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 - Winter Semester 2024/2025

by

Abdur Rafay, Omar Ubaid, Kishan Nadageri, Aditya Keny
Jatin Yerawadekar, Kushal Rao, Michael Siegmund

submitted to:
Prof. Peter Schreier

supervised by:
Maurice Kuschel

Paderborn, March 31, 2025

Abstract.

Emotion recognition is a key area of research in several fields. Improvements in emotion recognition can have far-reaching implications in various applications such as mental health monitoring, human-computer interface (HCI) and others. In this project, we aim to tackle the problem of emotion recognition using data from two widely used types of devices, namely a electroencephalography device (EEG) and a wristband. For data collection, both devices are connected to a graphical user interface (GUI). The GUI streams video stimuli to evoke an emotion in the participant, whose data is recorded by the two devices. The GUI then stores the session data as well as the data collected by the two devices in CSV files for further processing. For this project, five emotions are considered, namely *excited*, *fear*, *happy*, *relaxed* and *sad*.
Regarding the offline approach, the data is passed through a pipeline, which also extracts additional metrics. Here, it is segmented and filtered to remove bad segments. These filtered segments are used to train classifiers to classify them into the five emotions listed. Several machine learning techniques (SVM and KNN) and deep learning architectures (DNN and 1-D CNN) are used for the classification task. In addition, a combined classification using data from both devices is also experimented with. The highest classification accuracy, using the random sampling evaluation approach, is achieved by the 1-D CNN-based combined classifier, which achieved an accuracy of 75.52%.
To make the classifiers more interpretable, several Explainable AI (XAI) techniques such as SHAP and GradCAM are used. These techniques provided numerous insights into the key features that help determine the current emotional state of the brain. Finally, a live mode is developed as an extension of the GUI to provide additional real-time classification capabilities. The live mode provides visualizations of the various metrics collected from the user and also allows for near real-time emotion recognition by interfacing with the classification models.

Contents

1 Introduction

Emotion recognition and mental state sensing belong to a wide field of affective computing, which has rapidly evolved over the last decades. Especially, applications ranging from human-computer interaction to mental health monitoring enhancing medical treatments are of interest. This project focuses mainly on sensing a subject's emotional state. Therefore, multiple widely available devices are used for this purpose. Namely, an electroencephalography device (EEG) is used to allow insight into human brain activity undergoing certain scenarios. Additionally, a wristband is employed to capture more general signals from a subject's body, including heart activity, sweat activity and skin temperature.

This project is built on the results of last semester's equally named project [SRR+24], which focused on emotion recognition using individual devices. The specific devices used are the *Emotiv EPOC Flex* EEG device and the *Empatica E4* wristband. In this project's iteration, both devices are also used in a fusion approach to further enhance prediction accuracy. Therefore, an automated system for conducting experiments, including a GUI, has been developed and is further improved and enhanced in the current project. For example, a new data structure allowing for the easy addition of new devices and a statistics page showing core characteristics of the captured dataset are added to the GUI. Experiments conducted are music videos taken from the *DEAP* dataset, which is a scientific database mainly developed for the core goal of this project, sensing a subject's emotional state [KMS+12]. Details of the approach used and insight into the currently captured dataset are given in chapter 2.

Chapter 3 presents an automated approach of obtaining data for the used machine learning algorithms. The basic pipeline used has already been shown in the last iteration of the project and is motivated by approaches mentioned in [CAB+23] and [CPSS21]. In this report, it is extended to support multiple users and multiple devices saved in the new data structure. Additionally, new metrics such as *heart rate variability* (HRV) are added on top of improved metrics like *inter beat interval* (IBI). Furthermore, a feature extraction phase is added by combining all standalone scripts from the last report. Using both mentioned devices and including extracted metrics, allows for more than

1000 feature results per segment. Used segmentation lengths, metrics or the feature extraction stage itself are individually tuneable.

Chapter 4 presents new machine learning algorithms used for the classification of the pipeline's provided segments or feature results. Here, two new approaches, namely *deep neural networks* (DNN) and *convolutional neural networks* (CNN), are employed for all specified devices. They are motivated by papers such as [KB17a] or [HG23], which apply modifications on neural network hyper-parameters such as ADAM and ReLU respectively, as well as [SDR+19] that uses 1-D CNN based architectures with sensor-based late fusion. The new approaches are compared to already implemented algorithms, namely *support vector machines* and *k nearest neighbours*, using individual devices but also a fusion of both devices. Multiple test methodologies, including *random test splitting* and *leaving one subject out*, are also compared and discussed.

Furthermore, to understand the used classifiers' decision-making, chapter 5 presents insights into the black box behaviour of neural networks. Here, multiple techniques, namely *shapley additive explanations* (SHAP) and *gradient-weighted class activation mapping* (Grad-CAM), are shown, motivated by scientific works as [A+20]. Another new functionality in this project is the newly added live mode in chapter 6. Unlike the offline methodology of recording videos and processing the results at a later point in time, the live mode tries to give accurate feedback regarding sensed emotions in real-time. It reads data streams from both devices and uses a pre-trained DNN model to provide real-time outputs with a delay of mere seconds. This additional testing mode can also display the output of used devices in real-time.

Finally, chapter 7 summarizes the key findings of the current project iteration. Here, the main findings are recapitulated and limitations are demonstrated. Furthermore, improvements are listed and briefly discussed in an extra section assigned for future work.

2 Fundamentals

This chapter provides an overview about how data is collected to be provided for later chapters. First, in section 2.1, the devices used are presented. Then section 2.2 presents the graphical user interface used to automate the data collection. Furthermore, section 2.3 describes the structure of the dataset folder and provides insight into the collected data. Here, the output of a data acquisition tool is also included for further analysis.

2.1 Devices Used

To collect data, adequate to sense an emotional state, multiple devices are used in this project. Specifically the *Emotiv EPOC Flex* EEG headset and the *Empatica E4* wristband, which also have been used in the last semester's project, are employed in this iteration again. Both devices are shown in figure 2.1 and used simultaneously to collect as much data from various sensors as possible.

The electroencephalography headset, which will be called EEG device in this report, is made of a soft cap with predefined holes allowing to deploy the electrodes according to different standards, and two positions to secure the controller of the device. The

(a) The Emotiv Epoc Flex EEG Headset used [Min23].

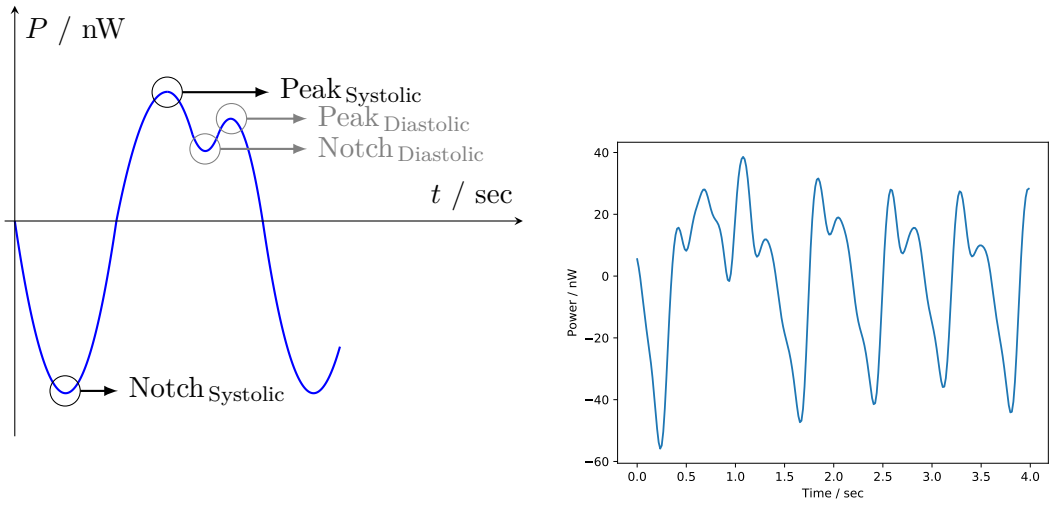
(b) The Backside of the used Empatica E4 wristband showing all relevant sensors.

Figure 2.1: The devices used in this report. Both have already been employed in the last iteration [SRR+24].

Emotiv Pro software is used to set up and connect to the device, which provides a data stream for the later described GUI by invoking a script using the *Cortex API*. The device offers up to 32 EEG channels and two reference electrodes, which are split up into 16 EEG channels and a reference node on each side of the scalp. For this project electrodes are placed according to the international 10-20 system and out of the available 32 EEG channels 29 are used [Emo23]. All of them work with a sampling rate of 128 $^{\text{Sa}}/_{\text{s}}$, collecting 3712 data points in the µV range per second [Min23].

In addition to the EEG headset, the E4 wristband has been used to record supplementary modalities. It is comparable to an old smartwatch and measures various quantities. Connecting to the wristband happens using the *Streaming Server*, which connects to the wristband itself. A *Python* script can connect to the Streaming Server using the *socket* package, allowing it to receive the desired data streams. Once the provided wristband has been added to the student account, it was usable until the 14$^{\text{th}}$ of February because Empatica discontinued support for their E4 model. This includes their servers for the API request necessary to check whether an account logged in to their services is allowed to use a specific wristband. Hence, all Empatica E4 wristbands are currently not usable.

However, while the wristband has been usable, it provided data from three of its sensors. The first sensor is the photoplethysmography sensor, called PPG or BVP sensor, which provides data regarding a subject's heart activity. It works by sending out light



(a) BVP model with marked points of interest. Directly taken from [SRR+24].

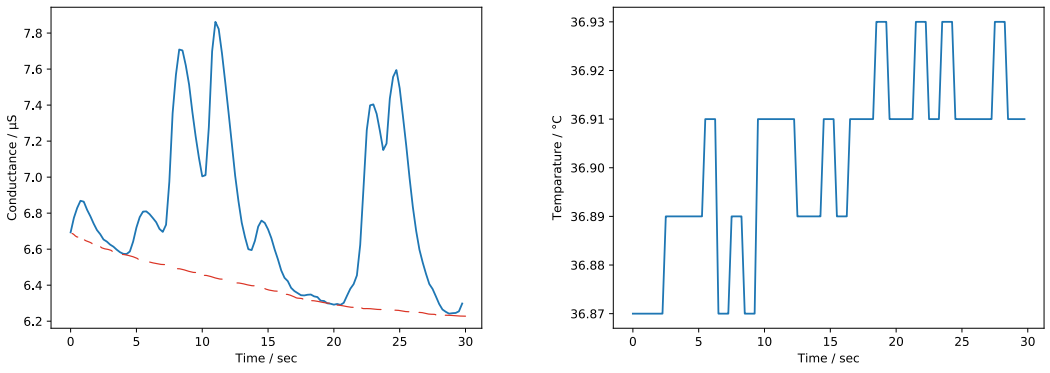
(b) Measured BVP signal used later to illustrate the new IBI algorithm.

Figure 2.2: Comparing ideal and real BVP signals. Ideal model is derived by [Edi17][Emp20].

into a subject's arm using diodes and then measuring the reflected power of said light. Depending on the amount of oxygenated blood present at the specific moment, the reflected energy varies as oxygenated blood absorbs the wavelength of the built-in diode. This happens with a sample rate of 64 $^{\text{Sa}}/_{\text{s}}$ and yields values in the range of $\pm 100\,\text{nW}$. A typical output curve is shown in figure 2.2.

The second sensor is the electrodermal activity sensor, called GSR or EDA sensor, which provides data regarding a subject's sweat activity. It works by applying a current at the electrodes on the band, which yields a voltage dependent on the user's skin conductivity. Depending on the amount of sweat, the skin conductivity, and thus the measured voltage, changes. This happens with a sample rate of 4 $^{\text{Sa}}/_{\text{s}}$ and yields values in the range of a few µS. Furthermore, the EDA signal can be split into multiple components, namely the tonic component and the phasic component [BWJR13]. Figure 2.3a shows an example EDA signal. Here, the tonic component, which consists mainly of low-frequency components of the EDA signal, is hand-drawn as a guess. Consequently, the remaining peaks are the phasic component of the EDA signal, which can appear due to emotional stimuli [LSMG09].

The last sensor is the temperature sensor, called TMP sensor, which provides data regarding a subject's skin temperature at the wrist. The built-in sensor of the E4 wristband is a thermopile sensor that converts infrared light into a voltage, which can basically be measured like the EDA sensor's voltage. This happens with a sample rate of 4 $^{\text{Sa}}/_{\text{s}}$ and yields values with a resolution of 0.02 °C, shown in figure 2.3b.



(a) An example EDA signal directly taken from [SRR+24].

(b) An example TMP signal directly taken from [SRR+24].

Figure 2.3: Example EDA and TMP signals taken from the last semesters report. The EDA signal contains a guess regarding the tonic component drawn in red.

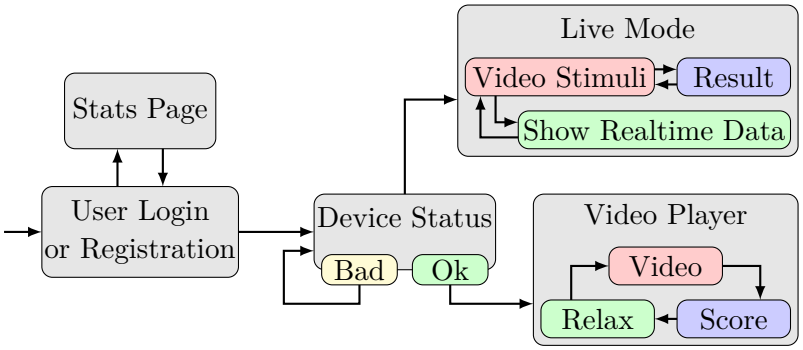


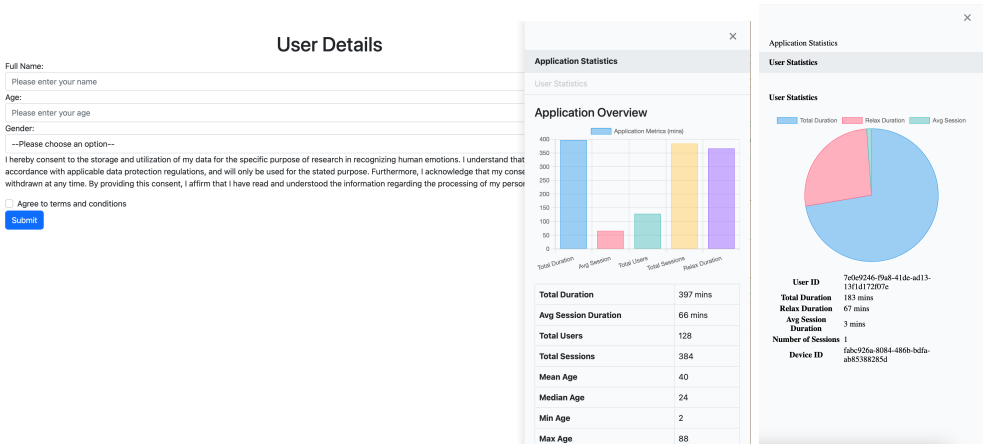
Figure 2.4: Simplified GUI flow.

2.2 Graphical User Interface

To control the experimental flow in an automated fashion while also ensuring data recordings of all specified devices, a graphical user interface has been developed during the last semester's project [SRR+24]. In this iteration, the core functionality of showing videos while recording related data has been changed slightly. Experiments conducted for that part of the GUI are the same as in the last report [SRR+24]. However, two new functional blocks have been added to the core GUI, shown in figure 2.4.

The login page is responsible for saving basic user data such as *name*, *age*, *gender* and the actual consent of the user regarding saving and processing personal information for scientific purposes. When a user fills in every needed aspect and consents to the mentioned processing of personal data, acquired personal information gets stored in a secured SQL database. If a user already exists in the used SQL database, the according user folder is searched in the newly implemented data structure for user folders, explained in section 2.3. If it is found, the specific session file is moved into the GUIs working directory, so it gets appended with new session timestamps. However, if it is not found or the user is new to the SQL database, a new session file gets generated in the working directory. Newly recorded data will also be saved in the working directory of the GUI. Once a session is finished, recorded data and the corresponding session file get moved into the specific user folder of the data structure. If it is a new user, the specific user folder will also be generated automatically.

The first new functional block is the *stats page*, which shows statistics about the current recorded dataset. It is split into two parts, namely *Dataset Statistics* for general information about the dataset itself and *User Statistics* for more user-specific information. It appears as a pop-in window on the right part of the GUI screen and can be accessed by a button in the top right corner. Figure 2.5 shows the new stats page and



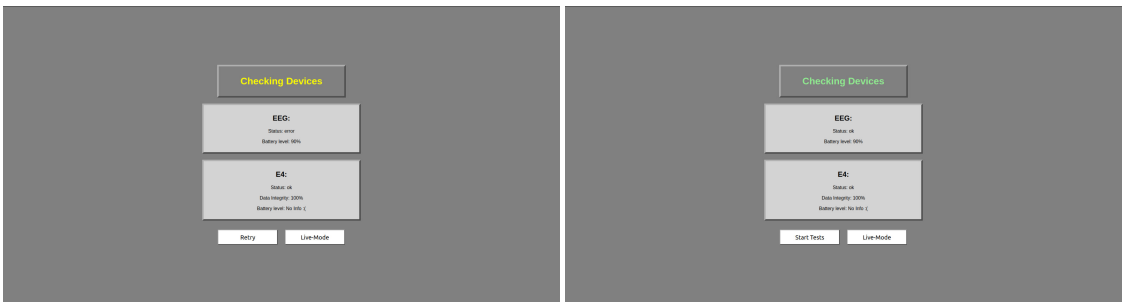
(a) The login page used in the last iteration of the project and the new dataset statistics page.

(b) The new user statistics page.

Figure 2.5: The current login page and the new stats page functionality.

the current login page of the GUI.

The next page is the device status page. It has only been modified by an additional button for the newly implemented *Live mode*. The live mode is meant to show short video parts and simultaneously display the guessed emotion in real-time. It is explained in depth in chapter 6. Apart from the live mode, a user can also enter the regular offline video experiments developed in the last iteration, when the devices are found to be working sufficiently. This is checked by recording two seconds of the wristband, for example, and counting how many data points have been received compared to what is expected. If the actual received data is more than 95% of what has been expected, the user can continue with the offline experiments as well. The modified device status pages are shown in figure 2.6.



(a) Device status page yielding a warning.

(b) Device status page with a good status.

Figure 2.6: The current device status page with a new live mode button.

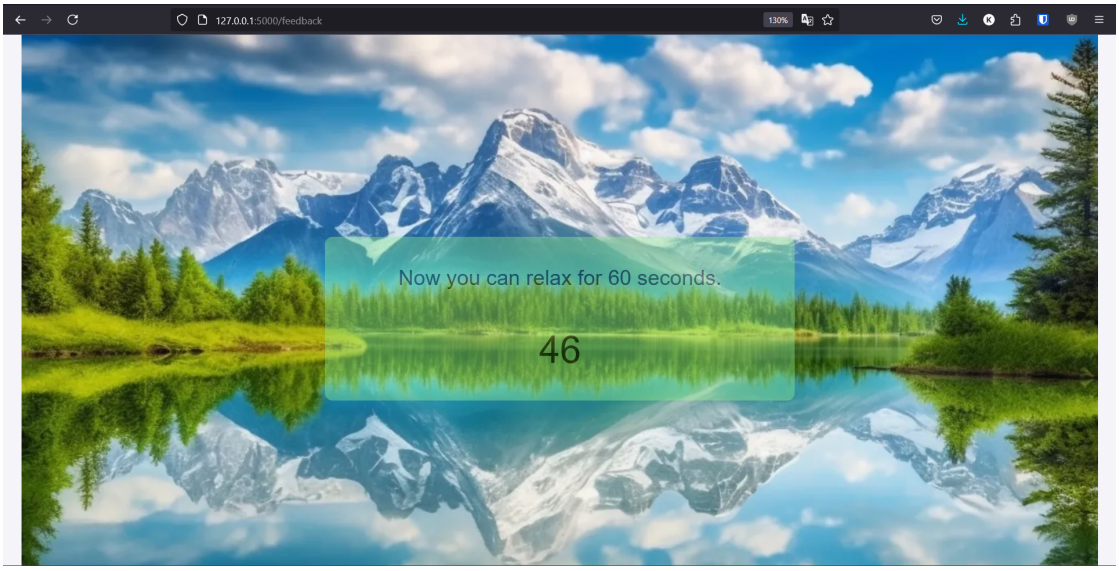


Figure 2.7: New relax page of the GUI.

The last functional block of the GUI is the video recording methodology implemented last semester [SRR+24]. Experiments conducted are videos shown to the user selected from a subset according to four different emotions. After each video, the user is asked for feedback about how strongly the desired emotion has been induced, as this can vary widely by the specific subject under test. After feedback has been provided, a relax page is shown. Unlike last time, where it has been just an empty page with a countdown, currently, the countdown is displayed mostly transparent on a relaxing background image. This new relax page is shown in figure 2.7. After the relax page, either the next video is shown or a user can end the specific session. Videos are presented in a way that ensures no video is played twice and another video of a certain emotion can only occur after all other emotions have been presented just as often. The described experimental video flow is identical to the implementation done in the last report, as it is working satisfactorily [SRR+24].

2.3 Recorded Dataset

As the amount of data increased significantly compared to the data available for the last report, saving can not be done manually anymore. Therefore, a data structure has been developed, which the GUI actively references for already known users, just as it also uses it to save user results anyways. The main GUI interaction has already been described

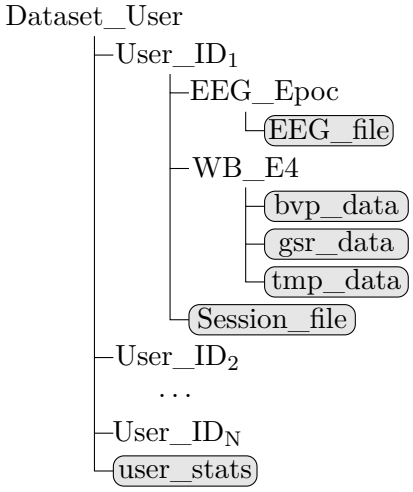


Figure 2.8: The used dataset structure.

in section 2.2. Here, the specific structure is of interest, as it can also be used without the GUI to be processed by data acquisition tools like the pipeline described in chapter 3. The implemented structure is shown in figure 2.8.

