuring optimal adaptability. Subsequently, the resulting segments are used for the feature extraction stage to generate time and frequency-domain features. To reduce the complexity of the combined feature set, techniques like PCA and LDA are utilized, whereas techniques like t-SNE are used to visualize the effectiveness of PCA and LDA based on specific feature sets. Testing the effectiveness of each feature set, it is observed that the frequency-domain feature set is most beneficial for classification using SVM, achieving an accuracy of 60.58 %, whereas the combined feature set provided the highest accuracy for the kNN classifier, yielding an accuracy of 52.55 %. Additionally, it is observed that there are several miss-classifications for the *relax* class, implying that the relaxation period is insufficient for a subject's emotional state to return to a baseline. It could also be explained by factors that cannot be accounted for, such as random thoughts, evoking unexpected emotions. The low overall classification accuracy highlights the difficulty of classifying emotions due to factors that are very difficult to account for. However, there are several possible ways of improving the accuracy, which are explored in the next section.

5.2 Future Work

As mentioned earlier one core idea of this project is to combine multiple devices to improve accuracy regarding emotion sensing. However, thus far this has not been done in a proper way. Consequently, the fusion of data acquired by the already used and in-depth analyzed devices has to take part in the second half of this project. A detailed examination, including a comparison between multiple and single devices, stating exact benefits and whether they are worth the extra effort will be included by then. Additionally, to improve any conducted analysis or comparison as much as possible, a broader dataset has to be collected. Currently, only group members or some friends have been used for testing, which are all males between the ages of 23 to 31. Expanding the used demographic to other ages and also including females allows for more insight, granting a more meaningful analysis. Another way of improving results is to use more or better devices. Thus far an EEG device and a rather old wristband are being used. Upgrading from the E4 wristband to the EmotiBit wearable devices, which for example features a twenty times better resolution in temperature, will improve data quality. Likewise, using a second wristband or an ECG-band for example allows for more data to be analyzed, which further improves results and allows for more meaningful comparisons.

Other ways of improvement lay in the videos used to evoke desired stimuli. Some of the currently utilized videos do not elicit strong emotional reactions in any participant tested so far. Therefore, the used music videos have to be improved or switched to other displayed options. For example, inducing *fear* can be done by relying on a short real-world scenario instead of a music video. Likewise, it is important to disregard the assumption that a specific video, for example, *sad*, only arouses sadness for the entire duration of a video. Some videos even invoke multiple stimuli over their runtime. To tackle this issue timestamps can be used to mark specific video parts as highlights. Highlighted parts

represent mainly the desired stimuli induced in a strong form. However, this further complicates segmentation but also reduces interference of other stimuli, strengthening the robustness of the developed approach. Additionally, more emotion classes can be introduced. Currently, only five different stimuli are utilized. Expanding them by more nuanced emotional states, like *disgust*, *surprise*, or *boredom*, can provide a more comprehensive understanding of the emotional spectrum and improve the applicability of the presented approach.

Last but not least, regarding data acquisition the featured GUI can be improved further. Except for a cleanup of the underlying architecture, a statistics feature can be added. This can be used to visualize general stats, such as age, gender, durations, dates, ratings, and so on, for the entire dataset, subsets, or even individual users.

Furthermore, data processing can also be improved in multiple ways. Currently, the presented pipeline can be copied into a user folder and executed from there. However, this only yields results for a single user per execution. To further improve automation, the pipeline can be fed with another file from the GUI, which states a specific set of multiple users. This set can be based on the age or gender of certain subjects and thus rely on the internal database of the GUI. Additionally, the pipeline can be expanded to support more metrics such as *heart rate variability*, and already implemented metrics, such as *inter beat interval*, can be improved. Furthermore, feature extraction can be integrated into the pipeline as well.

The feature extraction functionality can also be enhanced. One option is to expand it to support even more extraction options such as additional frequency features based on entropy of the utilized data. Another option is to expand its applicability to all metrics of a segment, like *HR* instead of just *BVP*. Final ways of improvement can be done in the classification. Apart from improving the implementation of already existing variations, adding more advanced techniques can further improve results. For this, ensemble methods or deep learning architectures such as transformers and graph neural networks could be utilized. Specifically, these advanced models may be better suitable to capture the temporal and spatial dynamics of EEG signals.

Bibliography

[AC16] J. Atkinson and D. Campos. Improving bci-based emotion recognition by combining eeg feature selection and kernel classifiers. *Expert Syst. Appl.*, 47:35–41, 2016.

[AMMK16] M. Ali, A. H. Mosa, F. Al Machot, and K. Kyamakya. Eeg-based emotion recognition approach for e-healthcare applications. *Proc. 8th Int. Conf. Ubiquitous Future Netw.*, 8:946–950, 2016.

[BFF+20] Marília Barandas, Duarte Folgado, Letícia Fernandes, Sara Santos, Mariana Abreu, Patrícia Bota, Hui Liu, Tanja Schultz, and Hugo Gamboa. Tsfel: Time series feature extraction library. *SoftwareX*, 11:100456, 2020.

[Bis06] Christopher M. Bishop. Pattern recognition and machine learning. `https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf`, 2006. Chapter 7-Sparse kernel methods| Visited 2024-10-07.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[BWJR13] Dr. Jason J Braithwaite, Dr. Derrick G Watson, Robert Jones, and Mickey Rowe. A guide for analysing electrodermal activity (eda) & skin conductance responses (scrs) for psychological experiments. `https://www.biopac.com/wp-content/uploads/EDA-SCR-Analysis.pdf`, 2013. complex description of GSR components | Visited 2024-09-18.

[CBHK02] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[CCC+03] L.J. Caoa, K.S. Chua, W.K. Chong, H.P. Lee, and Q.M. Gu. A comparison of pca, kpca and ica for dimensionality reduction in support vector machine. *Neuro Computing*, 55:331–336, 2003.

[CL14] Bingyu Wang Cheng Li. Fishers lda. `https://www.khoury.northeastern.edu/home/vip/teach/MLcourse/5_features_dimensions/lecture_notes/LDA/LDA.pdf`, 2014. short description of LDA| Visited 2024-10-03.

[Con21] ConnectedFutureLabs. Emotibit wearable. `https://www.emotibit.com/`, 2021. Website for Emotibit device | Visited 2024-09-22.

[Cor24a] Mozilla Corporation. Css mdn documentation. `https://developer.mozilla.org/en-US/docs/Web/CSS`, 2024. Documentation of CSS 2024-10-11.

[Cor24b] Mozilla Corporation. Html mdn documentation. `https://developer.mozilla.org/en-US/docs/Web/HTML`, 2024. Documentation of HTML 2024-10-11.

[Cor24c] Mozilla Corporation. Javascript mdn documentation. `https://developer.mozilla.org/en-US/docs/Web/JavaScript`, 2024. Documentation of JS 2024-10-11.

[Cor24d] Oracle Corporation. Mysql oracle documentation. `https://www.mysql.com/`, 2024. Documentation of MySQL 2024-10-11.

[CST00] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[DP20] Vikrant Doma and Matin Pirouz. A comparative analysis of machine learning methods for emotion recognition using eeg and peripheral physiological signals. *Journal of Big Data*, 7:18, 2020.

[DZL13] Ranran Duan, Junxia Zhu, and Bao-Liang Lu. Differential entropy feature for eeg-based emotion classification. In *6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 81–84. IEEE, 2013.

[Edi17] BD Editors. Systole. `https://biologydictionary.net/systole/`, 2017. simple visualization of systole | Visited 2024-09-25.

[Emo23] Emotiv. Understanding the 10-20 system of eeg electrode placement. =https://www.emotiv.com/blogs/how-to/understanding-the-10-20-system-of-eeg-electrode-placement, 2023. Accessed: 10.10.2024.