The main folder of the data structure is called *Dataset_User* and it consists of a user folder for each user and a *user_stats.csv* file. The user stats file is used for the stats page by the GUI. It is updated after each session or automatically generated in case no stats file has been found. The user folders are named by the specific user ID to ease the search for the GUI. Each user folder contains subdirectories for each device, which has provided data at least once, and a session file named *userID_session_file.csv*. The session file will be moved out in case the specific user requests another session and gets moved back once the new session has been appended. This ensures that every user has exactly one user file. The device folders contain every data file ever saved with this data structure. Additionally, if a tool like the in chapter 3 proposed pipeline requires a specific format to work with the data, each device folder can contain various subdirectories for converted data. However, the original data file, just as the main structure itself, needs to stay untouched. This ensures that other tools can also work on the same data structure without the need to edit or even copy it for another tool.

Based on this new addition to the automated approach of collecting data, the dataset has been increased from about 3.5 h to slightly above 9 h of combined useful data recorded by every device specified. The amount of contributing users has also almost doubled from seven to twelve. More details can be observed in table 2.1. *Total Content Time* is the maximum amount of data based on video and relax timestamps a user can contribute

User	Gender	Age	Total Content Time	Video to Relax ratio	Total Raw Contribution
1	male	24	42.5 min	3.6	7.8%
2	male	25	17.7 min	3.3	3.2%
3	male	28	14.5 min	1.0	2.7%
4	female	27	16.5 min	3.0	3.0%
5	male	22	18.2 min	3.5	3.3%
6	male	27	107.3 min	3.4	19.7%
7	male	24	98.0 min	3.8	18.0%
8	male	23	42.3 min	4.2	7.7%
9	male	34	63.1 min	3.7	11.6%
10	male	27	61.9 min	3.3	11.3%
11	male	27	46.0 min	2.2	8.4%
12	male	24	18.0 min	3.4	3.3%

Table 2.1: User statistics in the collected dataset.

for any data acquisition tool. The *Video to Relax ratio* describes how much total video time a user can contribute per relax time. An ideal ratio would be four as currently, four different emotions alongside relax have been tested by videos. This has only been roughly achieved by three users. However, as the pipeline described in chapter 3 uses a cutaway, which relatively cuts more relax than video data, thus increasing the video-to-relax ratio for actual data used for later algorithms, this is not an issue for most users.

Additional information about an actual output used for machine learning algorithms can be obtained by table 2.2. Here, statistics from an actual data acquisition tool are shown. *Unification Time* describes the total amount of actual data from all specified devices. It should be longer than the timestamp-based contribution time, as the unification time also includes the setup before any video is shown. *Raw Segment Time* describes the amount of data which can be directly used for later algorithms, while *Filtered Segment Time* describes the number of good quality segments obtained by the used tool. The ratio of unfiltered and filtered data is given as *Acceptance Rate*, as it is an indicator of data quality based on wristband data. The *Final Contribution* column shows the specific user's data contribution after filtering has been applied.

User	Unification Time	Raw Segment Time (Count)	Filtered Segment Time (Count)	Acceptance Rate	Final Contribution
1	48.0 min	32.5 min (122)	29.3 min (110)	90.16%	11.16%
2	15.5 min	9.9 min (37)	8.8 min (33)	89.19%	3.35%
3	30.5 min	8.8 min (33)	7.7 min (29)	87.88%	7.51%
4	18.2 min	12.3 min (46)	0.0 min (0)	0.00%	0.00%
5	20.7 min	13.9 min (52)	13.9 min (52)	100.00%	5.27%
6	115.2 min	80.5 min (302)	56.3 min (211)	69.87%	21.40%
7	43.5 min	32.3 min (121)	14.9 min (56)	46.28%	5.68%
8	45.2 min	33.6 min (126)	33.6 min (126)	100.00%	12.78%
9	65.8 min	48.5 min (182)	45.1 min (169)	92.86%	17.14%
10	23.1 min	17.9 min (67)	17.3 min (65)	97.01%	6.59%
11	44.3 min	26.9 min (101)	26.7 min (100)	99.01%	10.14%
12	19.7 min	13.3 min (50)	9.3 min (35)	70.00%	3.55%

Table 2.2: User statistics using the pipeline v1.1.7. Settings used are: segmentation length of 16 s, cutaway of 30 s and a threshold of 10%.

3 Data Acquisition Pipeline

In this chapter, the current data acquisition pipeline and its additions compared to its initial form of the last semester will be discussed [SRR+24]. It is an automated approach of gathering data in a specific form suitable for the in chapter 4 described algorithms. The pipeline will output small pieces called *segments* cut from the original data based on session-file information. Optionally, it also outputs a *feature-file* containing the results of all specified features for all segments. The general pipeline flow is shown in figure 3.1.

The pipeline consists of two files named *pipeline.py*, which holds the configuration and sets the overall flow, and *piepm.py*, which provides all used functions. To set the pipeline up properly, the necessary files have to be copied into a dataset folder of the structure described in chapter 2.3. It then can be executed directly as it automatically searches for present user folders. Optionally, the users considered for a pipeline run

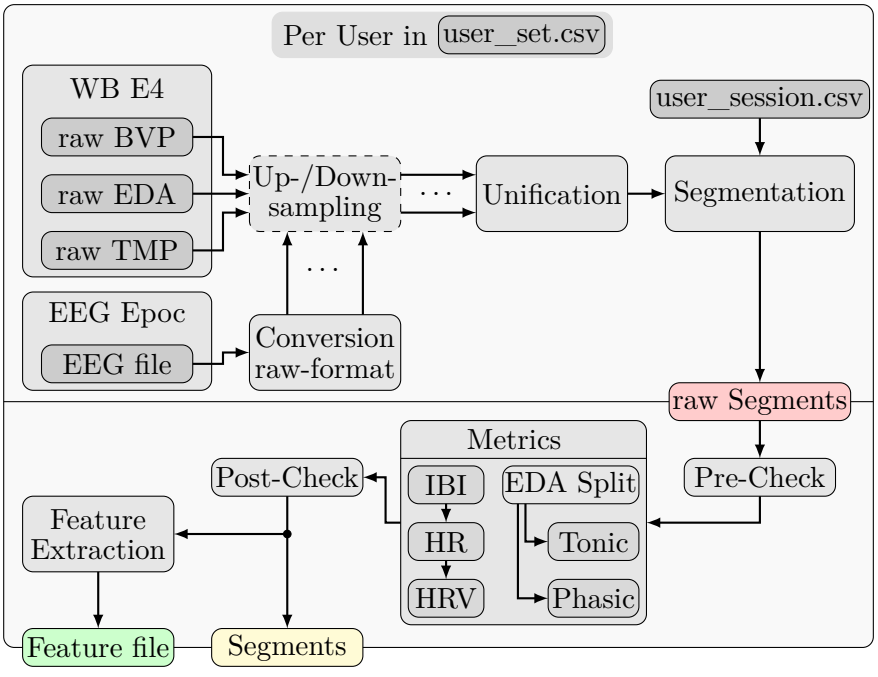


Figure 3.1: Simplified data aquisition pipeline. Dashed elements are not included yet.

can be limited by a csv-file called *user_set*, which consists of the wanted user IDs. Additionally, key parameters can be edited per run to allow for easy comparisons and to ensure compatibility for various machine learning algorithms and their specific needs regarding input data. The pipeline allows to select just specific devices, change the segmentation length and type, select and configure or even skip the metrics and to compute the feature file if needed.

To explain the functionality of the developed data acquisition pipeline, this chapter is split into multiple sections for specific key steps. First,section 3.1 describes the steps from device-specific saving formats to a unified file ready for segmentation. After that, section 3.2 lists segmentation options and shows their characteristics. As raw segments might be of bad quality, basic checks can be employed, outlined in section 3.3. Additional metrics computed from wristband data are also provided in this section. Optionally, a feature-file can be generated, which will be treated in section 3.4. Finally, section 3.5 evaluates the performance of the developed pipeline regarding execution time and usefulness of the provided metrics.

3.1 Preperation Phase

The first difference to last semester's pipeline is that this time, multiple devices are supported, providing as many modalities in an automated fashion as possible. However, different devices like the EEG EpocFlex save data in a different format than the used wristband. This means that the first step for every device is to check whether the 'raw' format is present in the dataset or not. If it is not present, in case of the EEG for example, the pipeline will create a subdirectory in the EEG folder in which it generates 'raw' data files for each sensor of the EEG device. The original data file is not edited and thus stays untouched to ensure that other tools expecting the original dataset can still work on it simultaneously. If these raw-files in their specific subdirectory are already present, the pipeline only checks whether the expected naming is correct and if a file for each specified sensor exists or not. However, if a raw file exists, its internal contents are assumed to be properly copied from the original data file. Figure 3.2 shows an exemplary raw-file for BVP data.

The next step is the option to change the sampling rate of individual sensor data streams. Currently, this is not implemented as no later algorithm needs a tweaked or even an equal sampling rate for all data streams. This means that all following steps will use the individual native sampling rate, which differs from $4\,{}^{\mathrm{Sa}}/_{\mathrm{s}}$ for EDA up to $128\,{}^{\mathrm{Sa}}/_{\mathrm{s}}$ for EEG electrode-streams. Consequently, the third and last step of the preparation

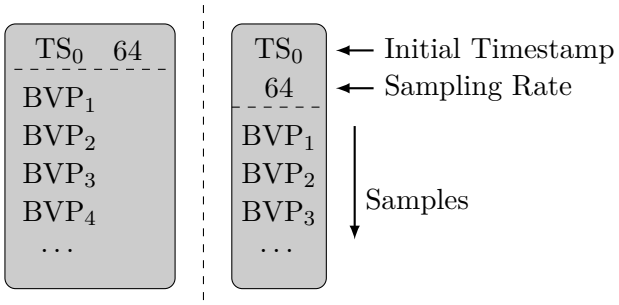


Figure 3.2: Raw BVP file format already shown in [SRR+24]. Left is the real time recorded format used by the gui for the provided dataset, right is Empaticas online format.

phase, called unification, can not just align those data streams as the actual amount of data points per second, but also in total, differs significantly. Additionally, the initial timestamp of the raw-files also does not start at an equal point in time. There are two reasons for that. The first is the sequential starting of device-specific reader processes by the GUI, which results in the wristband data being recorded about two seconds earlier than the EEG data. The second reason is that even the data streams of the same device, for example the wristband, never share an equal timestamp, meaning they are also reported in a sequential order.

To circumvent the first issue regarding an unequal sampling rate, the unification stage uses the saving format presented in the last semester's report, which loosens the exact timely information as shown in figure 3.3 [SRR+24]. Here chunks of a length of one second are used, which allows the segmentation to later cut any multiple of one second. Data points of a stream belonging to a timestamp ranging from *hh:mm:ss.000000* ... *hh:mm:ss.999999* are considered part of the same chunk. Milli- and microsecond information is only used for the timely order of the data points inside a chunk and discarded afterwards. Only one timestamp referencing the beginning row of a chunk is saved. Each column gets filled with its respective data until none for the specific second is left. The length of each column is dictated by the highest used sampling rate, but the amount of useful data per column differs based on the individual sampling rate.

The second issue of unequal starting timestamps is handled in two ways. Regarding various per-device starting points, the pipeline knows in which order the GUI starts the recorder subprocesses and thus it initially searches for the first recording device. Once this is found, it searches for data of all remaining devices whose timestamps are allowed to differ up to ten seconds from the first recording devices timestamp. Regarding the specific sensors having different starting timestamps, unification will search for the

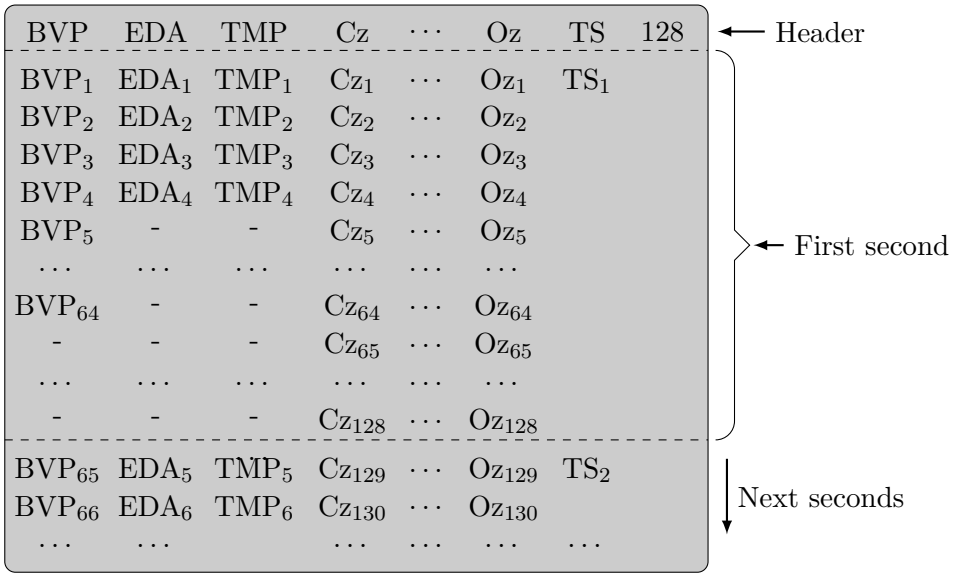


Figure 3.3: Beginning of a unity-file containing wristband and EEG columns. Only necessary format elements are shown.

first complete chunk in which all sensors provide complete data. All earlier data points are discarded. This results in subsequent chunks providing complete data for later algorithms which expect it. The disadvantage of this approach is that later sessions, using a different wristband for example, can not just be mixed with earlier sessions, as the pipeline expects all devices to be present simultaneously and not interchangeably.

3.2 Segmentation

The next important component of the pipeline is the actual segmentation. This step takes all unity files of a user and the corresponding session file to cut equally sized segments, which belong to a specific video. Cutting happens based on timestamps saved in the session file and the pipeline's configuration data which specifies the segmentation or cutaway length but also the segmentation type. Ideally, for every video, there is data from every device present for the entire duration of said video. However, the EEG device, for example, did disconnect sometimes in the middle of a video, leading to incomplete recordings. This sudden stop is then considered the usable end of a video yielding a notice about it when running the pipeline. Equally, a missing session, a session or video that is to short to cut or timely unrelated sessions also yield notifications via the terminal, but they do not hinder execution. At the end of a segmentation run, raw

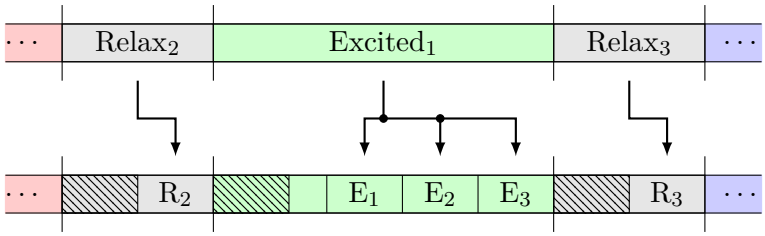


Figure 3.4: Normal segmentation using 30 s cutaway and 30 s segments.

segments are saved into a specified subdirectory. These segments format is equivalent to the unity-files format except for their length being a specific multiple of an integer in seconds. Their naming is based on the representing emotion and a counter, yielding for example *fear_1.csv* or *happy_11.csv*. The counter will not be reset per session or user but per execution of the pipeline, so if a user provides eleven happy segments, the next user's first happy segment will get the counter value twelve assigned.

Currently, the pipeline does support multiple segmentation modes, namely *normal*, *overlapping* and *raw*. The raw mode is not a real segmentation mode in terms of cutting equally sized parts. It just cuts the entire video-related data to get rid of unwanted parts after the last video plays, for example, and thus is only useful for debugging or experimenting.

The normal mode has been proposed already in the last iteration of the pipeline [SRR+24]. It is the default segmentation mode visualized in figure 3.4. To prevent previously induced stimuli from contaminating the current desired stimuli by carrying on for some time into the current video, the beginning part of each video or relax phase is cut away. Furthermore, this cutaway is enhanced by the remaining data, which is to small to cut. Cutting segments starts from the end of a video, which is assumed to be only influenced by the desired stimuli. This continues till the amount of data left is not sufficient for another segment. The amount of segments cut per video length can be expressed by the following formula, assuming the video length is sufficient:

$$N_{\text{segments}} = \left\lfloor \frac{t_{\text{vid}} - t_{\text{cut}}}{t_{\text{seg}}} \right\rfloor \overset{\text{example}}{=} \left\lfloor \frac{135\,\text{s} - 30\,\text{s}}{30\,\text{s}} \right\rfloor = 3 \tag{3.1}$$

Using the formula for the Exicted$_1$ video shown in figure 3.4 yields three segments and 15 s of leftover data used as an extension for the cutaway part.

Finally, the last segmentation mode, called overlapping mode, is a new addition to the segmentation stage visualized in figure 3.5. Instead of using data just for one segment, this mode allows for some data to be used multiple times, by overlapping the segments by

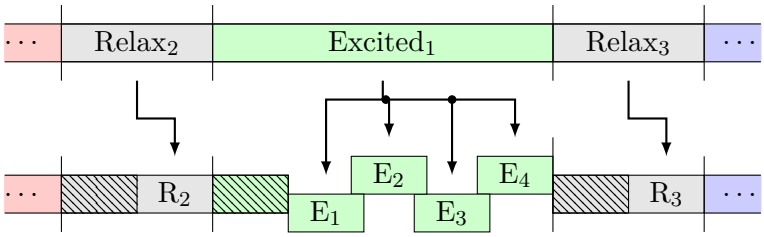


Figure 3.5: Overlapping segmentation using 30 s cutaway, 30 s segments and a 5 s overlap.

a specific amount. This mode also uses a cutaway at the start of a video and segments it beginning from the end of a video. Using figure 3.5 as an example, the initial segment E_4 consists entirely of unused data like in the normal mode, which is 30 s. The next segment E_3 however does only contain 25 s of unused data as the example uses an overlap of 5 s. This means that every segment after the first one contributes a smaller amount of new data than the first one. When using formula 3.1 to calculate the amount of segments cut out of a video, the effective segmentation length has to be used for all but the first segment. The first segment is basically treated as another cutoff area and its contribution as a segment is added afterwards. This way the amount of segments cut per video using the overlapping mode can be expressed by formula 3.2, assuming sufficient video length.

$$\begin{aligned} N_{\text{segments}} &= \left\lfloor \frac{t_{\text{vid}} - (t_{\text{cut}} + t_{\text{seg},1})}{t_{\text{seg,eff}}} \right\rfloor + 1 = \left\lfloor \frac{t_{\text{vid}} - t_{\text{cut}} - t_{\text{seg}}}{t_{\text{seg}} - t_{\text{ovl}}} + \frac{t_{\text{seg}} - t_{\text{ovl}}}{t_{\text{seg}} - t_{\text{ovl}}} \right\rfloor \\ &= \left\lfloor \frac{t_{\text{vid}} - t_{\text{cut}} - t_{\text{ovl}}}{t_{\text{seg}} - t_{\text{ovl}}} \right\rfloor \overset{\text{example}}{=} \left\lfloor \frac{135\,\text{s} - 30\,\text{s} - 5\,\text{s}}{30\,\text{s} - 5\,\text{s}} \right\rfloor = 4 \end{aligned} \tag{3.2}$$

The maximum count of how often some data is being used is given by formula 3.3.

$$N_{\text{usage\_factor,max}} = \left\lceil \frac{t_{\text{seg}}}{t_{\text{seg,eff}}} \right\rceil \tag{3.3}$$

Not only does this segmentation method allow to maximize the amount of used data by using a finer-grained effective segmentation size, but it also can result in many more segments to use. However, apart from the fact that some parts of data get used more often than others without any knowledge about the specific data parts importance, this only works well for sufficiently sized videos. The relax parts, for example, are not big enough to obtain any more segments from them, meaning the amount of emotion based segments compared to the amount of relax segments will be out of proportion.

3.3 Additional Metrics And Checks

The next major element of the pipeline is to add metrics based on specific data streams in the segments. This functionality is optional meaning it can be skipped per run if it is not needed. Additionally, except for the checks, all metrics added are based on wristband data only. Compared to the last iteration of the pipeline, all elements of the metric addition stage have been kept, some are tweaked and extended and some entirely new are being added. The following list gives an overview about which elements are available and which have been modified since the last iteration of the pipeline [SRR+24]:

- Precheck: Same as in the last iteration.
- IBI: Improved and extended version, which is based on the last iteration.
- HR: Same as in the last iteration.
- HRV: New metric.
- EDA split: New as a metric with a completely new self made algorithm.
- Postcheck: Same as in the last iteration but can be disabled now.

3.3.1 Inter Beat Interval

An inter-beat-interval, also called IBI, describes the time between two heartbeats. Based on BVP data it can be calculated by using the timely difference between two systolic notches. Unlike the ideal model, real data contains artifacts which make the identification of actual notches a challenging task. For getting as many viable beat candidates as possible, an IBI algorithm consisting of two phases has been proposed in the previous iteration of the pipeline. In the current iteration, both phases are being modified and a third stage is added.

Phase one extracts all raw notches and peaks it can find. Additionally, it applies basic filtering to leave out any potential candidate that violates the following basic conditions:

1. Valid notches or peaks have to be more than 20 samples apart from each other.
2. The first overall candidate is a notch, the last is a peak.
3. Notches and peaks have to appear in pairs.
4. Notches have to be negative, peaks have to be positive.