[Emp14] Empatica. Datasheet of empatica-e4 wristband. `https://box.empatica.com/documentation/20141119_E4_TechSpecs.pdf`, 2014. Datasheet for early E4 Device | Visited 2024-09-17.

[Emp18] Empatica. Empatica-e4 wristband. `https://www.empatica.com/en-eu/research/e4/`, 2018. Empatica Website for E4 Device | Visited 2024-09-17.

[Emp20a] Empatica. Discription of empatica-e4 bvp sensor. `https://support.empatica.com/hc/en-us/articles/204954639-Utilizing-the-PPG-BVP-signal`, 2020. short discription of BVP of E4 Device | Visited 2024-09-17.

[Emp20b] Empatica. E4 data - bvp expected signal. `https://support.empatica.com/hc/en-us/articles/360029719792-E4-data-BVP-expected-signal`, 2020. short discription of BVP signals | Visited 2024-09-17.

[Emp21] Empatica. What should i know to use eda data in my experiment? `https://support.empatica.com/hc/en-us/articles/203621955-What-should-I-know-to-use-EDA-data-in-my-experiment`, 2021. simple description of | Visited 2024-09-18.

[Emp22] Empatica. Empatica embraceplus wristband. `https://www.empatica.com/en-eu/embraceplus/`, 2022. Empatica Website for EmbracePlus | Visited 2024-09-22.

[Emp24] Empatica. E4 connect. `https://e4.empatica.com/connect/login.php`, 2024. Empatica connect website | Visited 2024-09-17.

[Fou24] Python Software Foundation. Python organization. `https://www.python.org/`, 2024. Documentation of Python 2024-10-11.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GSK+17] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutnik, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017.

[GVl+16] Alberto Greco, Gaetano Valenza, Antonio lanatà, Enzo Scilingo, and Luca Citi. cvxeda: A convex optimization approach to electrodermal activity processing. *IEEE Transactions on Biomedical Engineering*, 2016:797–804, 04 2016.

[GWB+03] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer. Knn model-based approach in classification. In R. Meersman, Z. Tari, and D.C. Schmidt, editors, *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 986–996. Springer, Berlin, Heidelberg, 2003.

[HIB23] Aouidad Hichem Idris and Abdelhamid BOUHELAL. Machine learning-based short-term solar power forecasting: A comparison between regression and classification approaches using extensive australian dataset, 12 2023.

[HR02] Geoffrey Hinton and Sam Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15 (NeurIPS)*, pages 857–864. MIT Press, 2002.

[ICJ02] Ifeachor, Emmanuel C., and Barrie W. Jervis. Digital signal processing: a practical approach. *Pearson Education India*, 2002.

[IDT+23] Abhishek Iyer, Srimit Sritik Das, Reva Teotia, Shishir Maheshwari, and Rishi Raj Sharma. Cnn and lstm based ensemble learning for human emotion recognition using eeg recordings. *Multimedia Tools and Applications*, 82(4):4883–4896, 2023.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[JC16] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[KMS+12] Sander Koelstra, Christian Mühl, Mohammad Soleymani, Ashkan Yazdani, and Jong seok Lee. Deapdataset. `https://www.eecs.qmul.ac.uk/mmv/datasets/deap/index.html`, 2012. Deap dataset project website | Visited 2024-09-17.

[KP98] Ron Kohavi and Foster Provost. Glossary of terms. *Machine Learning*, 30(2-3):271–274, 1998.

[Lee21] Aaron Lee. Choosing a baseline accuracy for a classification model. `https://towardsdatascience.com/calculating-a-baseline-accuracy-for-a-classification-model-a4b342ceb88f`, 2021. Choosing a Baseline Accuracy for a Classification Model| Visited 2024-10-10.

[LFF12] Dominik Johannes Leiner, Andreas Fahr, and Hannah Früh. Eda positive change: A simple algorithm for electrodermal activity to measure general audience arousal during media exposure. `https://www.researchgate.net/figure/EDA-Response-SCR-Pattern-and-Relevant-Parameters_fig1_233883743`, 2012. simple description of | Visited 2024-09-19.

[LSMG09] Christopher M. Laine, Kevin M. Spitler, Clayton P. Mosher, and Katalin M. Gothard. Behavioral triggers of skin conductance responses and their neural correlates in the primate amygdala. `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2695635/`, 2009. simple description of | Visited 2024-09-20.

[LTX+16] S. Liu, J. Tong, M. Xu, J. Yang, H. Qi, and D. Ming. Improve the generalization of emotional classifiers across time by using training samples from different days. *Proc. Annu. Int. Conf. IEEE Eng. Med. Biology Soc.*, 8:841–844, 2016.

[Lux22] Ulrike Von Luxburg. Universitat tubingen. https://www.tml.cs.uni-tuebingen.de/teaching/2022_statistical_learning/downloads_free/vorlesung_main.pdf, 2022. Documentation of MySQL 2022-21-07.

[LYH+22] Shuang Liang, Mingbo Yin, Yecheng Huang, Xiubin Dai, and Qiong Wang. Nuclear norm regularized deep neural network for eeg-based emotion recognition. *Emotion Science*, page Front. Psychol. 13:924793. doi: 10.3389/fpsyg.2022.924793, 2022.

[Mal96] Marek Malik. Heart rate variability. https://www.escardio.org/static-file/Escardio/Guidelines/Scientific-Statements/guidelines-Heart-Rate-Variability-FT-1996.pdf, 1996. detailed description of HRV, posted in euroopean heart journal 1996 vol 7| Visited 2024-09-19.

[Mat20] Tsuyoshi Matsuzak. Mathematical introduction to svm and kernel methods. https://tsmatz.wordpress.com/2020/06/01/svm-and-kernel-functions-mathematics/, 2020. Mathematical Introduction to SVM and Kernel Methods| Visited 2024-10-07.

[Min23] Mindtec Store. Emotiv epoc flex saline sensor set, 2023.

[MLM19] MLMath.io. Math behind svm(support vector machine). https://ankitnitjsr13.medium.com/math-behind-svm-support-vector-machine-864e58977fdb, 2019. Math behind SVM(Support Vector Machine) 2024-10-09.

[MMG24] Medizinische-Messtechnik-GmbH. Photo-plethysmographie (ppg). https://www.medis.company/de/methoden/photo-plethysmographie, 2024. short discription of PPG measurements | Visited 2024-09-17.

[Mur11] M. Murugappan. Electromyogram signal based human emotion classification using knn and lda. In *2011 IEEE International Conference on System Engineering and Technology*, pages 106–110, Shah Alam, Malaysia, 2011.

[Nef24] Megan Anna Neff. The arousal-valence model. https://neurodivergentinsights.com/blog/arousal-valence-model#:~:text=What%20is%20the%20Arousal%2DValence,emotion%20is%20positive%20or%20negative., 2024. short description of vam | Visited 2024-09-19.

[oC21] Moderators of Cleveland. Heart rate variability (hrv). https://my.clevelandclinic.org/health/symptoms/21773-heart-rate-variability-hrv, 2021. simple description of HRV | Visited 2024-09-19.

[Pal24] Pallets. Flask documentation. `https://flask.palletsprojects.com/en/3.0.x/`, 2024. Documentation of Flask 2024-10-11.

[PP08] Cheong Hee Park and Haesun Park. A comparison of generalized linear discriminant analysis algorithms. *Pattern Recognition*, 41(4):1381–1395, 2008.

[PRP08] Jonathan Posner, James A. Russell, and Bradley S. Peterson. The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2367156/`, 2008. detailed description of modeling emotions | Visited 2024-09-19.

[QCY+24] Sen Qiu, Yongtao Chen, Yulin Yang, Pengfei Wang, Zhelong Wang, Hongyu Zhao, Yuntong Kang, and Ruicheng Nie. A review on semi-supervised learning for eeg-based emotion recognition. *Information Fusion*, 104:102–107, 2024.