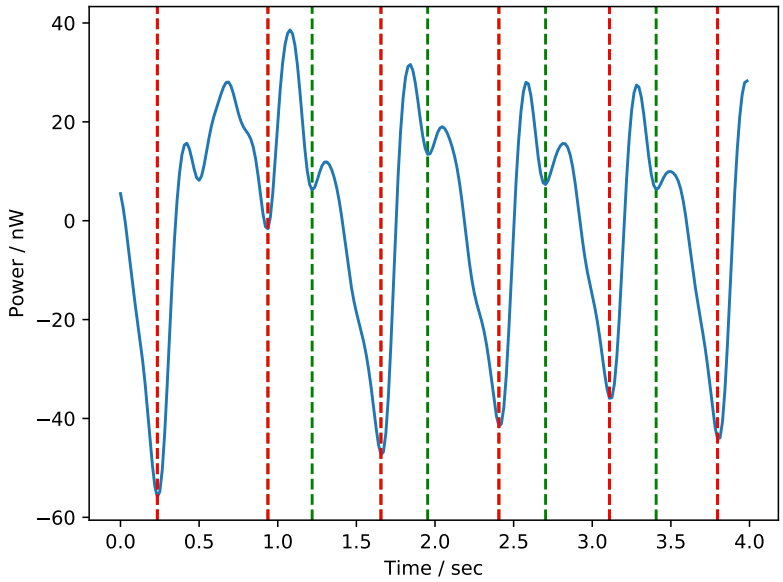


Figure 3.6: First seconds of a segment with marked candidates. Red candidates are left after phase one, while green candidates have been filtered out.

Compared to the last iteration only the fourth statement is new. Given a sampling rate of $64\,{}^{\mathrm{Sa}}/_{\mathrm{s}}$ regarding the used BVP data, the first statement ensures that an instantaneous heart rate of slightly more than $180\,{}^{\mathrm{Beats}}/_{\mathrm{min}}$ can be detected. This is more than sufficient for the conducted experiments. The second statement ensures that the first notch and the last peak belong to a pair, as they have to exist as pairs representing the heart contraction activity. This is ensured for the entire data by the third statement. Using it, the first phase will filter out consecutive notches and peaks in a way that the later notch or earlier peak belongs to a proper pair, while remaining singles are deleted. The last and also new statement enhances the chance of picking a 'good' starting point for later phases by ensuring that a notch has to be below and a peak above zero. It basically uses the mean value of BVP data as a threshold. Figure 3.6 shows the result of phase 1 at the beginning of a segment.

After phase one has identified valid candidates, phase two filters them further by their main characteristic values shown in figure 3.7. Consecutive pairs and their respective notches and peaks are expected to have relatively similar characteristics. For example, the heart compression time of a specific pair is expected to be inside a threshold around the previously accepted pair's value. If it is inside of the expected range, this specific

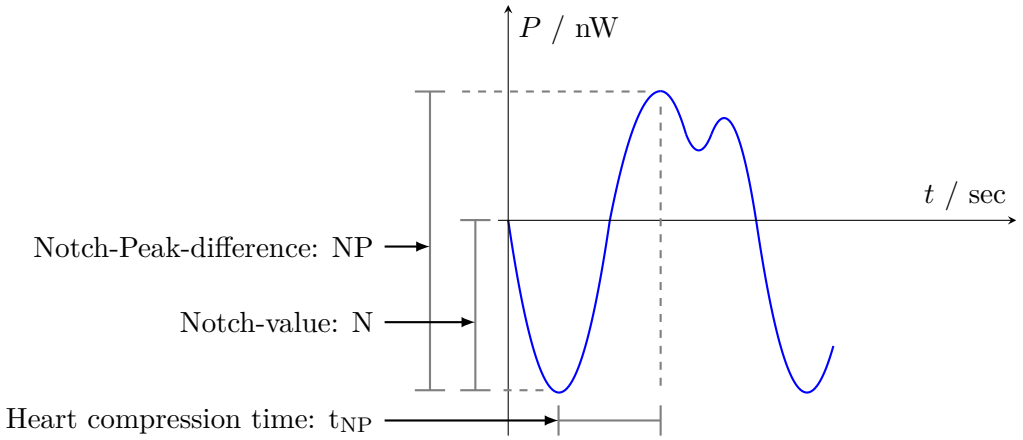


Figure 3.7: Used BVP characteristics to determine IBI candidates in phase 2. The figure is directly taken from [SRR+24].

pair's value will define the new threshold values. If it is not inside the expected range, it is marked as a bad candidate and the threshold will be increased by 10% for the next pair to check. This is done for all characteristics of all consecutive candidates and has been implemented in the last iteration already. In the current version however, this algorithm is not just executed once but up to five times, as the first selected candidate can make a huge difference in proper filter performance. Each time a different starting candidate is chosen and after all five executions finished, a majority vote decides which candidates to keep and which candidates need to be filtered out. A short comparison of the old single run and the new multi run phase two is given in appendix A.1. It uses the given beginning of a segment and a complete segment with a bad first candidate.

However, even with the new phase two, candidates with bad characteristics will be filtered out even when their actual position makes sense based on the resulting interval lengths. For the given example in figure 3.6 the second, and also third, candidate will still be filtered out, as the characteristics of the second candidate are just way to different from its surrounding neighbours. Assuming only the second candidate gets filtered out, this would result in an interval of about twice the following intervals lengths. Phase three calculates all intervals based on those candidates left, which after all filtering applied are assumed to be representing actual beats. From those, it chooses one of the shorter intervals, usually the 40^{th} percentile, as a reference interval length because, due to removed candidates, longer intervals are far more likely than shorter intervals. For every interval which is more than 50% longer than the reference interval, potentially useful candidates from all removed candidates are searched. First the amount of smaller

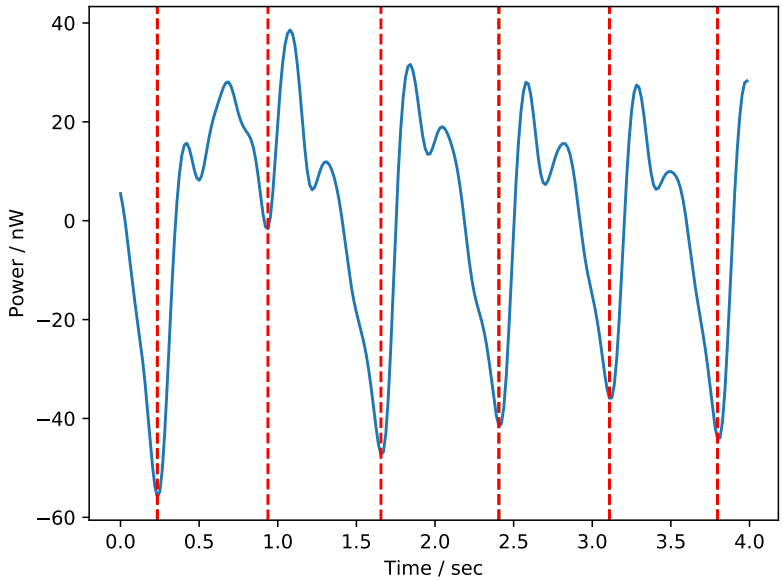


Figure 3.8: First seconds of a segment with marked candidates. Red candidates are left after phase three. No bad candidates but an artifacted one are visible.

intervals to fit is estimated by dividing the problematic interval length by the reference interval length and rounding to the nearest integer. Then the problematic interval length is divided by the amount of estimated smaller intervals to get the specific smaller interval size. Now, starting from the problematic intervals beginning index, adding a multiple of the specific smaller interval size, beginning by one up to the amount of estimated candidates, plus-minus an epsilon of uncertainty is used to search for removed candidates. As candidates are at least 20 sample points apart an epsilon of ten sample points has been chosen. If a removed candidate lies inside of the searched area it is added back and if no removed candidate is found around the estimated positions, nothing is added back.

Regarding the initial example, figure 3.8 shows the result after phase three. The second candidate which is removed by phase two due to bad characteristics, is now added back by phase three due to its expected position. Here, the difference between the first and third candidate yields an interval of about 1.42 s. Choosing a reference interval of 0.76 s and dividing the problematic intervals length of 1.42 s results in two smaller intervals to fit with a specific length of 0.71 s each. Starting with the first candidate's position of about 0.24 s, adding the specific length of 0.71 s and also accounting for the

allowed uncertainty of $\pm 0.156\,\mathrm{s}$ yields a searching area of $0.794\,\mathrm{s} \ldots 1.106\,\mathrm{s}$ for a removed candidate. As the second candidate is positioned at about $0.94\,\mathrm{s}$ it is found and thus added back. In this case the result of phase three is equivalent to the result of phase one. However, especially for segments of bad quality this is not always the case. For those, adding back specific candidates removed by the second phase often is not enough to fit all intervals to a reasonable size.

3.3.2 Heart Rate

The second metric calculated by the pipeline is the instantaneous heart rate, given in formula 3.4, which is exactly calculated as in the previous iteration [SRR+24].

$$\mathrm{HR}_{\mathrm{inst}} = \frac{{}^{\mathrm{sec}}/_{\mathrm{min}}}{{}^{\mathrm{sec}}/_{\mathrm{Beat}}} = \frac{60}{\mathrm{IBI}}\,{}^{\mathrm{Beats}}/_{\mathrm{min}} \tag{3.4}$$

As the IBI algorithm still outputs raw intervals by their middle as their respective position in time, chunks get unevenly filled, which gets even worse when the specific segment is of bad quality. Therefore, the average IBI length of a chunk and its direct neighbouring chunks is calculated, under the assumption that only two consecutive chunks can be empty, and is then used for the instantaneous heart rate. This is shown by formula 3.5, done using $W = 1$ and results in a sampling rate of $1\,{}^{\mathrm{Sa}}/_{\mathrm{s}}$.

$$\mathrm{HR}_{\mathrm{win}} = \frac{60}{\frac{\sum_{w=0}^{2W}\sum_{l=1}^{L_w}\mathrm{IBI}_{w,l}}{\sum_{w=0}^{2W}L_w}}\,{}^{\mathrm{Beats}}/_{\mathrm{min}} = \frac{60}{\frac{1}{L'} * \sum_{m=1}^{L'}\mathrm{IBI}_m}\,{}^{\mathrm{Beats}}/_{\mathrm{min}} = \frac{60}{\overline{\mathrm{IBI}}'}\,{}^{\mathrm{Beats}}/_{\mathrm{min}} \tag{3.5}$$

This approach still has the same disadvantage of slightly smoothed data, which is a compromise of losing only as much information as needed to get a segment past the final filter. One option for fixing this issue would be to guess an evenly sampled IBI function based on the obtained IBI values, which yields a fixed amount of data per chunk from which the heart rate can be directly calculated.

3.3.3 Heart Rate Variability

The third metric, which has been newly implemented in this iteration, is called heart rate variability. It is basically a measure of how stressed an individual is due to physiological or psychological events. A rather constant high heart rate results in a low HRV indicating stress, while a rather low heart rate changing quickly by spontaneous needs results in a high HRV indicating a relaxed state [HRV21]. HRV is calculated by the difference in

consecutive intervals between beats instead of the difference in the heart rate [Ass96]. However, as metrics of the current pipeline are basically implemented as standalone functions, the exact timely information of the specific intervals is unknown to the HRV function. Given that the IBI data currently is not equally sampled, it can also not be roughly guessed like the timely information of other data.

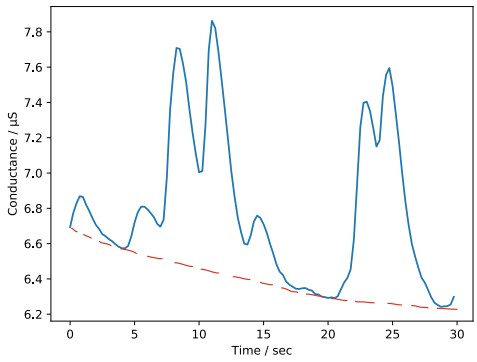
To circumvent this issue the average IBI per chunk is used, which is calculated from the heart rate using the inverse of formula 3.4. This results in the following formula to calculate HRV:

$$\mathrm{HRV}_i = \overline{\mathrm{IBI}}_i - \overline{\mathrm{IBI}}_{i-1} = 60 * (\frac{1}{\mathrm{HR}_i} - \frac{1}{\mathrm{HR}_{i-1}}) \tag{3.6}$$

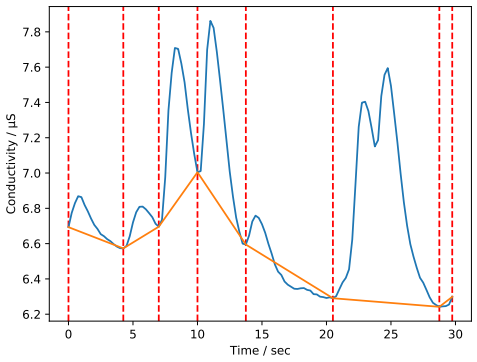
The resulting value inside a chunk is the HRV based on the average IBI of that chunk and the average IBI of the previous chunk. This way, the HRV of the first chunk would be empty as the chunk before it is unknown. Here, the value of the second chunk is copied to ensure a sampling rate of $1\,{}^{\mathrm{Sa}}/_{\mathrm{s}}$.

3.3.4 Electrodermal Activity Components

The last supported metric of the current pipeline is actually a split into two additional modalities. The EDA signal consists of two main components, namely the tonic and the phasic component, which this algorithm tries to extract from the given EDA signal. In the last iteration, this has been done by a third party algorithm but the results were not satisfying, so currently a self made approach is implemented. The main idea of the given approach is to identify the tonic function by identifying the beginning of all phasic components. As phasic components last longer than a second and their start points are also separated by more than one second, the first step is to get the minimum of every second, so basically for each chunk of a segment. Then, those minima representing a phasic component each, have to be filtered out from the list of all minima, as a second without any phasic activity still has a minimum value. This is done by a moving window which notes down the maximum of all minima inside it. After it moves over the entire list of candidates, those noted down will be removed. This yields the beginning of all phasic activity or a slightly later point in case they overlap each other partially. Initially, a window size of three has been used but a size of two yielded better results. Additionally, adding the first and last value of a given segment and connecting the obtained candidates results in the raw tonic function. Figure 3.9 shows a manual guess compared to the algorithmic guess for a good quality segment with multiple even interleaved phasic components.



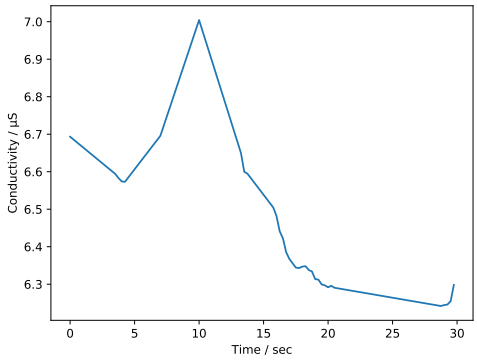
(a) EDA signal with a hand drawn guess.



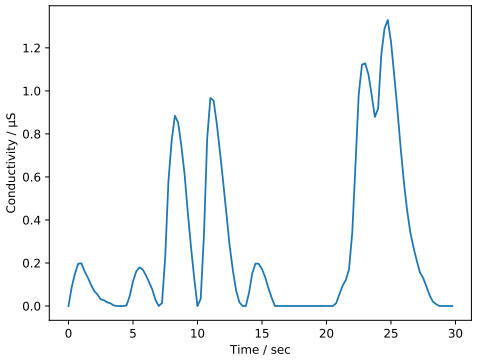
(b) EDA signal and the raw tonic function.

Figure 3.9: Comparison of a guess compared to the raw tonic function obtained by the proposed algorithm. Candidates, and also the start and endpoint, are marked in (b) as well.

Only the last phasic component at about 24 s is not detected, which in this case would result in an unreasonable peak in the tonic function. However, the raw tonic function intersects the EDA function multiple times, which should never happen as the EDA function is the sum of both only positive components. To fix this issue, for each data point the minimum of the raw tonic function and the EDA signal is set as the new value for the tonic function. The phasic function can then be calculated by subtracting the



(a) Tonic function of the EDA signal.



(b) Phasic function of the EDA signal.

Figure 3.10: Phasic and tonic functions obtained by the proposed algorithm.

tonic function from the EDA signal. The result can be seen in figure 3.10.

One obvious problem is the peak in the obtained tonic signal which dampens the amplitude of the phasic signal at the specific time. Such peaks could be filtered or at least dampened by using a max slope or max slope change condition for the tonic signal. However, as most of the amplitude and all peaks of the phasic component are identified correctly, most of the desired information is extracted. Additional problems lay in the start or end of a segment when a phasic component is only partially present. This leads to a steep tonic function at those parts, completely erasing the desired phasic component. Another rare problem lies in the candidate generation itself, as a steady slope with a phasic component at its end will likely not have the beginning of the phasic component as a minimum. And even when it does, it will be filtered out by the previous slope part. This can be circumvented by combining the current candidate specifier with a search for relatively steep slopes. However, such a slope detection does not work on bad quality EDA signals, because the sensor resolution being close to the total segment value variation leads to many of said steep slopes compared to the total value range of a segment. Examples of a problematic start or a mediocre quality EDA signal with the algorithmic results are given in appendix A.2.

3.3.5 Checks and Filter

Prechecks are identical to the last iteration's implementation. They still check the same core criteria being a segments length matching the expected length, all core modalities being present and also all modalities providing data to the segment. If one of the core criteria is not matched, a segment is deemed to be bad and thus removed. Compared to the last iteration the amount of modalities changed as the current pipeline allows for multiple devices which add dynamically to the list of modalities to check. Postchecks also are quite similar to the last iteration's implementation. They are still done as a filter checking how much data of the specified metrics is outside of a given range of good values. Those ranges are also the same as stated in the last report [SRR+24]. If the relative amount of bad metric data is bigger than a given threshold the specific segment is marked as bad and thus removed.

Removing is also implemented the same way as before, ensuring a consistent naming scheme. If for example after removing segments five happy segments are left, all numbers $1 \ldots 5$ exist as an indize for the left happy segments. This is ensured by renaming checked good segments to fill otherwise emerging holes regarding the indizes. The only new part is that for debugging reasons the filter can be turned off without skipping the metrics.

However, using metrics without filtering those segments with an empty metric result leads to problems for the following feature extraction. Segments of bad quality can lead to the IBI algorithm for example being unable to identify any beats thus leaving an empty column. This issue raises the urge to include a forced filter for segments containing empty columns when the feature extraction is also selected.

3.4 Feature Extraction

The last main function of the proposed pipeline is the included feature extraction. Feature extraction means that core characteristics of a segments signal are calculated, for example the mean values. In the last semester this step has been done by multiple people for multiple devices individually. In the current pipeline all used features are available just as the specific preprocessing. Preprocessing is done to remove unwanted artifacts or to use normalization of some signals if desired. Currently, only the EEG data is preprocessed using a 5^{th} order butterworth bandpass filter with a lower cutoff frequency of 0.5 Hz and an upper cutoff frequency of 50 Hz. Optionally, even other filter types are supported. Specific preprocessing can also be done just for frequency domain features, but this is not selected by default.

After preprocessing the actual feature extraction takes place. It is split into two groups regarding time- and frequency domain features. Each group is further split into a global and specific part. The *time_features_global* list, for example, contains all time domain features applied on each modality of a segment. Similarly, the *freq_features_specific* set contains lists of all frequency domain features to be applied to a specific device or even just to a specific sensor. The following listing contains the pipeline's default groups of features applied:

- time features global: mean, mean-diff, mean-abs-dev, mean-abs-diff, median, median-diff, median-abs-dev, median-abs-diff, standard-deviation, variance, rms, min, max, percentile-25, percentile-75, inter-quantile-range, peak-peak, zero-crossing, area-under-curve, area-under-curve-abs, slope, skewness, kurtosis
- freq features specific:
  - wb_bvp: mean-power-lf, mean-power-mf, mean-power-hf, peak-freq-lf, peak-freq-mf, peak-freq-hf
  - eeg: mean-power, mean-power-delta, mean-power-theta, mean-power-alpha, mean-power-beta, mean-power-gamma

The specific definitions for all of those features are already given in last semester's report [SRR+24]. The pipeline just makes them all available for all devices in a grouped and ordered functionality. The groups applied consist of 23 general time domain features, zero general frequency features, six BVP specific frequency features and six EEG specific frequency features. Using the three wristband modalities and all 29 EEG sensors without any additional metrics yields a dimensionality of:

$$D_{\text{default}} = D_{\text{time}} + D_{\text{freq,bvp}} + D_{\text{freq,eeg}} = (3 + 29) * 23 + 6 + 29 * 6 = 916 \tag{3.7}$$

Including the previously described metrics extends the dimensionality by 115 to 1031 values per segment.

All of those results per segment are saved in csv-format in a so-called *feature-file*. The first row contains the information about which column saves which feature for which specific sensor, resulting in, for example, *bvp_mean*, *bvp_zcross* or *Fz_zcross*. Each following row contains the results of a single segment. Additionally, the labeled emotion of each segment is also saved in a column in this file.

3.5 Evaluation

The last section of this chapter evaluates the functionality and versatility of the proposed pipeline, which will be done in two parts. First, in section 3.5.1 the usefulness of the metrics added, including filter capabilities, are evaluated by choosing an actual machine learning algorithm described in chapter 4. The second part shown in section 3.5.2 focuses on the general pipeline performance and states possibilities for future improvement.

3.5.1 Metric Performance

As the pipeline has been made as an automation tool regarding data acquisition for as many machine learning algorithms as possible, any algorithm from chapter 4 can be chosen for evaluation. However, to include more recent work of this semester's project group, the CNN approach described in chapter 4.4 has been selected for evaluation of the current pipeline. Currently, enhancing or filtering happens exclusively based on wristband data, thus only wristband accuracy is of interest for this comparison. To examine the filter's influence regarding test accuracy all test and training data sets are included with and without filtering. The comparison will show all four permutations of combining filtered and unfiltered train and test data. Additionally, to evaluate the

metrics influence regarding test accuracy, three different modality combinations are used for the wristband, shown in the following listing:

- Default: Standard wristband modalities only
  → BVP, EDA, TMP
- Extended: All default modalities and all but one of the metrics
  → BVP, HR, HRV, EDA, EDA_tonic, EDA_phasic, TMP
- Selected: Most of the metrics and one default modality
  → BVP, HR, HRV, EDA_phasic

The IBI metric is not present in any of the used data sets, as it currently is not an equally sampled signal, which is a requirement for the used CNN methodology. The 'default' data set provides the baseline consisting of raw signals only, while the 'extended' data set includes all usable metrics provided by the pipeline as well. Furthermore, a 'selected' data set is used, which excludes signals that are rather constant for the given segment length.