[SDJ22] J. Smith, A. Doe, and B. Johnson. Skcv: Stratified k-fold cross-validation on ml classifiers for predicting cervical cancer. *Frontiers in Nanotechnology*, 4:972421, 2022.

[SFJI20] M. N. A. H. Sha'abani, N. Fuad, N. Jamal, and M. F. Ismail. knn and svm classification for eeg: A review. In A. N. Kasruddin Nasir et al., editors, *InECCE2019*, volume 632 of *Lecture Notes in Electrical Engineering*, pages 467–474. Springer, Singapore, 2020.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[SL09] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437, 2009.

[sl24] scikit learn. Rbf svm parameters. `https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html`, 2024. Documentation of scikit-learn 2024-10-09.

[Sub07] Abdullah Subasi. Eeg signal classification using wavelet feature extraction and a mixture of expert model. *Expert Systems with Applications*, 32(4):1084–1093, 2007.

[Sub19] Abdullah Subasi. *Signal Processing and Machine Learning for Brain-Machine Interfaces*. Academic Press, 2019.

[TM20] Jonas Teuwen and Nikolay Moriakov. Chapter 20 - convolutional neural networks. In *Handbook of Medical Image Computing and Computer-Assisted Intervention*, The Elsevier and MICCAI Society Book Series, pages 481–501. Academic Press, 2020.

[WMdW+20] Nikolas S. Williams, Genevieve M. McArthur, Bianca de Wit, George Ibrahim, and Nicholas A. Badcock. A validation of emotiv epoc flex saline for eeg and erp research. *PeerJ*, 8:e9713, 2020.

[Won15] Tzu-Tsung Wong. Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Expert Systems with Applications*, 42:1796–1808, 2015.

[WW+13] Fatimah Wan, Noraimi Wan, et al. Feature extraction based on windowing and statistical moments for emotion classification. *International Journal of Computer Theory and Engineering*, 5(5):845, 2013.

[ZL15] Wei-Long Zheng and Bao-Liang Lu. Investigating critical frequency bands and channels for eeg-based emotion recognition with deep neural networks. *IEEE Transactions on Autonomous Mental Development*, 7(3):162–175, 2015.

[ZW20] Huiping Jiang Zequn Wang, Rui Jiao. Emotion recognition using wt-svm in human-computer interaction. *Journal of New Media*, 2(3):121–130, 2020.

[ZWYG18] Bobo Zha, Zhu Wan, Zhiwen Yu, and Bin Guo. Emotionsense: Emotion recognition based on wearable wristband. In *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pages 346–355, Guangzhou, China, December 2018. IEEE.

[ZZPL14] Wei Zheng, Jiaojiao Zhu, Yong Peng, and Bao-Liang Lu. Eeg-based emotion classification using deep belief networks. In *2014 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2014.

[ZZY+24] Shuping Zhao, Bob Zhang, Jian Yang, Jianhang Zhou, and Yong Xu. Linear discriminant analysis. *Nature Reviews Methods Primers*, 4:70, 2024.

Appendix A

Appendix

A.1 DEAP dataset video clustering

In order to cluster the videos obtained from the DEAP data set based on emotional metrics, first K-Means clustering is applied to group the videos according to their average valence and arousal scores. This approach organizes the videos into four distinct emotional categories — Excited, Fear, Happy and Sad. These clusters help in identifying representative emotional patterns among the videos by grouping similar ones together. The most prominent or representative samples from each cluster are then selected by sorting the videos based on valence and arousal values, ensuring that the top 15 most fitting examples from each emotional group are chosen for further analysis.

Following the clustering, Support Vector Machine (SVM) is deployed to distinguish between these emotional categories. The classifier is trained using the selected representative samples, relying on the valence and arousal features to predict emotional labels. This enables a clear boundary to be drawn between different emotional categories, making it easier to classify videos in terms of their emotional content. The SVM's decision boundaries are then visualized as shown in Fig. A.1, allowing for a clear understanding of how well the classifier separates emotional clusters in the valence-arousal space, and highlighting the effectiveness of the model in distinguishing between different emotional categories.