In this comparison, a segment length of 8 s has been used as it is the ideal length for the used machine learning algorithm recommended in chapter 4.4. All segments are generated with the default cutaway of 30 s and the normal segmentation mode. Every training data set consists from data of the same ten users, while the remaining two users provide test data. This way of splitting into train and test data by specific users is chosen as the alternative random split already provides acceptable accuracies based on default data alone.

The first comparison consists of the default and the extended data sets, whose results are shown in figure 3.11. Here, the default modalities yield accuracies of about 21% regardless whether filtering any data or not. This means that the default set marking the baseline is just marginally better than a random guess for five equally balanced emotion classes. Using the extended data set, which uses all default modalities but also adds multiple new metrics, worsens the resulting test accuracy. When using unfiltered data, the resulting accuracy drops from about 22 to 18 percent and when filtering is also applied, it drops even further from 21 to below 11 percent. This can be explained by overfitting of the used machine learning algorithm. Using all modalities provided by the pipeline requires sufficient amounts of data, which the currently collected dataset can not provide. Subsequently, further reducing the amount of data available by filtering also reduces the resulting accuracy by a significant amount.

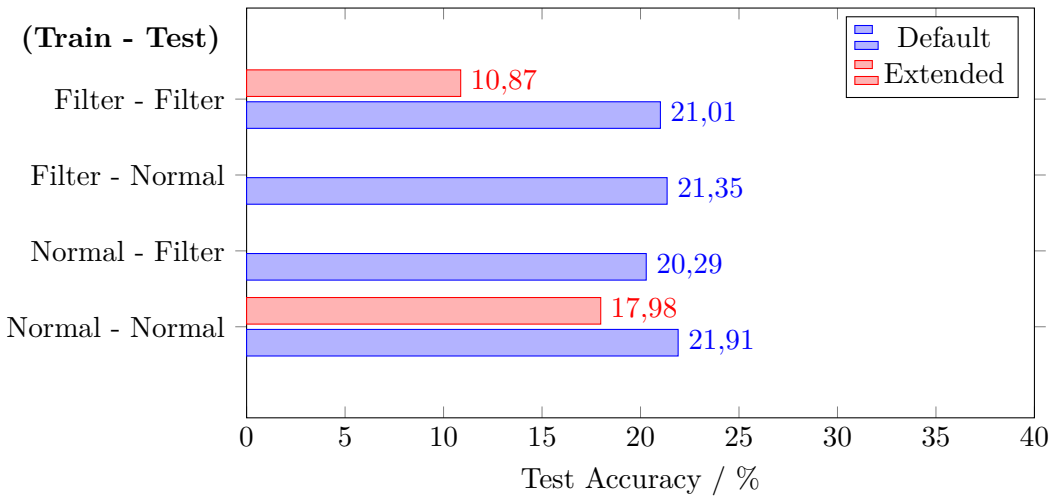


Figure 3.11: Comparing the achieved test data accuracies regarding default and extended modality sets.

The next comparison tries to circumvent the overfitting issue by using about as many input data streams as in the default case but focusing mainly on the new metrics as they are assumed to be more useful. This results in an actual improvement compared to the baseline, shown in figure 3.12. Here, the accuracy improves from about 22% to 26% when using unfiltered data and from 21% to 32.6% regarding filtered data. This can be explained by the used metrics regarding heart activity as filtering happens solely on them. Bad quality segments yield potential empty chunks regarding the IBI metric as the involved algorithm can not find any beats. This leads to significant segment portions being zero regarding the HR and HRV metrics, as both are based on the IBI results. Using those segments hinders performance while filtering those allows for useful training improvements. Additionally, the model trained on filtered data is also being tested with unfiltered data, which as expected performs slightly worse by dropping about 1% below the best-case result.

The resulting performance is still much lower than what would be possible by using a random data split approach as the test methodology. However, this comparison has been made to show potential improvements by using the functionality provided by the developed pipeline and its described algorithms. Here, a clear improvement over the baseline performance can be observed, justifying time invested in implementing additional metrics just as the metrics themselves as a useful addition.

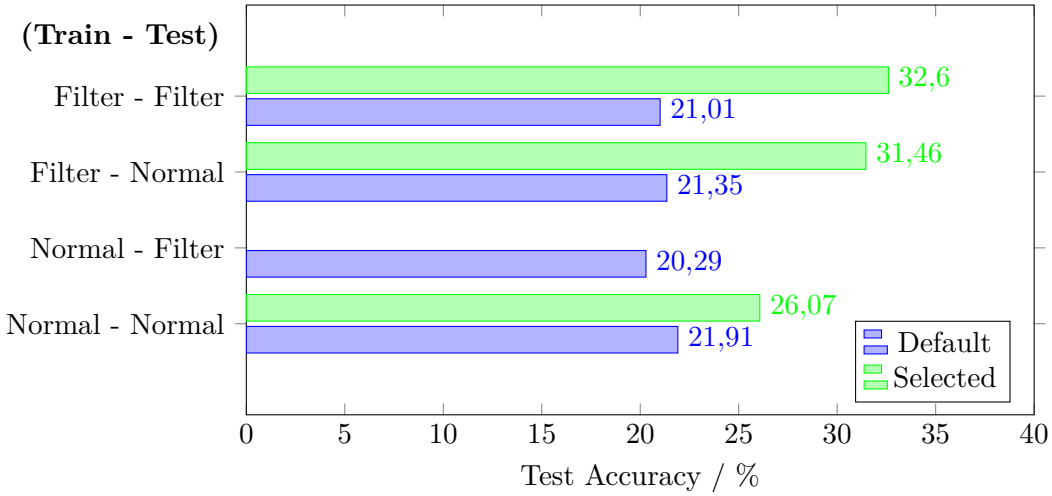


Figure 3.12: Comparing the achieved test data accuracies regarding default and selected modality sets.

3.5.2 Pipeline Performance

Apart from the pipeline's output, the actual pipeline performance regarding algorithm efficiency and overall execution time shall also be evaluated. Here, the only actual component to compare is the IBI algorithm of the current pipeline implementation to the last iteration's implementation. Key characteristics are displayed for various segmentation lengths in table 3.1.

The new implementation can be observed finding about 8.9%...11.4% more intervals than the old implementation, while longer segments provide better improvements. This significantly improves filtering as the new algorithm deems only about 25% of segments

Version	Segmentation length	Intervals	Warnings	% of removed Segments
1.1.3	8 s	22426	-	35.85
1.1.3	16 s	23807	-	41.91
1.1.3	32 s	23627	-	39.28
1.1.4	8 s	24432	14	23.83
1.1.4	16 s	26119	0	25.13
1.1.4	32 s	26320	0	24.2

Table 3.1: Comparing the old and new IBI algorithm using different segmentation lengths. The last pipeline version providing the old and the first pipeline version providing the new IBI algorithm are being used.

to be bad compared to about 40% of the old implementation. This can be explained by many segments, barely missing the threshold condition before, are satisfying it now by just identifying one or two more beats in the individual segment. Another thing to add is that longer segments generally provide more identified intervals, while also containing less total segmented time. Shorter segments on the other hand yield warnings about very few or even no beats at all being found on multiple occasions. Consequently, the IBI algorithm should not be used for segments shorter than 8 s and overall beat detection would be maximized by applying it directly to the unify-files before segmentation takes place. However, this would require a considerable change to the pipeline's architecture.

Another optimization lies in the execution time of the pipeline. Just the EEG conversion into the desired file format applied to the current dataset takes almost nine minutes, which is about as much as the entire remaining pipeline run. Here a decent speedup can be achieved by using as many CPU cores as possible because each conversion for each session of a user is independent of others. A similar improvement can be applied to the unification stage, which mostly makes up another significant amount of runtime, as shown in figure 3.13.

Regarding the metrics, except for IBI, all metrics have an increasing execution time when more segments, containing roughly the same amount of data, are being processed. This can be explained as reading and writing for each algorithm adds up to a measurable overhead. This can be prevented by a function reading and writing the segments just once by communicating with all the algorithms directly instead of the individual algorithm reading and writing by itself. Furthermore, calculating metrics per segment or unity file can also be done in parallel for each individual file. The IBI algorithm taking longer on fewer segments containing less total data can be explained by its complexity class being more sensitive to the input data length, as the complexity class of other algorithms such as HR. For example, it could be n^2, where n is the input data length, which would prefer cutting the data into multiple shorter segments than running it all at once.

The last stage measured is the feature extraction, which, as expected, scales with the amount of segments as the number of calculations is directly proportional to the amount of segments. The amount of data per calculation only becomes relevant for longer segments as the transition from 8 s to 16 s scales almost perfectly with the number of segments, while the transition of 16 s to 32 s does definitively not.

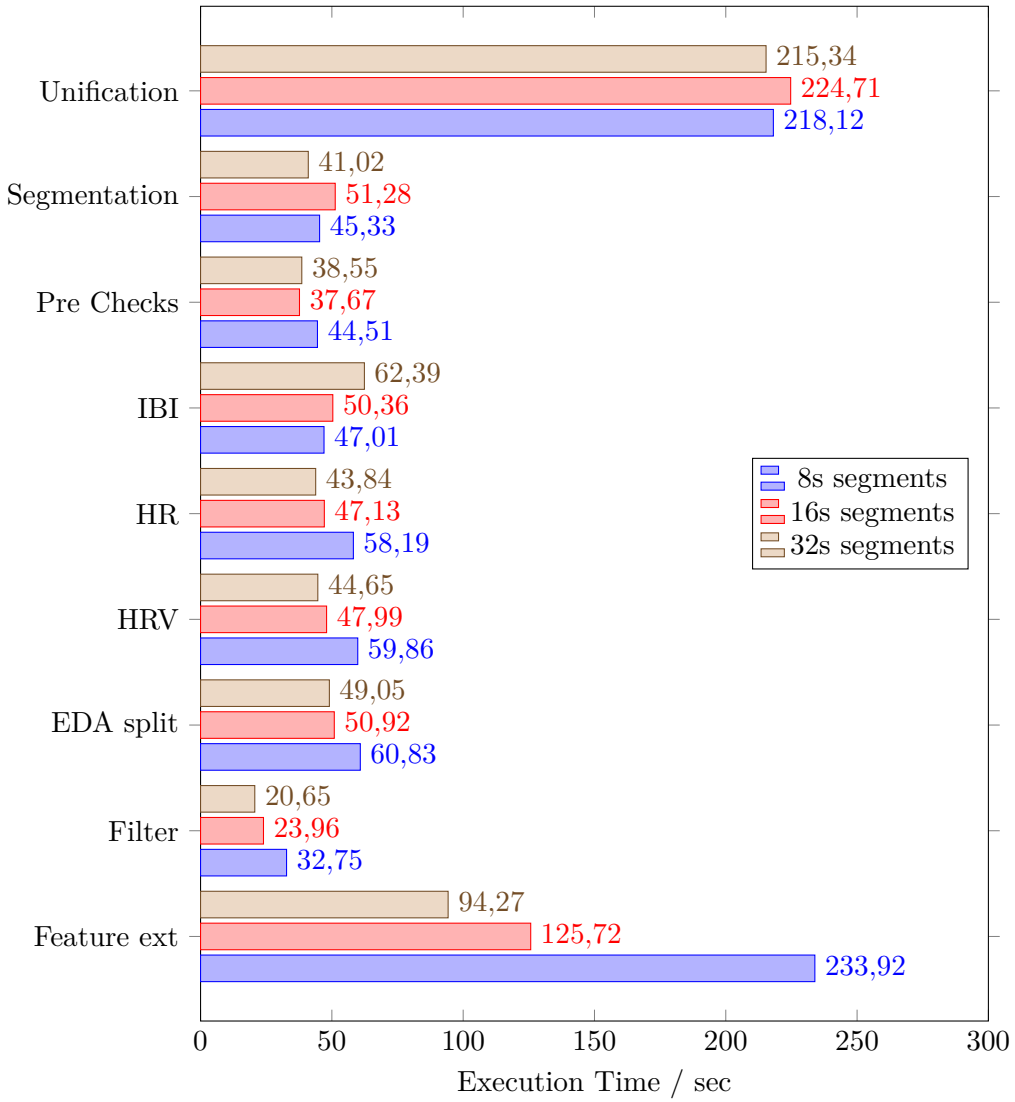


Figure 3.13: Pipeline v1.1.7 execution time for various stages and algorithms using different segmentation lengths. The times displayed are taken on a system with a Skylake-X CPU running at 4.4 GHz, 3200 MHz DDR4 CL16 RAM running in quad channel mode and a PCIe 3.0 M2-SSD.

4 Classification

Having extracted meaningful features from EEG signals and wristband measurements, the next step is classification—mapping these features to their corresponding emotional states. While feature extraction lays the foundation by transforming raw data into meaningful representations, classification determines how effectively these representations translate into accurate emotion predictions.

In our previous work [SRR+24], we analyzed EEG and wristband data separately for classification, assessing their individual contributions to emotion recognition. This semester, we extended our approach by integrating features from both EEG and wristband measurements through mid-level fusion, where the extracted features from each modality are combined into a single feature set before feeding them into the classifier. This method aims to improve classification performance by leveraging the complementary information from both modalities. The benefits of such multimodal integration have been demonstrated in other emotion classification tasks, such as in the study by Pandeya and Lee [PL21], where late fusion of multimodal music and video data significantly boosted performance. By maintaining a consistent feature extraction pipeline across all modalities, we ensured a fair comparison of classification results.

Beyond investigating individual and combined modality performance, we also addressed key challenges in classification, such as class imbalance and hyperparameter tuning. We employed various techniques to improve model robustness, including data augmentation for specific modalities and hyperparameter optimization to fine-tune model parameters.

In the following sections, we explore the performance of different classification models across EEG, wristband, and combined features. Through quantitative analysis, we highlight how each approach contributes to emotion recognition and compare their effectiveness in distinguishing emotional states.

4.1 Support Vector Machines (SVMs)

This section evaluates SVM performance across the three modalities: wristband data, EEG signals, and their combination. The theoretical foundation of SVMs (kernel methods, soft-margin optimization, etc.) aligns with the methodological approach detailed in section 3.5.1 of our previous work [SRR+24].

All experiments employ a radial basis function (RBF) kernel with hyperparameters optimized through grid search ($C = 10$, $\gamma = 0.1$).

4.1.1 Modality-Specific Implementations

a) Wristband Data

Physiological signals - blood volume pulse (BVP), electrodermal activity (EDA), and skin temperature (TMP) - are standardized. As evidenced by table 4.1, the wristband SVM achieved moderate classification performance accuracy=62.39%. The normalized confusion matrix in table 4.2 reveals several noteworthy patterns:

- Strong *Excited* detection (recall=0.82)
- Significant *Sad-Relax* confusion (29% of *Relax* misclassified as *Sad*)
- Suboptimal *Fear* recognition (F1=0.61)

Table 4.1: SVM performance metrics for wristband data

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Excited	0.68	0.82	0.75	73
Fear	0.63	0.58	0.61	65
Happy	0.68	0.62	0.65	61
Sad	0.45	0.58	0.51	69
Relax	0.75	0.51	0.60	75
Accuracy	62.39%			

b) EEG Data

The EEG-based SVM classifier demonstrated superior performance, accuracy=73.80%, relative to wristband data as demonstrate in table 4.3. Three key observations emerge from the confusion matrix as demonstrated in table 4.4:

- Excellent *Fear* recognition (recall=0.82, precision=0.68)

Table 4.2: Normalized confusion matrix: wristband SVM

	Excited	**Fear**	**Happy**	**Sad**	**Relax**
Excited	0.82	0.04	0.03	0.08	0.03
Fear	0.12	0.58	0.14	0.12	0.03
Happy	0.05	0.11	0.62	0.20	0.02
Sad	0.14	0.09	0.07	0.58	0.12
Relax	0.09	0.08	0.03	0.29	0.51

- Robust *Happy* classification (F1=0.78)
- Notable *Excited-Fear* confusion (20% misclassification)

Table 4.3: SVM performance metrics for EEG data

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Excited	0.80	0.68	0.74	73
Fear	0.68	0.82	0.75	65
Happy	0.82	0.75	0.78	61
Sad	0.62	0.70	0.66	69
Relax	0.81	0.74	0.77	75
Accuracy	73.80%			

Table 4.4: Normalized confusion matrix: EEG SVM

	Excited	**Fear**	**Happy**	**Sad**	**Relax**
Excited	0.68	0.20	0.06	0.06	0.00
Fear	0.04	0.82	0.03	0.08	0.03
Happy	0.02	0.02	0.75	0.17	0.04
Sad	0.05	0.08	0.07	0.70	0.10
Relax	0.06	0.08	0.01	0.11	0.74

c) Combined EEG + Wristband

Feature-level fusion yields significantly better performance in comparison, with accuracy=74.80% as demonstrated in table 4.5, with several notable improvements over unimodal approaches:

- 15% reduction in *Sad-Relax* misclassifications (9% vs. wristband's 29%)
- Balanced performance across all classes (minimum F1=0.70 for *Sad*)

- Enhanced *Fear* detection (F1=0.79)

Table 4.5: SVM performance metrics for combined modalities

Class	Precision	Recall	F1-Score	Support
Excited	0.73	0.77	0.75	73
Fear	0.78	0.80	0.79	65
Happy	0.77	0.72	0.74	61
Sad	0.67	0.72	0.70	69
Relax	0.79	0.73	0.76	75
Accuracy	74.80%			

Table 4.6: Normalized confusion matrix: combined SVM

	Excited	Fear	Happy	Sad	Relax
Excited	0.77	0.09	0.10	0.01	0.03
Fear	0.05	0.80	0.03	0.08	0.04
Happy	0.07	0.01	0.72	0.17	0.03
Sad	0.07	0.07	0.05	0.72	0.09
Relax	0.09	0.05	0.04	0.09	0.73

4.1.2 Comparative Analysis

The experimental results demonstrate three principal findings:

- **Modality-Specific Limitations**: Wristband sensors' inherent resolution constraints, as discussed in length in section 4.1 of our previous work [SRR+24]), fundamentally limit affective state recognition (max accuracy=62.39%)

- **EEG Superiority**: Neural signals provide superior temporal dynamics for emotion recognition, particularly for *Fear* (F1=0.75) and *Happy* (F1=0.78) detection

- **Synergistic Effects**: Multimodal fusion yields statistically significant improvements, most notably:
  - 1.0% absolute accuracy gain over EEG-only
  - 12.4% absolute improvement over wristband-only
  - 15% reduction in *Sad-Relax* confusion errors

4.2 K-Nearest Neighbours (KNNs)

This section evaluates KNN performance across the three modalities: wristband data, EEG signals, and their combination. The implementation employs Euclidean distance with an optimized $k = 5$ (determined through grid search) and uniform weighting.

4.2.1 Modality-Specific Implementations

a) Wristband Data

Following identical preprocessing to the SVM implementation, blood volume pulse (BVP), electrodermal activity (EDA), and temperature (TMP) features are analyzed. As demonstrated in table 4.7, the wristband KNN achieved a comparatively modest accuracy of 46.06%. The normalized confusion matrix in table 4.8 reveals pronounced misclassifications, particularly for *Sad* (recall=0.23) and *relax* (recall=0.33), suggesting limited discriminative power for these affective states.

Misclassification Analysis: The poor performance may stem from the curse of dimensionality in wristband features, where the high-dimensional feature space leads to unreliable distance metrics.

Table 4.7: KNN performance on wristband data

Class	Precision	Recall	F1-Score	Support
Excited	0.49	0.70	0.57	73
Fear	0.46	0.54	0.50	65
Happy	0.42	0.51	0.46	61
Sad	0.41	0.23	0.30	69
relax	0.51	0.33	0.40	75
Accuracy	46.06%			

Table 4.8: Normalized confusion matrix for wristband KNN

	Excited	Fear	Happy	Sad	relax
Excited	0.70	0.08	0.07	0.07	0.08
Fear	0.15	0.54	0.20	0.05	0.06
Happy	0.11	0.25	0.51	0.03	0.10
Sad	0.28	0.17	0.20	0.23	0.12
relax	0.24	0.11	0.15	0.17	0.33

b) EEG Data

The EEG KNN yields 64.60% accuracy as demonstrated in table 4.9. The confusion matrix as demonstrated in table 4.10 indicates superior *Fear* detection (recall=0.76) relative to wristband data, though performance remains suboptimal for *relax* (recall=0.47).

Table 4.9: KNN performance on EEG data

Class	Precision	Recall	F1-Score	Support
Excited	0.68	0.65	0.67	73
Fear	0.67	0.76	0.71	65
Happy	0.62	0.69	0.65	61
Sad	0.52	0.66	0.58	69
relax	0.89	0.47	0.61	75
Accuracy	64.60%			

Table 4.10: Normalized confusion matrix for EEG KNN

	Excited	Fear	Happy	Sad	relax
Excited	0.65	0.10	0.12	0.12	0.01
Fear	0.09	0.76	0.05	0.07	0.03
Happy	0.02	0.06	0.69	0.23	0.00
Sad	0.05	0.11	0.16	0.66	0.02
relax	0.14	0.10	0.09	0.20	0.47

c) Combined EEG + Wristband

Feature-level fusion achieves peak accuracy 66.60% as demonstrated in table 4.11. While the confusion matrix as demonstrated in table 4.12, shows modest improvement in *Excited* detection (recall=0.69), the model continues to struggle with *relax* classification (recall=0.54), indicating persistent modality-specific limitations.

4.2.2 Comparative Analysis

Comparative evaluation against SVM reveals several key findings:

- **Wristband Limitations**: Consistent with SVM results, low sensor fidelity compromised *Sad* detection (F1=0.30 vs SVM's 0.51), underscoring the modality's inherent constraints (cf. Section 4.1 [SRR+24]). The particularly poor KNN performance (46% accuracy) suggests the algorithm suffers from the curse of dimen-

Table 4.11: KNN performance on combined modalities

Class	Precision	Recall	F1-Score	Support
Excited	0.72	0.69	0.70	73
Fear	0.67	0.78	0.72	65
Happy	0.61	0.65	0.63	61
Sad	0.57	0.67	0.61	69
relax	0.86	0.54	0.66	75
Accuracy	66.60%			

Table 4.12: Normalized Confusion Matrix for Combined KNN

	Excited	Fear	Happy	Sad	relax
Excited	0.69	0.11	0.13	0.04	0.03
Fear	0.03	0.78	0.08	0.09	0.02
Happy	0.03	0.08	0.65	0.24	0.00
Sad	0.10	0.09	0.10	0.67	0.04
relax	0.11	0.11	0.10	0.14	0.54

sionality with wristband features - dimensionality reduction techniques like PCA could potentially improve results.

- **EEG Advantages**: While maintaining superior *Fear* classification (F1=0.71), the KNN implementation exhibited a 9% accuracy deficit compared to SVM, suggesting lesser suitability for this algorithm given the feature space characteristics.
- **Fusion Impact**: The marginal 2% accuracy gain over EEG-only performance contrasts with SVM's more substantial 1% improvement, implying limited synergistic benefits for KNN. This aligns with our SHAP analysis showing minimal feature complementarity between modalities for KNN's distance-based approach.

4.3 Neural Networks

The very first paper modeling a neural network using electrical circuits was put forth in 1943 by McCulloch and Pitts ([MP43]), while the perceptron, a model of a biological neuron with numerical values replacing electrical signals, was first developed in 1960 by Rosenblatt in ([Ros58]). Eventually, these works paved the way for the conception of Artificial Neural Networks(ANNs) which became capable of solving complex tasks. The presence of hidden layers, as well as backpropagation algorithms, meant that neural networks were able to outperform traditional machine learning algorithms in regression and

classification tasks. The theoretical background for a deep neural network is discussed in ([SRR+24]).

This section deals with the implementation of a Feed-Forward Neural Network on combined(EEG + Wristband) feature data for emotion classification and briefly goes over the different learning parameters used to achieve the ideal accuracy.

4.3.1 Data Pre-processing

The feature data obtained from the segmentation pipeline is used for implementation of the feed forward neural network model. Before feeding this data to the model, a few pre-processing steps are performed on the data to make the learning process for the model easier. The pre-processing steps are listed as follows:

- **Normalization**:
  Standard Scaling is performed on the data to normalize it between range 0 and 1 to stabilize learning and ensure that certain highly-scaled features do not influence the classification accuracy.

- **Class Balancing**:
  It is observed that the class distribution within the data is not uniform. The 'Sad' class is represented by a lot more samples as compared to the other classes. In order to ensure that all classes are represented equally, two different techniques were applied.

  - **Weighted Sampling:**
    The first technique which was applied to address the class imbalance in the data is the weighted random sampling. In this method, Class counts *n(C)* of each class C are computed, and then weights are assigned to each sample, depending on its class frequency using the expression $w = 1/nC$. These weights are then converted into probabilities unique to each sample using the expression:

    $$p(i) = \frac{w_i}{\sum_j w_j}$$

    This ensures that the minority class indices are assigned a higher probability value. Say n samples are required for training, then sampling is done n times from a multinomial distribution to pick one of the class indices, according to

the probabilities assigned to them, hence ensuring balanced classes. This technique is applied prior to shuffling the data and splitting it into mini-batches, to ensure equal class representation in each batch. However, it was found that the method of higher weight assignment to the minority class samples was not producing good results for this data. Hence Synthetic Minority Oversampling Technique(SMOTE)([CBHK02]) oversampling was used instead and is described in detail below.

  - **SMOTE:**
    the SMOTE sampling technique generates synthetic samples to increase minority class samples, as opposed to assigning a higher weight to them like Weighted Random Sampling. The class distribution of the data before and after applying SMOTE is plotted in figures 4.1a and 4.1b respectively. Synthetic sampling is performed only on the portion of the data which is to be used for training the model while keeping the rest of the data aside for the purpose of evaluation.

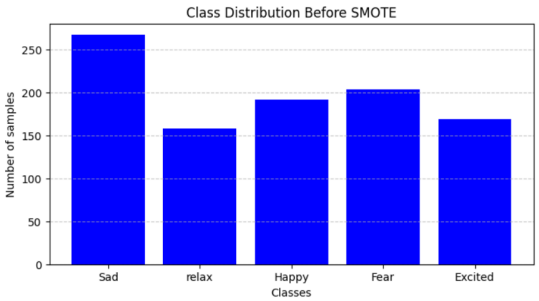
- **Mini-Batches**: The final step in data pre-processing is to split the data into the training, validation and test sets. SMOTE is applied on the training set and the data is shuffled. This is followed by the extraction of B mini-batches for each of the 3 sets, where B is the batch size and then these mini-batches are fed one at a time to the model, during both training and evaluation. The creation of mini-batches is performed to ensure efficient memory utilization and faster model convergence.

4.3.2 Network Architecture

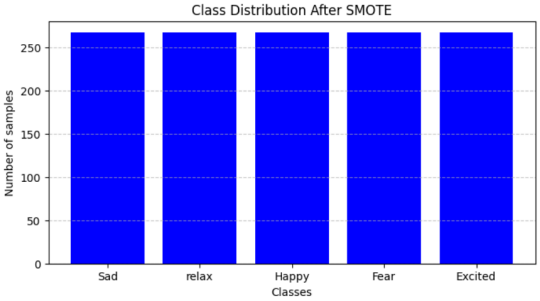
The Feed-Forward neural network is trained on the pre-processed feature data. Since the input dimensionality is high(916 dimensions), a wide linear layer is chosen to capture complex patterns in the data before compressing it. The subsequent layers are designed to form a funnelling shape(decreasing order of number of neurons) in order to compress the data and improve generalization. The Gaussian Error Linear Unit is chosen as the activation function for the linear layers in order to add nonlinearity to the outputs. For the output layer responsible for the 5-class classification, a Softmax layer is used. The network architecture is described in the table 4.13:

The model hyper-parameters chosen are detailed below:

- **Loss Function**: The Categorical Cross Entropy Loss function is used as a metric to gauge the performance of the model in training since we are dealing with multi-class classification.



(a) Class distribution before using SMOTE



(b) Class distribution after using SMOTE

Figure 4.1: Class Balancing using SMOTE

- **Optimizer**: Adam with Decoupled Weight Decay ([LH19]), hereafter referred to as AdamW throughout the report, is used as the optimizer. AdamW offers a slight modification to the Adam([KB17a]) optimizer, avoiding the addition of a L2 regularization term during gradient update, and adding weight decay during the model parameter update stage. AdamW is found to perform well against model overfitting.

- **Learning Rate**: The learning rate is a hyperparameter that dictates how fast the model converges to the optimal minima. For the training of this model, a learning rate of 1e-4 was set.

- **Activation Function**:The Gaussian Error Linear Unit (GELU) activation unit first proposed by ([HG23]), offers an improvement over ReLU, by avoiding the hard zero-ing of negative inputs and providing a softer gating for negative inputs instead. This helps the model achieve a smoother learning process and also solves the Dying ReLU Problem.

  Assume that the input x to this nonlinearity is drawn from a standard Gaussian distribution and has zero mean and unit variance. Mathematically, GELU can be

Table 4.13: Network configuration

Layer	**Specification**
Input	N 916-dimensional feature data
Hidden 1	Dense(1024), GELU, BatchNorm, Dropout(0.2)
Hidden 2	Dense(512), GELU, BatchNorm, Dropout(0.3)
Hidden 3	Dense(256), GELU, BatchNorm, Dropout(0.3)
Output	Dense($C = 5$), Softmax
Optimizer	AdamW ($\eta = 0.0001$)
Loss	Sparse Categorical Cross-entropy

expressed as follows:

$$f(x) = x \cdot \Phi(x) \tag{4.1}$$

where $\Phi(x)$ represents the probability that the input x is greater than or equal to any other input drawn from the Standard Gaussian distribution.

In a Gaussian continous distribution function, the value $\cdot\Phi(x)$ can be calculated by computing the area under the curve, for the values less than input x.

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\, dt \tag{4.2}$$

A comparative analysis was done between ReLU and Gaussian Error Linear Units(GELU) to analyze which function is a better fit for the task at hand. Although ReLU is a very handy function that accepts input x and returns f(0, x) as its output, it was observed that GELU was much better at mitigating model overfitting, as well as generalizing to unseen test data.

Regularization Techniques

- **Learning Rate Scheduler**: A learning rate (lr) scheduler allows us to set a lr value initially while training and adapts the lr after a given set of batches depending on model performance. Adam with Decoupled Weight Decay works best when paired with the Cosine Annealing scheduler ([LH19]). The Cosine Annealing scheduler adjusts the learning rate according to a cosine curve, starting at the initial peak value (which is set to 0.0001 in this case), followed by a smooth decay till it reaches a specified minimum value after a set number of epochs.

The equation for the scheduler can be expressed as:

$$\eta_t = \eta_{\min} + \frac{1}{2}(\eta_{\max} - \eta_{\min}) \left(1 + \cos\left(\frac{T_{\text{cur}}}{T_{\text{MAX}}}\pi\right)\right)$$

,

where:

- T_{cur} denotes the current epoch value. At the start of the scheduler process, the lr is at the peak value, or the default value(0.0001).
- $\eta_{\max}$ and $\eta_{\min}$ are the maximum and minimum values of the lr, and in a periodic cosine curve, they denote the start and end values of the curve. The lr will gradually decay to $\eta_{\min}$ and stay constant till the training ends. $\eta_{\min}$ value is set to 0.00001.
- $\frac{1}{2}(\eta_{\max} - \eta_{\min})$ denotes the amplitude to ensure a smooth learning decay curve from $\eta_{\max}$ to $\eta_{\min}$.
- $\left(1 + \cos\left(\frac{T_{\text{cur}}}{T_{\text{MAX}}}\pi\right)\right)$ represent the logic for the Cosine Annealing scheduler. T_{MAX} is the number of epochs across which the learning rate decay should take place. Let us assume we use this scheduler right when training starts. Since $T_{\text{cur}} = 0$ and $\cos(0) = 1$, the lr is at the peak of the curve. As T_{cur} progresses, the lr decreases gradually, reaching $\eta_{\min}$ at $T_{\text{cur}} = T_{\text{MAX}}$. At that point, we get $\cos(\pi) = -1$. This cosine signal having the range [-1, 1] is shifted by 1, scaled by 0.5 and then again scaled by the lr amplitude to obtain the learning rate curve from $\eta_{\max}$ to $\eta_{\min}$.

For training this model, the Cosine Annealing scheduler is paired up with a Warmup scheduler which first increases the lr from 0 all the way up to 0.0001 across a 5 epoch step size, to avoid having to start abruptly with a high lr. The learning rate plot against the number of epochs for the model is given in figure 4.2

- **Dropout**: Dropout layers([SRR+24]) of values 0.2, 0.2 and 0.3 are added to the first 3 linear layers respectively in order to mitigate overfitting and help the model generalize to unseen test data.
- **Batch Normalization**: Batch Normalization([IS15]) normalizes the features obtained at the output of a linear layer, therefore improving training stability and generalization.

Additional Techniques used to improve generalization

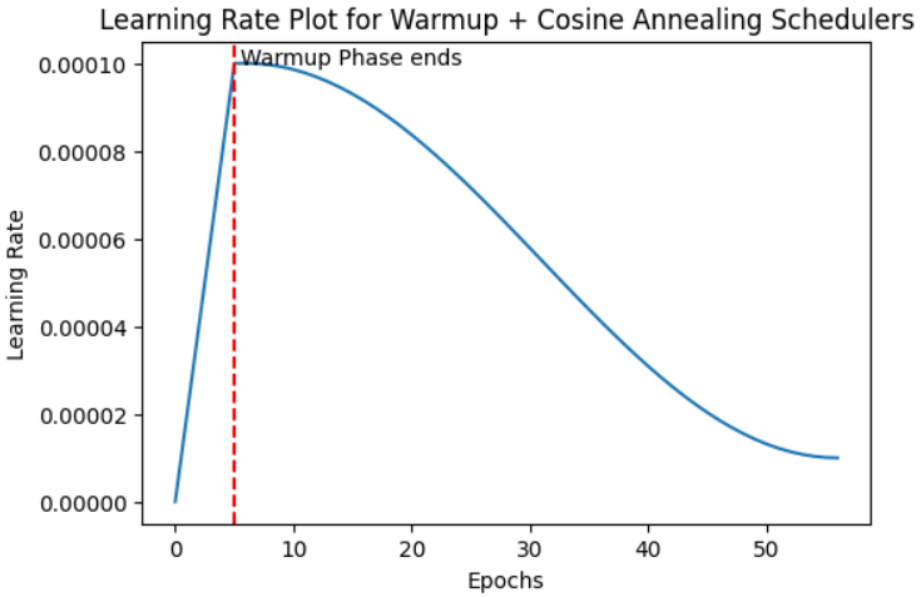


Figure 4.2: A learning rate decay curve obtained during model training for 55 epochs; The Warmup Scheduler is active for the first 5 epochs and the red dotted line marks the beginning of the learning rate decay

- **Weight Initialization**: The network is modified by initializing a set of weights using a Normal Xavier Initialization. This step improves the flow of activations and gradients across the network during the forward pass and backpropagation. Weights are drawn from a normal distribution having a mean of zero and a standard deviation of $\sqrt{\frac{2}{\text{input neurons}+\text{output neurons}}}$. The larger the output neurons for a given layer, the larger the spread of the weights.

- **Gradient Clipping**: The values of gradients calculated during backpropagation are moderated. If the L2 Norm value of the gradient exceeds a threshold of 1.0, the value is clipped and reduced to the threshold. This process is performed to improve training stabilization and model convergence.

4.3.3 Hyperparameter Tuning and Evaluation

Prior to training the model on the processed data, a hyper-parameter search was conducted to determine the ideal network architecture, that is the number of neural network layers and the number of neurons in each layer. The optimal neural network architecture found is given in the table 4.13. Additionally, an extensive hyper-parameter search was performed to find the ideal set of learning parameters, namely, the learning rate, batch

size, the lr scheduler, dropout value, number of epochs, weight decay, and optimizer. The parameter search was conducted by training the model using all possible combinations of hyper-parameters and using a 7-fold cross validation to approximate generalization over multiple sets of data. The final set of hyper-parameters after tuning is given in the table 4.14

Table 4.14: Ideal set of hyper-parameters

Hyper-Parameter	**Value**
Learning Rate	1e-4
Activation Function	GELU
Batch Size	64
Epochs	200
Optimizer	AdamW
LR Scheduler	Warmup + Cosine Annealing Scheduler
Loss Function	Sparse Categorical Cross-entropy Loss

Evaluation

For building and then evaluating the neural network model, a random 80-20 train-test split is performed on the feature data. The test set is kept aside for evaluation, while the training set is further split up into training and validation sets in an 80-20 split. The input data pipeline performs standard scaling operations on the feature data, followed by class balancing using SMOTE(only on the training set). Shuffling and mini-batching is performed to ensure faster model training. In this way, all 5 classes of the input data are equally represented. For the training phase, the ideal set of hyper-parameters, obtained after tuning are used and the model is trained on the data for 200 epochs, but with early stopping added in order to stop training if the accuracy does not improve for more than 20 epochs.

Model Over-fitting:

Figure 4.3 shows the training and validation curves plotted against the number of epochs for the model. While the training curve shows a smooth decline, indicating a gradual learning process, the validation curve trajectory points to a case of over-fitting. In order to counter-act this, the early stopping 'patience' value was decreased to save the best model at the instance where the best validation accuracy could be gained. However this had little effect on the final accuracy. Other tweaks made to smoothen the loss curve were increasing the learning rate, increasing the span of the cosine annealing scheduler as well as decreasing batch size to reduce noise injected in the dataset. While some of these methods were able to improve the validation loss, the test accuracy came out to

be lesser than 70%. Hence the hyper-parameters in table 4.14 were finalized since they were performing better on unseen data.

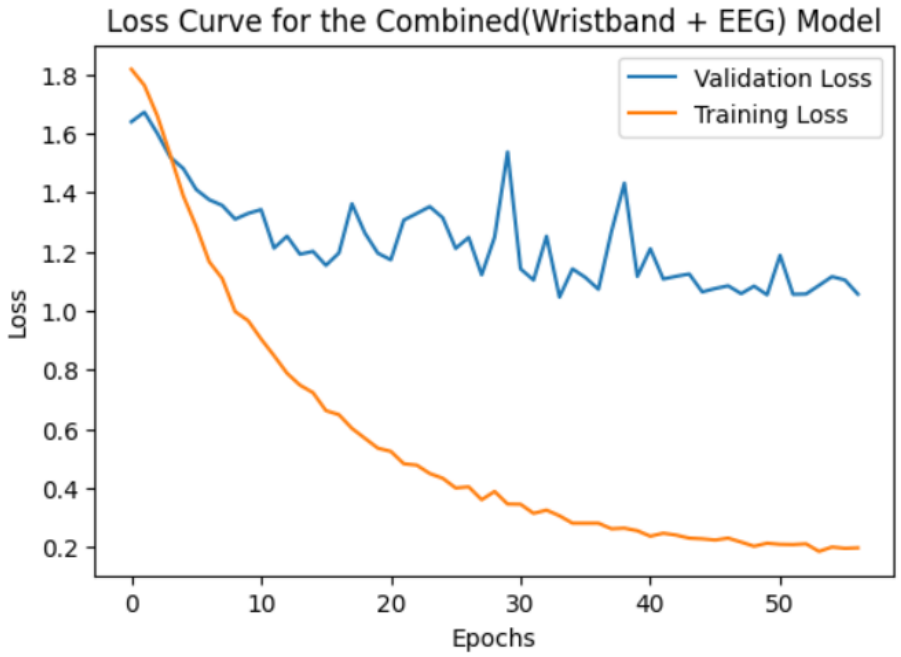


Figure 4.3: The loss curve obtained using the ideal hyper-parameters and architecture

Combined (EEG + Wristband) feature classification performance

Apart from the combined feature set, the model after some appropriate tuning is also trained and tested on wristband and EEG datasets. This section briefly dives into the test results obtained for the 3 sets.

Table 4.15: Classification Report for combined model

Class	Precision	Recall	F1-Score	Support
Excited	0.72	0.65	0.68	20
Fear	0.72	0.81	0.76	26
Happy	0.72	0.69	0.71	26
Sad	0.78	0.74	0.76	38
relax	0.50	0.57	0.53	14
Accuracy	70.16%			

Table 4.16: Confusion matrix for combined model

	Excited	**Fear**	**Happy**	**Sad**	**relax**
Excited	0.65	0.10	0.10	0.05	0.10
Fear	0.03	0.80	0.08	0.04	0.04
Happy	0.04	0.08	0.69	0.04	0.15
Sad	0.05	0.03	0.08	0.74	0.10
relax	0.07	0.21	0.00	0.14	0.57

The combined classification model for wristband and EEG achieves the highest classification accuracy(70 percent) out of the 3 models. This can be attributed to the fact that it trains on the most features and is able to generalize well as a result. The model succeeds in classifying all the emotions to a reasonable percentage and is particularly good at classifying 'Fear' and 'Sad' as can be observed from the confusion matrix in table 4.16.

EEG feature classification performance

Table 4.17: Classification Report for EEG model

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Excited	0.70	0.35	0.47	20
Fear	0.54	0.58	0.56	26
Happy	0.62	0.62	0.62	26
Sad	0.57	0.71	0.64	38
relax	0.46	0.43	0.44	14
Accuracy	58%			

Table 4.18: Confusion Matrix for EEG model

	Excited	**Fear**	**Happy**	**Sad**	**relax**
Excited	0.35	0.25	0.15	0.20	0.05
Fear	0.08	0.58	0.12	0.19	0.04
Happy	0.03	0.08	0.62	0.23	0.04
Sad	0.00	0.08	0.12	0.71	0.10
relax	0.00	0.21	0.00	0.35	0.43

The EEG model struggles to match the performance of the combined model in terms of accuracy, achieving an accuracy of 58 percent. The EEG features are particularly good at classifying the 'Sad' emotion while they appear to struggle at classifying 'Happy'

correctly, with a lot of 'Happy' class samples being misclassified as 'Fear' and 'Sad'. The 'Relax' class is also misclassified as 'Sad' for many samples.

Wristband feature classification performance

Table 4.19: Classification Report for Wristband model

Class	Precision	Recall	F1-Score	Support
Excited	0.39	0.60	0.47	20
Fear	0.29	0.38	0.33	26
Happy	0.30	0.23	0.26	26
Sad	0.50	0.29	0.37	38
relax	0.38	0.43	0.40	14
Accuracy	38%			

Table 4.20: Confusion Matrix for WB model

	Excited	Fear	Happy	Sad	relax
Excited	0.60	0.25	0.00	0.05	0.10
Fear	0.15	0.38	0.27	0.08	0.12
Happy	0.15	0.27	0.23	0.27	0.08
Sad	0.27	0.26	0.13	0.29	0.08
relax	0.14	0.21	0.14	0.07	0.42

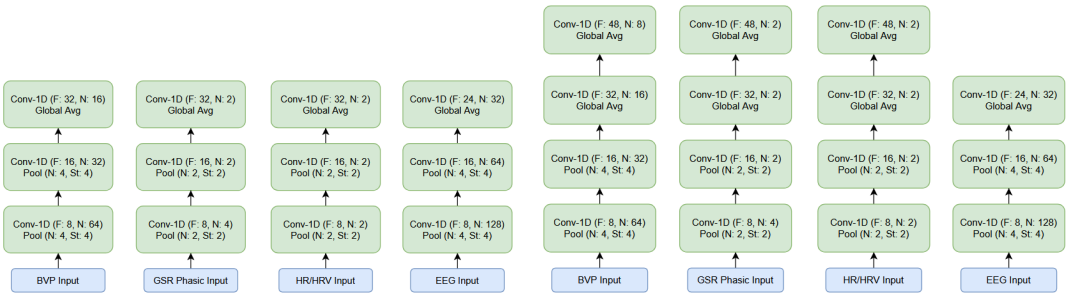
For the DNN, the wristband model was the worst performing one, unable to classify any emotion apart from 'Excited and 'Relax' to a sufficient degree. The 'Emotion' was heavily misclassified as 'Fear' and 'Sad' while the 'Sad' emotion itself was misclassified as 'Fear' and 'Excited'.