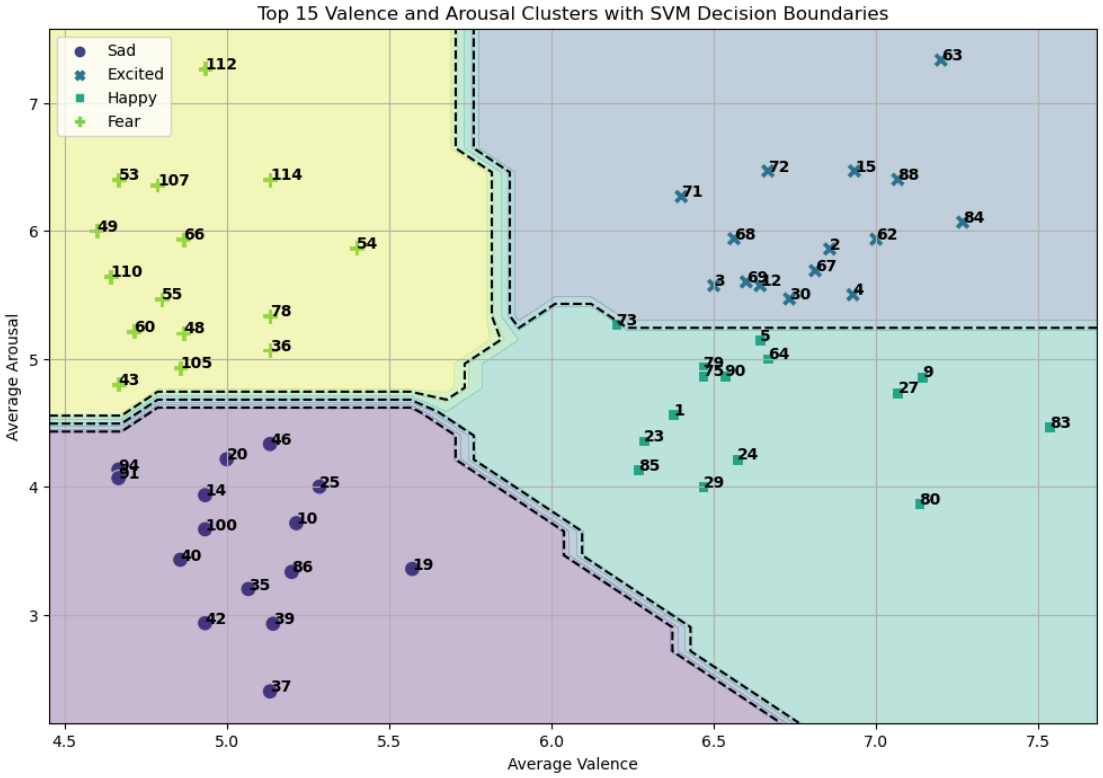


Figure A.1: Clusters in the valence-arousal model of 15 each for the four stimuli used in this project. Videos are represented based on their ID from the DEAP dataset.

Affirmation

Hereby we declare that this work is entirely our own and each author is responsible for his individually written parts. All used tools are stated and all used sources, whether they have been used directly or indirectly, are cited. This work, be it partly or entirely, has not been subject to any grading committee yet.

Paderborn, _____ _____
(Date) (Signature, Abdur Rafay)

Paderborn, _____ _____
(Date) (Signature, Omar Anber)

Paderborn, _____ _____
(Date) (Signature, Omar Ubaid)

Paderborn, _____ _____
(Date) (Signature, Surya Mani Kumar Jakka)

Paderborn, _____ _____
(Date) (Signature, Jatin Yerawadekar)

Paderborn, _____ _____
(Date) (Signature, Kushal Rao)

Paderborn, _____ _____
(Date) (Signature, Michael Siegmund)