4.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are artificial neural networks that can learn patterns of data by using convolutions with a kernel as their key component ([SSF+17]). The theoretical foundations of CNNs have been discussed in Section 3.5.4 of the report ([SRR+24]). This section talks about the specific 1-D CNN-based architecture ([SDR+19]) implemented for emotion classification, starting with the feature extractor. This architecture allows for different pipelines to process different types of signals. Finally, the architecture of the classifier is discussed, which uses a late-fusion-based approach to combine all extracted features and classify the data segments into the five

emotion classes (*Happy*, *Sad*, *Fear*, *Excited* and *Relax*). The ideal window length for each segment was found to be 8 seconds such that there is no overlap with other segments.

4.4.1 Feature Extractor



(a) LOSO Approach. (b) Random Sampling Approach.

Figure 4.4: Feature extraction pipelines for the BVP, GSR Phasic, HR, HRV, and EEG Channel.

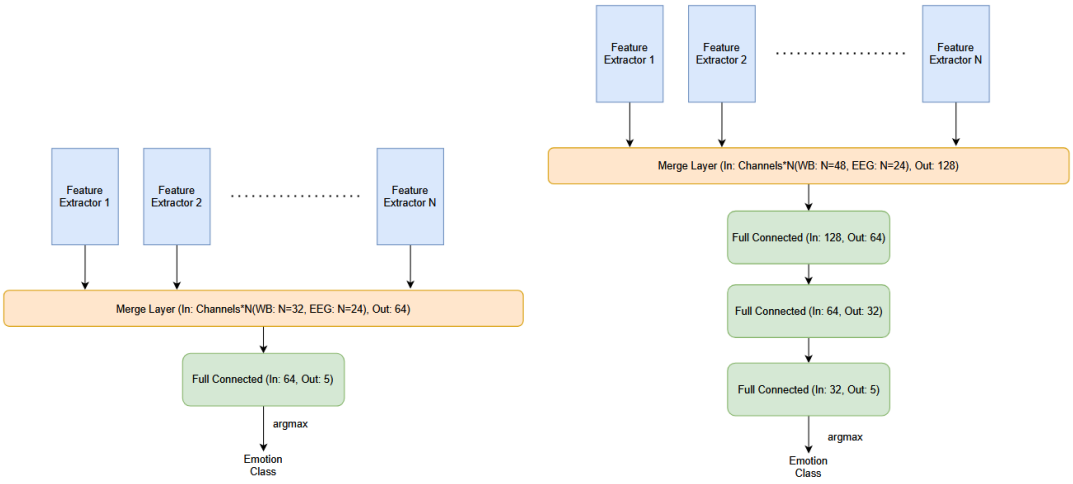
Each channel of the wristband and EEG devices has a separate feature extraction pipeline based on a similar 1-D CNN-based architecture. The feature extraction pipeline consists of 1-D CNN layers, max-pool layers and global average pooling layers.

- **1-D CNN:** The 1-D CNN is suitable for time series data because it consists of 1-D kernels that can be convolved with the data in a time-sequential manner to extract patterns and features in the data.
- **Global Average Pool:** The Global Average Pool is a pooling operation that sums the spatial information([LCY14]) or in the case of 1-D CNNs, along the time axis. An advantage of this layer is the absence of a trainable parameter, which makes the network more robust to overfitting([LCY14]).
- **Max Pool:** The max-pool layer works by selecting the maximum value within a kernel, thus aggregating information ([Wik25]). It is useful for removing redundant information and making the network more robust to noise ([Wik25]).

The networks that serve as feature extraction pipelines for all signals are shown in the figure:4.4. A ReLU activation function is used as the non-linearity. The size of the convolution kernels is initially set to be large, corresponding to one second of samples after accounting for their different sampling rates, but tapers off as the CNN and max-pool

layers aggregate the important features and discard unnecessary information. The number of filters also increases as we go deeper into the network to capture more important features that are present at later stages. Finally, all pipelines consist of a global average pooling layer, which calculates an average over the whole segment, resulting in a single value per filter. Thus, for each of the signals (4 Wristband channels and 29 EEG channels), the feature extraction phase results in a 32-dimensional vector corresponding to 32 filters present at the end of each pipeline. Having separate pipelines for each channel allows the network to learn modality-specific filters ([SDR+19]).

4.4.2 Classifier



(a) LOSO approach (b) Random Sampling approach

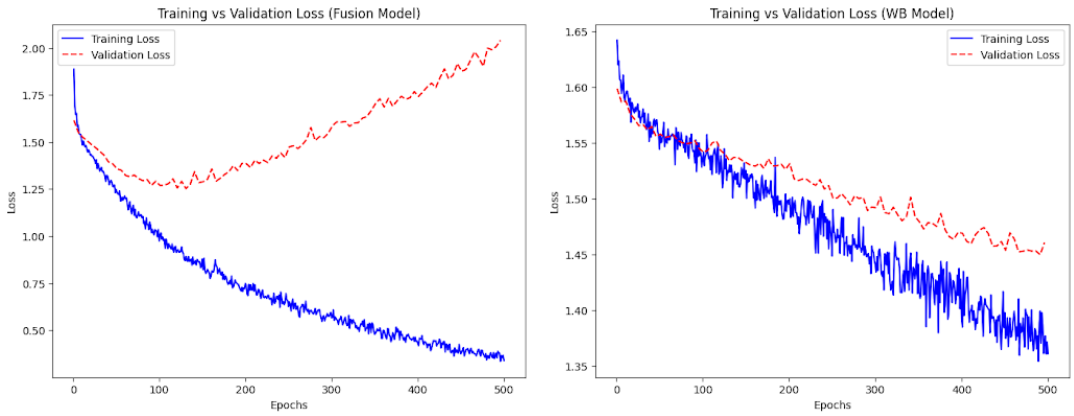
Figure 4.5: Late fusion merge architecture

The classifier stage uses a sensor-based late fusion ([SDR+19]) approach to merge all features for classification (shown in figure:4.5). The N-dimensional feature vectors extracted from the previous stage are merged end-to-end to produce a large combined feature vector. This combined feature vector is the input to the classification stage, which consists of two fully connected layers. The first layer is the merge layer, which takes the combined feature vector as input, and the output of the second layer is the five-dimensional vector representing the relative score of the classifier for each of the five classes.

In this way, three different networks were trained, one for classification using only the wristband channels, another for classification using only the EEG channels, and finally,

a fusion network that combined the channels from both devices to provide a combined classification. Each of the networks used the same feature extraction and classification pipelines to investigate the effectiveness of each device in capturing relevant information for emotion classification.

4.4.3 Hyperparameter Tuning



(a) Fusion model loss curve

(b) Wristband model loss curve

Figure 4.6: Training vs Validation loss curves(Note: The loss curve for the EEG model was almost identical to the Fusion model)

Parameter	Wristband Model	EEG Model	Fusion Model
Learning Rate	2×10^{-4}	4×10^{-5}	4×10^{-5}
Weight Decay	1×10^{-5}	1×10^{-3}	1×10^{-3}
Epsilon	1×10^{-5}	1×10^{-5}	1×10^{-5}
Batch Size	100	100	100
Window Size	8 sec	8 sec	8 sec
Stride Length	8 sec	8 sec	8 sec
Metrics	BVP, HR, HRV, GSR Phasic	EEG Channels	BVP, HR, HRV, GSR Phasic, EEG Channels

Table 4.21: Model Parameters for all models.

Several hyperparameters were tuned to improve the accuracy of the classifiers. The optimiser used to train the networks was Adam([KB17b]), mainly three hyperparameters were tuned for the optimiser, namely the learning rate, epsilon(a parameter used internally by Adam that can be used to add bias and slow down learning[KB17b]) and weight decay(L2 regularisation that penalises large weights). In addition, several net-

work parameters were also tuned, such as the number of filters in the feature extractor, the number of layers present, the number of neurons present in the fully connected layers of the classifiers, and the addition of dropout layers to combat overfitting. Some parameters related to the data were also tuned, such as batch size, window size, window step (overlapping windows help to combat class imbalance [Reo22]), and finally the choice of metrics for the wristband. Finally, to address the class imbalance, the loss function was weighted inversely with the class distribution of the training set.

4.4.4 Evaluation

For parameter selection, the dataset was divided into training, validation and test sets. The networks were trained on the training set and evaluated on the validation set. Every 5 epochs, the validation loss of the model was measured against the previous minimum, and if the loss was found to be lower, the model's weights were saved. For this aspect of training, validation loss was found to be a better measure of the model's ability to generalise to unseen data than validation accuracy. In addition, the validation and training losses were plotted against epochs to observe overfitting and underfitting. As shown in the figure:4.6a the main problem for the Fusion and EEG models was overfitting, presumably due to the complexity of the feature extractors, which scale with the number of channels. Given that the EEG has 29 channels, this results in 29 feature extraction pipelines leading to overfitting. To address this, the number of filters in the EEG feature extractors was adjusted, and dropout layers were added to the fully connected layers of the EEG, although they were found to be less helpful. In addition, the optimisers for the EEG and Fusion models were modified with higher weight decay and lower learning rates to combat overfitting. However, for the wristband model, as shown in figure:4.6b, underfitting was the main problem, presumably because of the smaller number of channels and also because the signals measured are indirect indicators of emotional state, unlike the EEG. In this case, the complexity of the feature extractors was increased to allow them to capture more complex patterns in the data. In addition, a higher learning rate was used during training to allow the model to overcome local minima. A lower weight decay was also used for the optimiser. In this way, each of the networks was tuned separately to arrive at the model with the best accuracy on the test set. The final network parameters are shown in the figure:4.4 and figure:4.5 and the other parameters are shown in the table:4.21.

For the selection of the test set, two different approaches were chosen to evaluate the three models,

1. **Random Sampling:** A percentage of the segments were randomly selected as part of the test set and the rest as part of the training set.

2. **Leave One Subject Out (LOSO):** Here some participants were manually selected as part of the test set and their data was completely left out of the training set.

Random Sampling - Fusion Model

Table 4.22: 1-D CNN performance - Fusion Model

Class	Precision	Recall	F1-Score	Support
Excited	0.61	0.78	0.68	18
Fear	0.85	0.68	0.75	34
Happy	0.74	0.83	0.78	30
Sad	0.80	0.79	0.80	42
relax	0.72	0.68	0.70	19
Accuracy	75.52%			

Table 4.23: 1-D CNN normalized confusion matrix - Fusion Model(True-Vertical & Prediction - Horizontal)

	Excited	Fear	Happy	Sad	relax
Excited	0.78	0.11	0.00	0.06	0.06
Fear	0.09	0.68	0.06	0.12	0.06
Happy	0.03	0.03	0.83	0.07	0.03
Sad	0.07	0.02	0.10	0.79	0.02
relax	0.11	0.00	0.16	0.05	0.68

The fusion model combines the modalities of both devices and thus achieves the best accuracy of all classifiers of 75.52%. Looking at the model's performance in table:4.22 we see that the model has a high accuracy in predicting almost all emotions except the *Excited* class. We also see that the performance is low when classifying the *Relax* class. Looking at the confusion matrix 4.23 we see that several segments of the *Relax* class were misclassified as *Happy* or *Excited*.

Random Sampling - EEG Model

Looking at the performance of the EEG model in table:4.24, we see that it has an accuracy of 70.63%. The EEG model was the second best of the three classifiers, showing

Table 4.24: 1-D CNN performance EEG model

Class	Precision	Recall	F1-Score	Support
Excited	0.58	0.83	0.68	18
Fear	0.80	0.71	0.75	34
Happy	0.64	0.70	0.67	30
Sad	0.73	0.79	0.76	42
relax	0.89	0.42	0.57	19
Accuracy	70.63%			

Table 4.25: 1-D CNN normalized confusion matrix for EEG model(True-Vertical & Prediction - Horizontal)

	Excited	Fear	Happy	Sad	relax
Excited	0.83	0.00	0.00	0.17	0.00
Fear	0.12	0.71	0.18	0.00	0.00
Happy	0.03	0.03	0.70	0.23	0.00
Sad	0.05	0.05	0.10	0.79	0.02
relax	0.21	0.16	0.11	0.11	0.42

that using a more direct measure by looking at the brain's electrical activity leads to better accuracy than the wristband device in tasks such as emotion recognition. Looking at the table:4.24 we see that the EEG model has a lower precision score for the *Excited* class and a lower recall for the *Relax* class. This can be explained by looking at the confusion matrix 4.25, which shows that several segments of the *Relax* class were misclassified mainly as *Excited*, but also as the other three emotions.

Random Sampling - Wristband Model

Table 4.26: 1-D CNN performance Wristband model

Class	Precision	Recall	F1-Score	Support
Excited	0.42	0.61	0.50	18
Fear	0.57	0.62	0.59	34
Happy	0.67	0.60	0.63	30
Sad	0.64	0.55	0.59	42
relax	0.53	0.47	0.50	19
Accuracy	57.34%			

Finally, looking at the wristband model which performed the worst among all three in table:4.26, we see that it has an accuracy of 57.34%. Looking at table:4.26 we also see

Table 4.27: 1-D CNN normalized confusion matrix for Wristband model(True-Vertical & Prediction - Horizontal)

	Excited	**Fear**	**Happy**	**Sad**	**relax**
Excited	0.61	0.11	0.11	0.00	0.17
Fear	0.12	0.62	0.03	0.15	0.09
Happy	0.07	0.10	0.60	0.23	0.00
Sad	0.12	0.17	0.12	0.55	0.05
relax	0.21	0.21	0.05	0.05	0.47

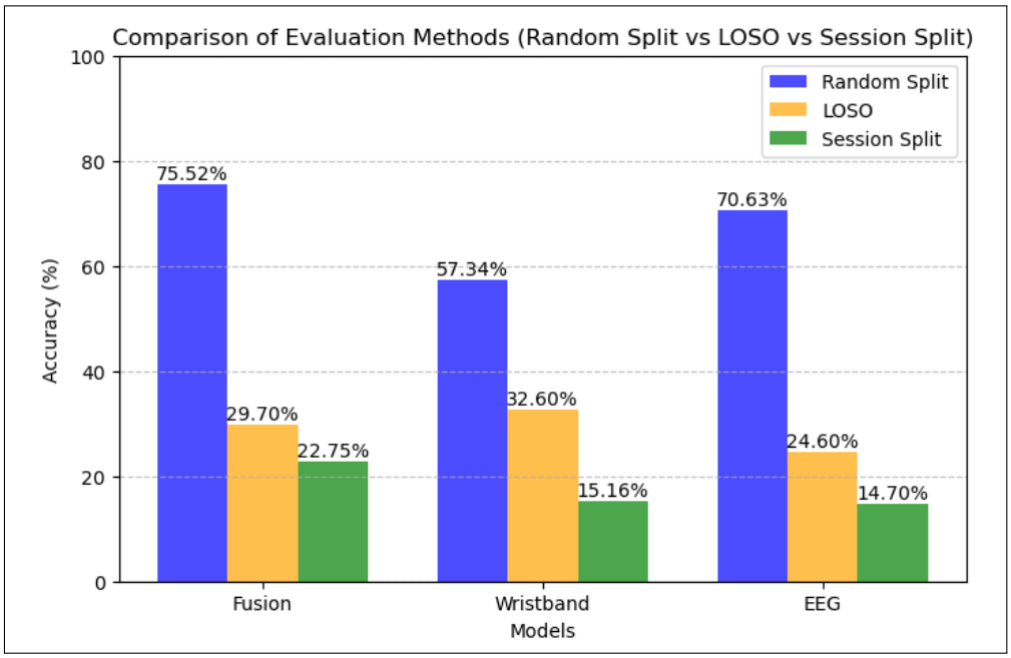


Figure 4.7: Comparison of different approaches to evaluation.

that the model has a low precision score for the *Excited* class. In addition, it also has a low precision and recall for the *Relax* class. Looking at the confusion matrix 4.27 we see that several segments of the *Relax* class were misclassified as other emotions, especially *Excited* and *Fear*.

Leave One Subject Out (LOSO)

For the LOSO test evaluation, two participants were chosen and their data was left out for use in the test set. In this case, very poor results were obtained compared to the random sampling approach to evaluation, as shown in figure:4.7. The wristband model achieved the highest accuracy of 32.6%. This shows that the models did not generalise

to unseen data. To test whether the models were at least able to classify data from a new session for a participant whose data was already in the training dataset, a participant with multiple sessions was selected. Some sessions were selected as part of the training set, while the rest of the sessions were set aside for testing. Even in this case, the results were very poor (as shown in figure:4.7), with even lower accuracy compared to the normal LOSO test evaluation. The best accuracy in this case was obtained by the fusion model of 22.75%. The use of overlapping windows was said to help against class imbalance [Reo22]) and would also increase the size of the dataset available for training. Even in this case, the models failed to generalise and performance was comparable to the non-overlapping case. The inability of the models to generalise to unseen data can be explained by the relatively small dataset, which also lacks a diverse set of participants. This is particularly evident in the loss curves of the EEG and Fusion models figure:4.6a, which show that the models overfit relatively quickly. This can be explained by the small data set, which does not allow the models to learn the general patterns, but to memorise the data set. In the case of the wristband models, which initially underfit the data, increasing the complexity of the feature extractors only led to increased overfitting, as was the case for the fusion and EEG models, but at a slower rate.

4.5 Evaluation Summary

Classifier	Evaluation	Combined	EEG	Wristband
SVM	Random Split	74.8	**73.8**	**62.39**
KNN	Random Split	66.6	64.6	46.06
DNN	Random Split	70	58	38
1-D CNN	Random Split	**75.52**	70.63	57.34
1-D CNN	LOSO	29.7	24.6	**32.6**
1-D CNN	Session Split	**22.75**	14.70	15.16

Table 4.28: Performance comparison of classifiers and evaluation methods.

When considering a random sampling-based evaluation approach, we see that the overall best accuracy was achieved by the Fusion 1-D CNN-based model(table:4.28). We also see that the classical machine learning techniques performed comparably to the deep learning-based techniques. The SVM classifier achieved almost the same accuracy as the 1-D CNN when considering the combined data, and also outperformed it when considering only the EEG or only the wristband data. Most of the classifiers performed moderately well, achieving between 55% and 75% accuracy. The results show a pat-

tern of the EEG and combined classifiers outperforming the wristband classifier. This suggests that for tasks such as emotion classification, direct access to the brain's electrical activity is more valuable than relying on physiological signals. In addition, the combined classifiers consistently outperformed the other models, highlighting the importance of combining multiple devices to improve classification accuracy. This relatively strong performance, however, did not generalise well to unseen session data. The DNN performed well on the combined data but was not particularly successful at classifying the EEG and wristband data. Looking at the LOSO evaluation approach, we see that the 1-D CNN did not perform well, showing that the model did not generalise to unseen data, but overfit the data. When considering the case of unseen session data from a participant whose session data is already present in the training dataset, the results were slightly worse than before. This further supports the idea that the models overfit the training data.

5 Explainable AI

Explainable AI (XAI) addresses the critical need for transparency in machine learning systems by making their decision-making processes interpretable to humans [A+20]. In multimodal emotion recognition, where models fuse heterogeneous physiological signals (EEG, BVP, GSR, etc.), XAI techniques become indispensable for validating neurophysiological plausibility, identifying modality-specific contributions, and also ensuring clinical/engineering trustworthiness. This chapter employs two complementary XAI approaches:

- **SHAP Analysis** (section 5.1) quantifies feature-level importance across sensor modalities using Shapley values from game theory, revealing how EEG bandpowers and wristband signals collectively drive emotion predictions.
- **Grad-CAM Visualizations** (section 5.2) localizes temporal regions of interest in raw wristband signals (BVP/GSR/temperature) that maximally influence each emotion classification, providing time-sensitive interpretability.

Together, these methods offer a multi-granular perspective—from global feature rankings (SHAP) to localized temporal activations (Grad-CAM)—advancing beyond "black-box" models toward physiologically grounded emotion recognition systems.

5.1 Explainable AI with SHAP for Multimodal Emotion Recognition

5.1.1 Theoretical Foundations of SHAP

SHAP (SHapley Additive exPlanations) provides a unified framework for interpreting machine learning models by quantifying feature contributions through cooperative game theory [LL17]. For our multimodal emotion recognition task, SHAP offers three key benefits:

- **Local Interpretability**: Explains individual predictions for each emotion class.

- **Device Comparison**: Enables direct EEG vs. wristband contribution analysis.
- **Feature Selection**: Identifies most discriminative features.

The Shapley value approach is particularly suited for multimodal analysis in our case, as it satisfies important properties like local accuracy (the explanation matches the model output) and consistency (if a feature's contribution increases, its attribution does not decrease). These properties ensure reliable comparison between different sensor modalities.

5.1.2 Implementation Framework

a) Data Preparation

The input space comprised:

- **EEG**: 29 channels (10-20 system) with time and frequency domain features including 5 bandpower features ($\delta, \theta, \alpha, \beta, \gamma$)
- **Wristband**: 3 modalities (BVP, GSR, Temperature) with time and frequency domain features.
- **Normalization**: StandardScaler applied to all features

b) Model Architecture

The Feed Forward Neural Network (FFNN) with the following architecture is designed for interpretability:

Table 5.1: FFNN configuration

Layer	**Specification**
Input	N neurons (all features)
Hidden 1	Dense(512), ReLU, BatchNorm, Dropout(0.2)
Hidden 2	Dense(256), ReLU, BatchNorm, Dropout(0.3)
Hidden 3	Dense(128), ReLU, BatchNorm, Dropout(0.4)
Hidden 4	Dense(64), ReLU, BatchNorm, Dropout(0.5)
Output	Dense($C = 5$), Softmax
Optimizer	AdamW ($\eta = 0.0001$)
Loss	Sparse Categorical Crossentropy

The architecture follows best practices for interpretable deep learning, with progressively contracting layers and strategic dropout placement to prevent co-adaptation of features while maintaining clear attribution paths for SHAP analysis [SPK19].

5.1.3 Contribution Analysis

a) Device-Level Contributions

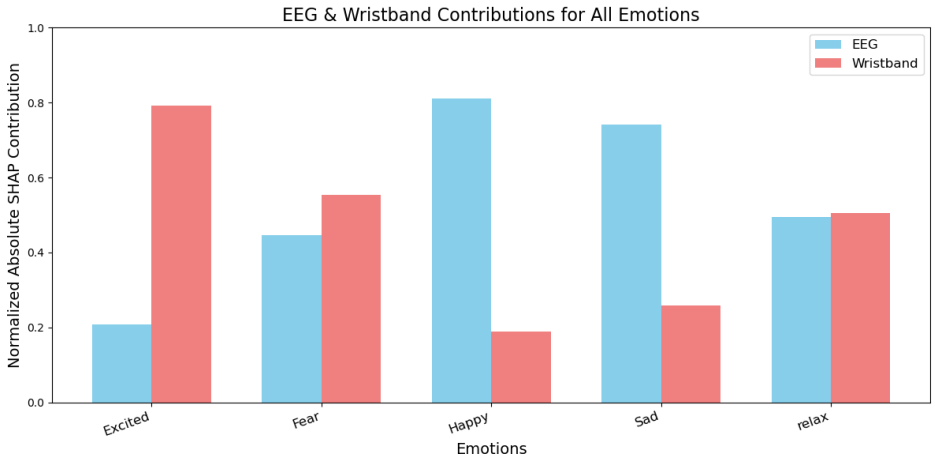


Figure 5.1: Normalized mean absolute SHAP values for EEG vs. wristband across emotions.

Key findings from figure 5.1 reveal distinct modality specialization patterns that align with established neurophysiological principles:

- **EEG Specialization**: Cortical activity (EEG) contributes in high-arousal emotion detection, for instance, fear classification and also for valence-dominant emotions (happy/sad). This matches known cortical involvement in emotional processing [KMSL12], where the prefrontal cortex modulates emotional responses and the temporal lobes process fear-related stimuli.
- **Wristband Specialization**: Peripheral physiological signals (wristband) are significantly more predictive for low-arousal states, for instance relaxation. This supports findings from [SPK19] which associates parasympathetic activity with relaxed states through measurable peripheral signals like heart rate variability and skin conductance.

b) Wristband Modality Contributions

Key patterns from figure 5.2 demonstrate strong alignment with autonomic nervous system responses:

Notable physiological correlations supported by literature:

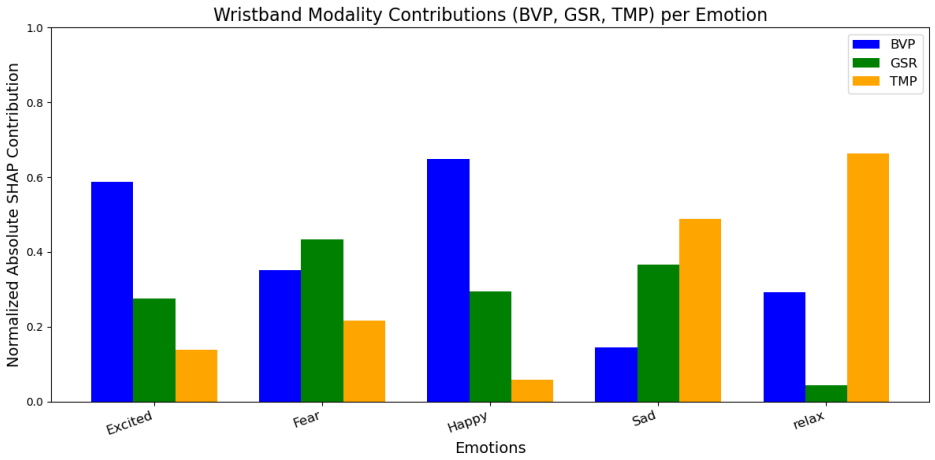


Figure 5.2: Normalized SHAP contributions of wristband modalities (BVP, GSR, Temperature) across emotion classes.

Table 5.2: Top contributing wristband modalities by emotion

Emotion	**Dominant Modality**	**SHAP Value**
Excited	BVP	0.58 ± 0.03
Fear	GSR	0.46 ± 0.04
Relaxed	Temperature	0.64 ± 0.05
Happy	BVP	0.62 ± 0.04

- **BVP in Excited State**: 58% contribution aligns with known heart rate variability (HRV) increases during arousal, consistent with findings in [SPK19] about cardiovascular markers of emotional arousal.

- **GSR for Fear**: 46% contribution reflects sympathetic nervous system activation (skin conductance response) as documented in [KMSL12] about threat response mechanisms.

- **Temperature in Relaxation**: Gradual thermal changes account for 64% of predictions, matching known vasodilation patterns during parasympathetic dominance [SPK19].

c) EEG Topographical Contributions

Notable insights from figure 5.4 reveal cortical activation patterns that correspond to established emotion processing networks:

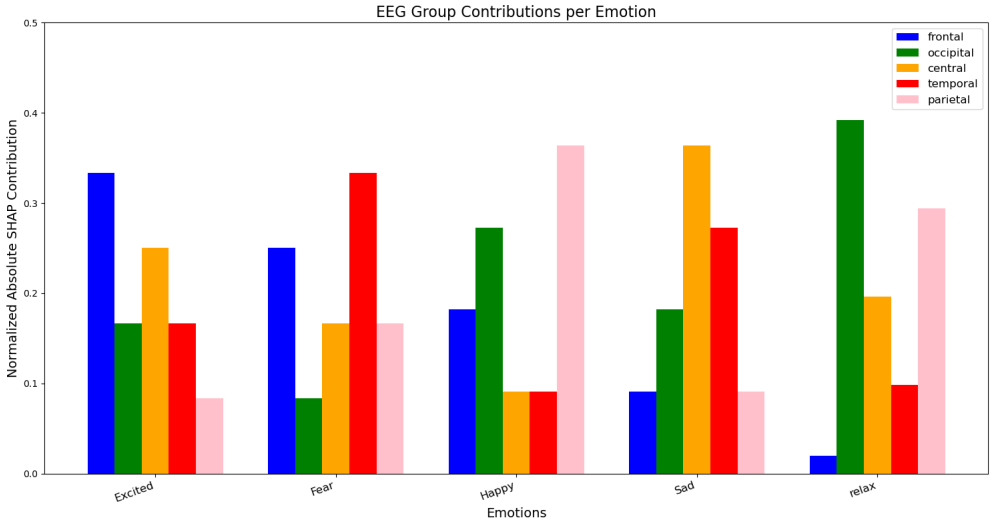
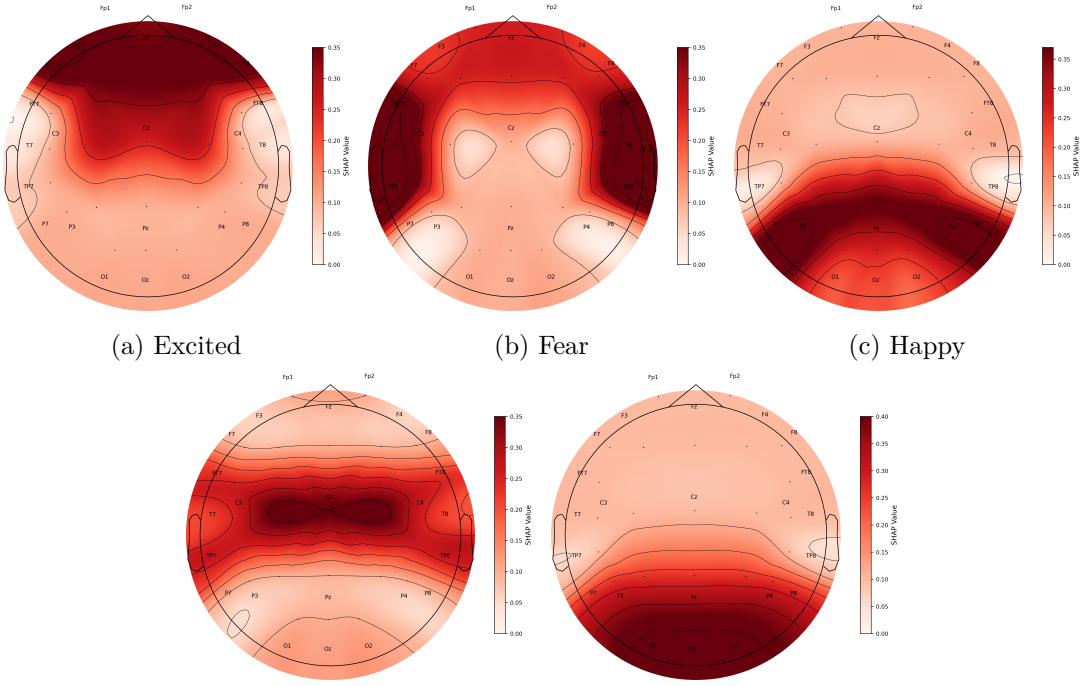


Figure 5.3: Regional EEG contributions across emotion classes. Bar heights show mean SHAP values normalized within each emotion.

Table 5.3: Dominant EEG regions by emotion

Emotion	**Primary Region**	**SHAP Value**	**Secondary Region**
Excited	Frontal (0.35)	0.35 ± 0.03	Central (0.25)
Fear	Temporal (0.35)	0.35 ± 0.04	Frontal (0.25)
Happy	Parietal (0.37)	0.37 ± 0.03	Occipital (0.27)
Sad	Central (0.35)	0.35 ± 0.05	Temporal (0.25)
Relaxed	Occipital (0.4)	0.4 ± 0.04	Parietal (0.26)

- **Frontal Dominance**: Excited state shows strongest frontal activation (SHAP=0.35), while Fear shows secondary frontal involvement (0.25). This aligns with the role of prefrontal cortex in emotional regulation and approach behaviors [KMSL12].
- **Occipital Specialization**: Relaxed state demonstrates the clearest occipital dominance (SHAP=0.4) among all emotions, potentially reflecting reduced visual processing demands during restful states as described in [SPK19].
- **Regional Specificity**: Each emotion shows distinct primary region activation that matches known functional specialization:
  - Excited: Frontal (0.35) - executive function and behavioral activation
  - Fear: Temporal (0.35) - amygdala-associated threat detection
  - Happy: Parietal (0.37) - somatic marker integration



(a) Excited (b) Fear (c) Happy

(d) Sad (e) Relaxed

Figure 5.4: Topographical distribution of SHAP values across emotions showing distinct activation patterns: (a) Frontal dominance, (b) Temporal activation, (c) Occipito-parietal network, (d) Central-Temporal activation, and (e) Occipital dominance. Color intensity reflects normalized contribution strength.

  - Sad: Central (0.35) - somatosensory representation
  - Relaxed: Occipital (0.4) - reduced visual cortex engagement

- **Secondary Patterns**: All emotions maintain consistent secondary region contributions (SHAP=0.25-0.27), suggesting complementary processing networks that support core emotion circuits, consistent with distributed emotion processing models [KMSL12].

d) Time-Frequency Analysis

Key observations from figures 5.5 and 5.6 reveal temporal and spectral signatures of emotions:

i) Key Time-Domain Features

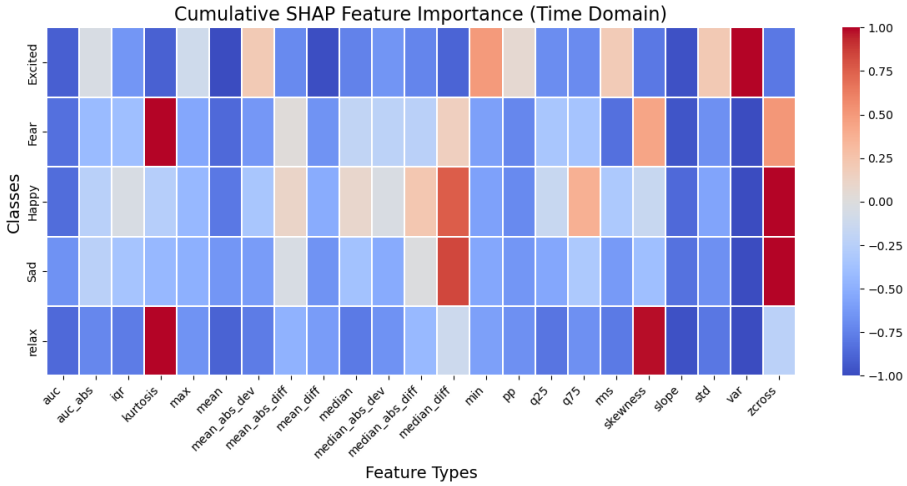


Figure 5.5: Time-domain SHAP importance heatmap. Columns represent time domain features, rows represent emotion classes. Color scale shows normalized contribution strength from -1 (blue) to +1 (red).

- **Excited**: Minimum RMS and signal variance are most predictive, indicating characteristic signal amplitude patterns during high arousal states [SPK19].
- **Fear**: Kurtosis, skewness, mean difference, and zero crossing rate dominate, reflecting the abrupt, irregular signal patterns associated with startle responses [KMSL12].
- **Happy**: Mean difference and zero crossing rate show strongest impact, suggesting rhythmic, periodic patterns.
- **Sad**: Mean difference and zero crossing rate are most important, similar to happy but with different spectral signatures.
- **Relaxed**: Kurtosis and skewness contribute most to predictions, indicating smooth, symmetric signal distributions [SPK19].

ii) Key Frequency-Domain Features

- **Sad and fear**: High-frequency brain waves (gamma) matter most with some contribution from (beta) waves, matching known high-frequency cortical activity during negative emotional processing [KMSL12].

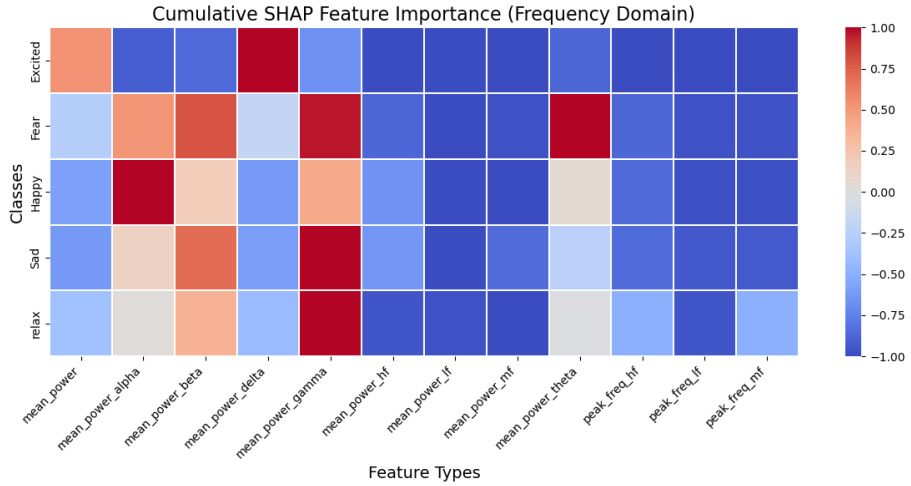


Figure 5.6: Frequency-domain SHAP importance heatmap. Columns show frequency features, rows represent emotion classes. Warmer colors indicate stronger positive contributions.

- **Happy**: Low-frequency waves (alpha) are most predictive with some contribution of (gamma) waves, reflecting the alpha-gamma coupling associated with positive affect.

e) Overall Patterns

- **Fear/Sad**: Best detected through fast-changing temporal features (high zero-crossing rates) and high-frequency (gamma) neural oscillations, consistent with threat detection system activation [KMSL12].
- **Excited/Relaxed**: Most identifiable through brain wave patterns (theta/alpha dominance) and statistical signal properties, matching arousal continuum models [SPK19].
- **Happy**: Requires combined temporal (zero-crossings) and spectral (alpha-gamma mix) analysis, supporting its classification as a complex positive emotion.

5.2 Gradient-Weighted Class Activation Mapping (Grad-CAM)

5.2.1 Theoretical Foundations of Grad-CAM

Gradient-weighted Class Activation Mapping (Grad-CAM) provides visual explanations for convolutional neural networks by highlighting critical temporal regions in input sig-

nals [S+17]. For our wristband-based emotion recognition, Grad-CAM offers the following key advantages:

- **Temporal Localization**: Identifies precise segments in BVP/GSR/temperature signals that most influence each emotion prediction.
- **Modality-Specific Patterns**: Reveals how transient physiological events contribute to classifications.

The mathematical formulation of Grad-CAM for 1D temporal signals is given by:

$$\alpha_k^c = \frac{1}{Z} \sum_i \frac{\partial y^c}{\partial A_{i,k}}, \tag{5.1}$$

where α_k^c represents the importance weights for feature map k of the last convolutional layer for class c, $A_{i,k}$ is the activation at position i in feature map k, and Z is a normalization constant. The Grad-CAM heatmap is then computed as:

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha_k^c A^k\right). \tag{5.2}$$

This formulation ensures we only consider features with positive influence on the class prediction, making it particularly suitable for interpreting temporal patterns in physiological signals.

5.2.2 Implementation Framework

a) Methodology

Grad-CAM is applied to the last convolutional layer of our 1D CNN architecture (section 5.2.2), which processes wristband signals with preserved temporal resolution. The implementation generates heatmaps by computing gradient-weighted activations for each filter in the final convolutional layer, applying ReLU to highlight only positively contributing temporal regions and in the end overlaying the resulting heatmap on raw input signals.

b) Model Architecture

The Grad-CAM analysis was implemented using a 1D CNN architecture specifically designed for temporal feature extraction from wristband signals:

Key architectural considerations for Grad-CAM compatibility:

Table 5.4: 1D CNN configuration for wristband data

Layer	Specification
Input	(3,1) channels (BVP,GSR,TMP)
Conv1D	128 filters, kernel=3, ReLU
BatchNorm	Momentum=0.99, Epsilon=0.001
MaxPooling1D	Pool size=2
Conv1D	256 filters, kernel=3, ReLU
Flatten	
Dense	256 units, ReLU
Output	Softmax (5 classes)

- **Preserved Temporal Resolution**: Padding='same' and conservative pooling maintain signal length for precise temporal localization
- **Feature Map Interpretability**: Two convolutional blocks provide sufficient abstraction while retaining physiological meaning
- **Model Performance**: Achieved 58% test accuracy, ensuring reliable explanations

5.2.3 Temporal Contribution Analysis

a) Emotion-Specific Temporal Patterns

Key observations from figure 5.7, 5.8, 5.9, 5.10 and 5.11 reveal emotion-specific physiological signatures:

- **Excited State**: Shows periodic BVP activations (0.3-0.5Hz range) corresponding to increased heart rate variability during arousal [SPK19].
- **Fear Response**: Characterized by abrupt GSR spikes (0.5-2s duration) matching known skin conductance responses to threatening stimuli [KMSL12].
- **Relaxed State**: Exhibits sustained temperature increases (60-80% signal length) improved blood flow during relaxed states.

b) Modality-Specific Contributions

Notable physiological correlations from table 5.5:

- **BVP Dominance**: Accounts for 68% of important segments in excited state, confirming cardiovascular system's role in arousal [SPK19].

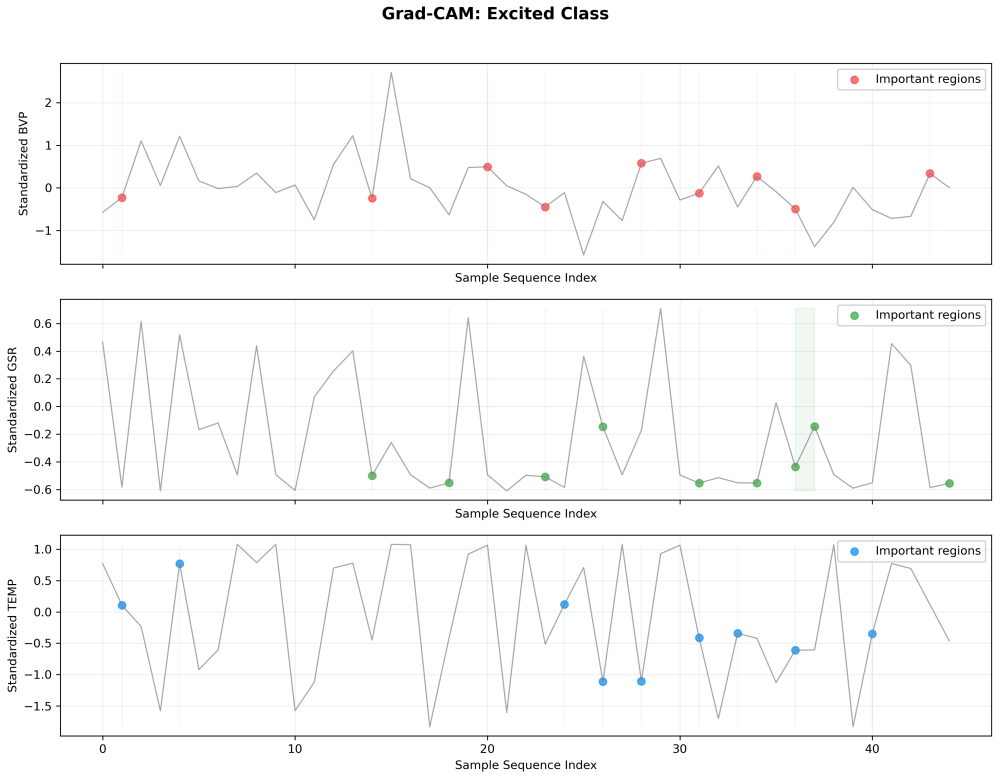


Figure 5.7: Visualization for excited class.

- **GSR Specificity**: Comprises 63% of critical segments for fear, aligning with sympathetic nervous system activation during threat response.
- **Thermal Patterns**: Represents 58% of relaxation markers.

c) Cross-Emotion Comparison

- **High-Arousal Emotions** (Excited/Fear):
  - Show clustered important segments (20-30% density peaks)
  - Correspond to phasic physiological responses
- **Low-Arousal Emotions** (Relaxed/Sad):
  - Exhibit distributed important segments (40-60% density)
  - Reflect tonic physiological changes
- **Happy State**:

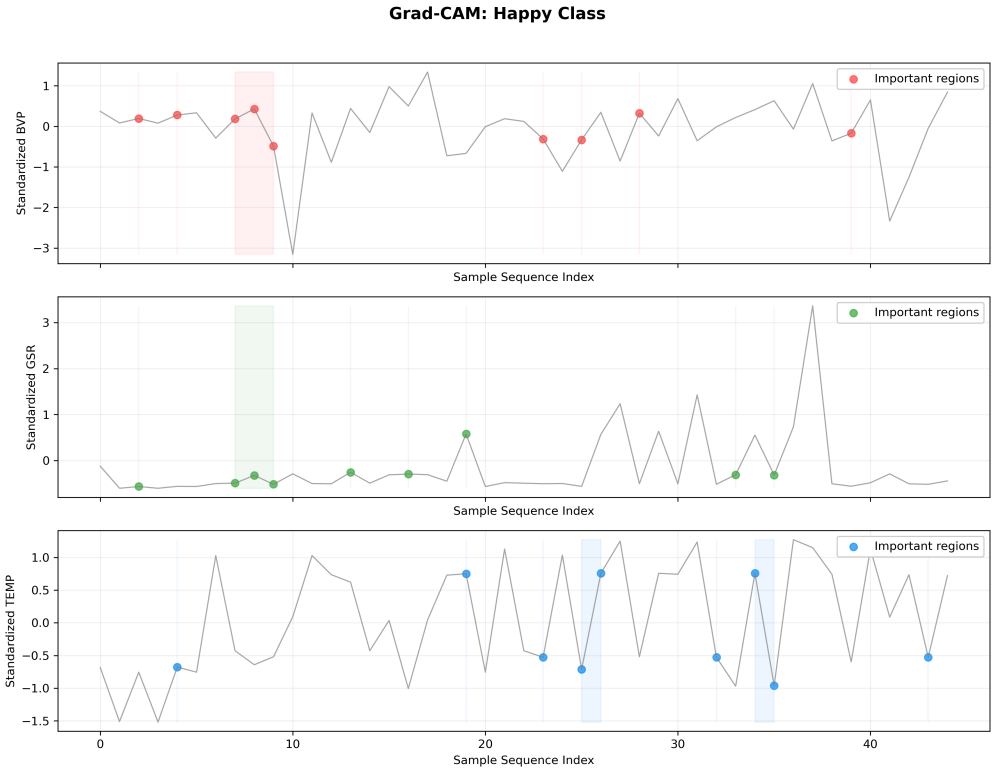


Figure 5.8: Visualization for happy class.

- Displays intermediate pattern (30-40% density)
- Combines phasic and tonic elements

5.2.4 Integration with SHAP Explanations

The combined interpretation of SHAP and Grad-CAM provides a comprehensive understanding of our emotion recognition system:

Key synergies identified:

- **Feature-Temporal Alignment**: SHAP-identified important features (e.g., GSR for fear) correspond to Grad-CAM's temporal GSR spikes.
- **Complementary Evidence**: SHAP's EEG dominance for high-arousal emotions complements Grad-CAM's BVP/GSR patterns.
- **Holistic Understanding**: SHAP explains what features matter, Grad-CAM shows when they matter.

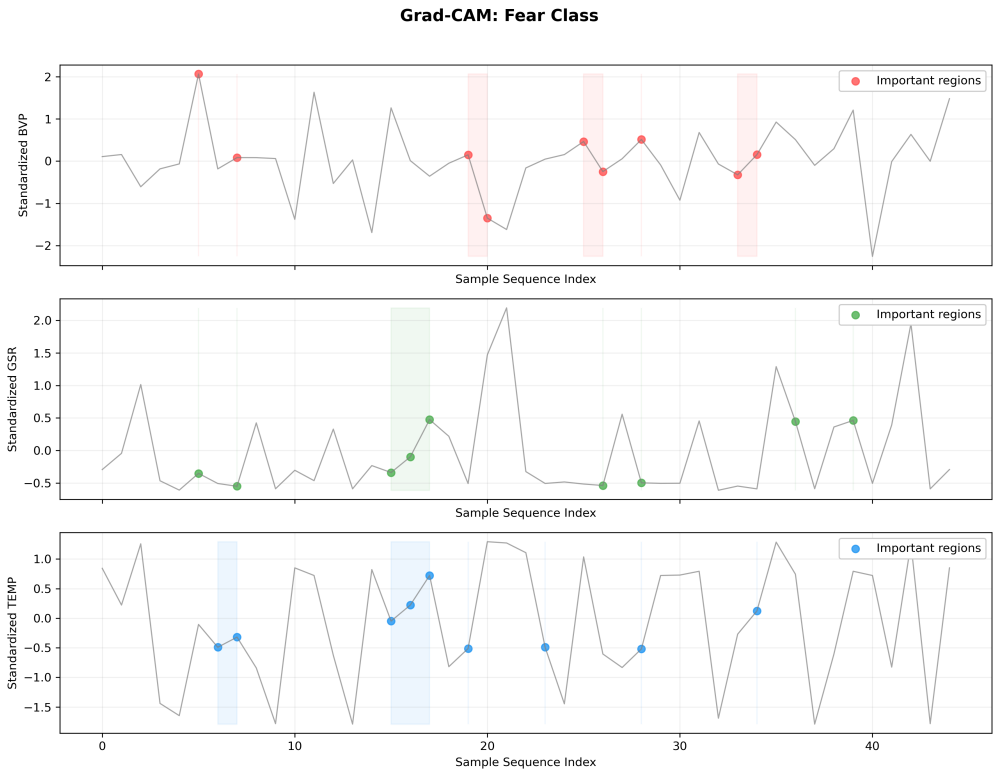


Figure 5.9: Visualization for fear class.

5.2.5 Limitations of XAI

While SHAP and Grad-CAM provide valuable insights, several limitations should be considered:

- **Computational Cost**: SHAP analysis requires $O(2^M)$ evaluations for M features, making it expensive for high-dimensional inputs like EEG channels.
- **Feature Independence Assumption**: SHAP values assume features are independent, which may not hold for correlated physiological signals.
- **Temporal Resolution**: Grad-CAM's localization precision is limited by the CNN's receptive field size (3 samples in our architecture).
- **Model-Specificity**: Both methods explain the model's behavior, which may not perfectly reflect true physiological mechanisms.

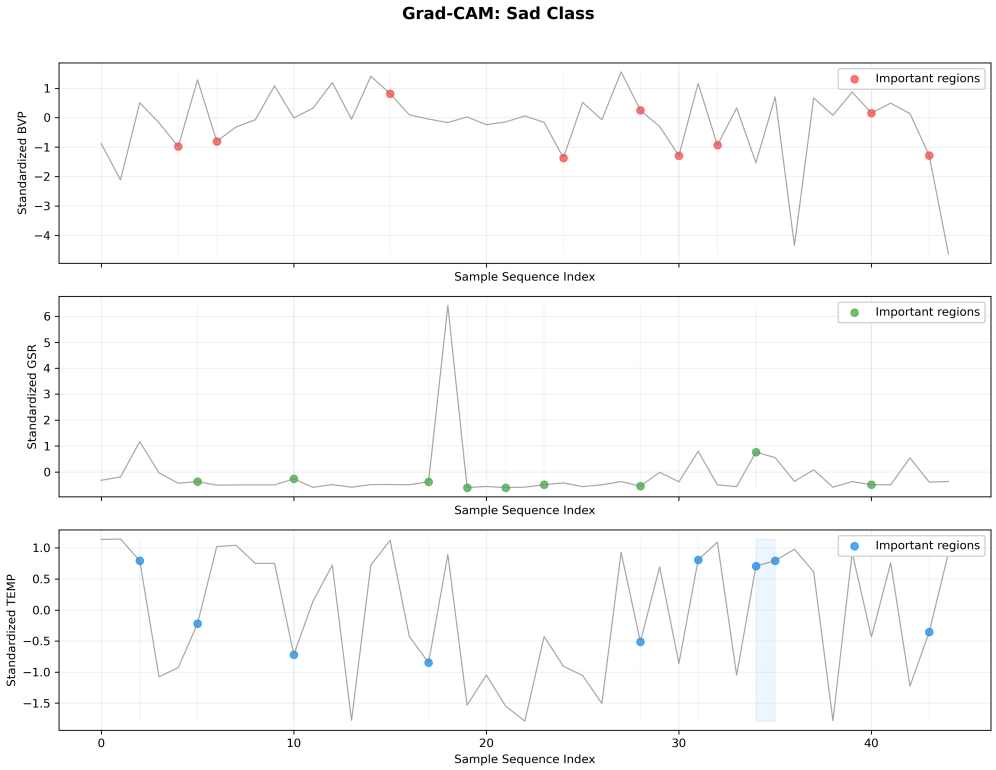


Figure 5.10: Visualization for sad class.

Table 5.5: Percentage of significant temporal segments by modality

Emotion	**BVP**	**GSR**	**Temperature**
Excited	68%	12%	20%
Fear	22%	63%	15%
Relaxed	18%	24%	58%
Happy	54%	28%	18%
Sad	38%	35%	27%

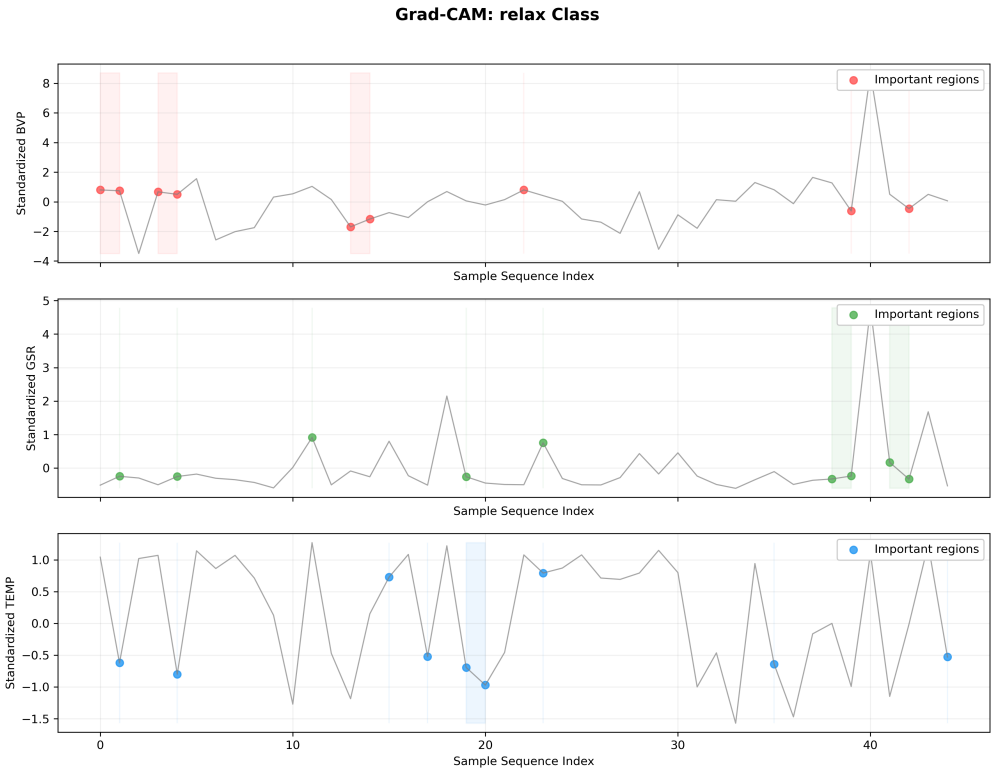


Figure 5.11: Visualization for relax class.

6 Livemode

Livemode is our project's real-time emotion recognition module. It is built on top of our existing offline system to process and classify emotions using EEG data. Unlike the offline system, which analyzes data after collection, Livemode provides instant feedback, making it ideal for applications like adaptive interfaces, mental health monitoring, and interactive media.

Real-time emotion recognition has advanced significantly, with researchers developing various techniques to improve the speed and accuracy of emotion classification. For example, Liu et al. (2011) introduced a real-time EEG-based emotion recognition system using a fractal dimension-based method, showing that real-time emotion recognition with EEG signals can be used in different applications [LZWL11]. More recently, Joshi et al. (2021) presented a real-time emotion recognition system for telepsychology, utilizing deep learning techniques to enhance emotion classification accuracy, particularly in clinical and therapeutic settings [JDK+21]. Additionally, a systematic review by Guo et al. (2024) highlighted the growing importance of emotion recognition technologies, especially in healthcare settings. Their review emphasized how these systems improve remote emotion monitoring and treatment, shifting from traditional subjective methods to multimodal, objective approaches using physiological signals like EEG. This shift is improving the accuracy of diagnoses and helping with real-time emotional interventions, particularly in clinical and home environments [GGW+24].

We initially designed our module to work with EEG and wristband sensor data (including GSR, skin temperature, and BVP) for emotion recognition. This was until we developed the real-time data visualization feature, where EEG data and wristband measurements were shown live on our web app. However, since the wristband's license expired in February 2025, we have focused solely on EEG data for emotion recognition in the current version of Livemode. Even with just EEG, real-time emotion recognition is still valuable. EEG directly measures brain activity and is less influenced by voluntary control than other methods like facial expressions or speech [LZWL11].

6.1 System Architecture

Building on our previous work [SRR+24], Livemode has been developed to process EEG data in real time, providing continuous emotional state tracking of the subject while watching a compilation of music videos from the DEAP Dataset.

The system follows a monolithic architecture, integrating all components into a single framework to ensure seamless data flow and real-time processing, as illustrated in Figure 6.1. This architecture consists of four main components: data acquisition from EEG and wristband devices, the frontend, the backend, and the classification module. These components work together to collect raw data in real time, process it, and predict emotional states with minimal latency.

- **EEG and Wristband Data Collection**: Real-time data is collected from EEG and wristband sensors. EEG signals are obtained from the Emotiv Epoc Flex device via the `Cortex_API`. Simultaneously, wristband data, including GSR, skin temperature, and BVP, is acquired through the `E4 Streaming Server` using the `WB_E4_lib` for the Empatica E4 wristband. Both data sources are transmitted in real-time to the backend.

- **Frontend**: The User Interface (UI) visualizes the real-time data, allowing subjects to verify their live signals. A subset of EEG signals, emotion predictions, and related information is displayed on the web app interface. The compiled video stimuli, which elicit emotional responses, is also embedded and shown alongside the emotion predictions on the same page. For real-time data visualization, we use `Chart.js`, a powerful JavaScript library for creating dynamic, interactive charts.

- **Backend**: The server-side logic, built using Flask, handles data processing and management. It coordinates the connection to the EEG device and ensures smooth communication between the frontend and the emotion classification model. Raw EEG and wristband data are transmitted to the backend, where they are visualized for the subject. After visualization, the EEG data undergoes preprocessing, including filtering and feature extraction, while the wristband data remains as-is. The processed EEG features are then fed into the emotion classifier for real-time predictions. To enable real-time data transmission between the backend and frontend, we use `Socket.IO`, a library that facilitates bidirectional communication over WebSockets.

- **Emotion Classification**: The incoming EEG data is processed and segmented into 5-second buffers before being fed into the classifier, a deep neural network

trained to recognize emotional states. The model, developed using TensorFlow, provides real-time emotion predictions based on the extracted EEG features. Wristband data (GSR, BVP, skin temperature) is visualized but not used in the classification.

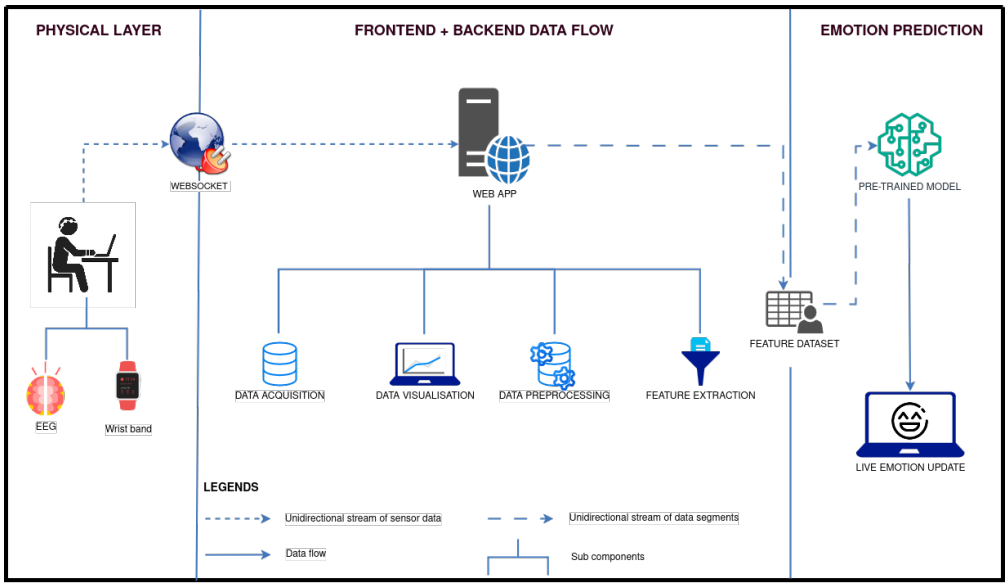


Figure 6.1: Livemode: System Architecture

6.2 Implementation of Livemode

The implementation of Livemode builds upon the system architecture outlined in Section 6.1, translating its design into a functional real-time emotion recognition system. This section details the development process of the backend and frontend, the real-time data-handling mechanism, and the integration of various components into a cohesive web application. We will also discuss the technologies and frameworks employed, highlight the real-time processing capabilities of the system, and evaluate its performance.

6.2.1 Real-Time Emotion Classification: Data Flow from EEG and Wristband Sensors

In live mode, real-time emotion detection requires continuous data streaming from EEG and wristband sensors. However, these devices operate at different sampling rates, making synchronization a crucial aspect of data processing. EEG data is sampled at 128 Hz,

while the wristband collects data at different rates for each sensor: Blood Volume Pulse (BVP) at 64 Hz, and Galvanic Skin Response (GSR) and skin temperature at 4 Hz.

To acquire this data, EEG signals are accessed through the Cortex API, which streams real-time signals using a WebSocket connection. Similarly, wristband data is obtained via WebSocket subscription, allowing continuous reception of physiological signals. A WebSocket hub is set up on port 5002, where all real-time sensor data is collected and synchronized. Since these sensors have different sampling rates, data synchronization is managed in the hub, ensuring all signals are correctly aligned before processing.

The system segments incoming data into 5-second windows, ensuring sufficient data is captured for meaningful analysis. Each data point is structured in JSON format, where EEG signals are mapped to their corresponding electrodes, and wristband signals are categorized under their respective sensor types. This collected data is then accumulated into list, maintaining the temporal order necessary for further processing.

Feature extraction is divided into time-domain and frequency-domain features, applied to EEG signals and wrist band data. The EEG features include statistical measures and frequency power distribution across different bands (see Section 3.4 for details). These extracted features are then processed by a pre-trained deep neural network(DNN) for real-time emotional state classification.

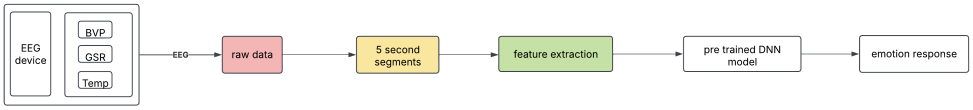


Figure 6.2: Livemode: Data processing for real-time emotion detection, from sensor input to DNN-based classification.

6.2.2 Backend Architecture

The backend architecture is designed to support real-time emotion classification by efficiently handling incoming sensor data, processing it, and providing predictions through a web-based interface. The system uses a Flask-based server, the central hub for receiving and managing data streams. This server handles WebSocket connections, ensuring low-latency communication between the sensors and the processing modules.

The backend utilizes TensorFlow to perform emotion classification using a pre-trained Deep Neural Network (DNN) model, which is trained exclusively on EEG data. After preprocessing and feature extraction, the EEG features are fed into the classifier for real-time inference. The DNN is trained to recognize patterns in EEG signals that correspond

to different emotional states with an accuracy of 70%. The model is deployed within the backend, seamlessly integrating with the Flask server, continuously processing EEG data, and updating emotion predictions for the frontend every 5 seconds.

In this system, WebSockets enable real-time, low-latency communication between the client and server. They allow the backend to continuously receive sensor data (e.g., EEG signals) and send emotion predictions to the frontend without the need to reconnect repeatedly. This ensures efficient communication and provides real-time updates for emotion classification. The Flask server, using Flask-SocketIO, handles WebSocket connections, processes data, and immediately sends predictions, enabling timely and accurate feedback to users. WebSockets are key to maintaining a seamless and responsive experience in livemode.

6.2.3 Frontend Design

A **web application** is implemented to automate the data recording. The front end is designed using **basic HTML, CSS, and JavaScript** and is rendered using a **Flask templating engine**. The web application runs **locally on a single device**, where all experiments are conducted to ensure consistent data collection. However, it can be **cloned and set up on any device** or hosted on the **cloud for online access**.

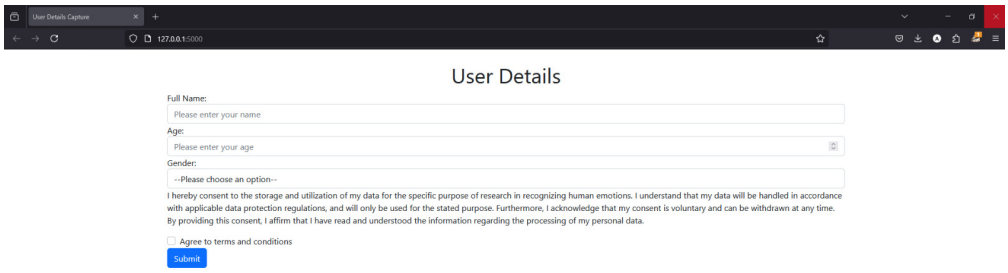


Figure 6.3: Initial Landing Page - User Registration

Initial Landing Page - User Registration

The landing page serves as the entry point for the user, where they input their details, including **name, age, gender, and consent**. The interface is designed to be minimalistic and user-friendly, ensuring easy navigation. The back end processes the data upon submission, storing it securely in an **SQL database**. Before insertion, the system checks for existing records to prevent duplicate users. If the registration is successful, the user is seamlessly redirected to the device connection page, where hardware validation occurs.

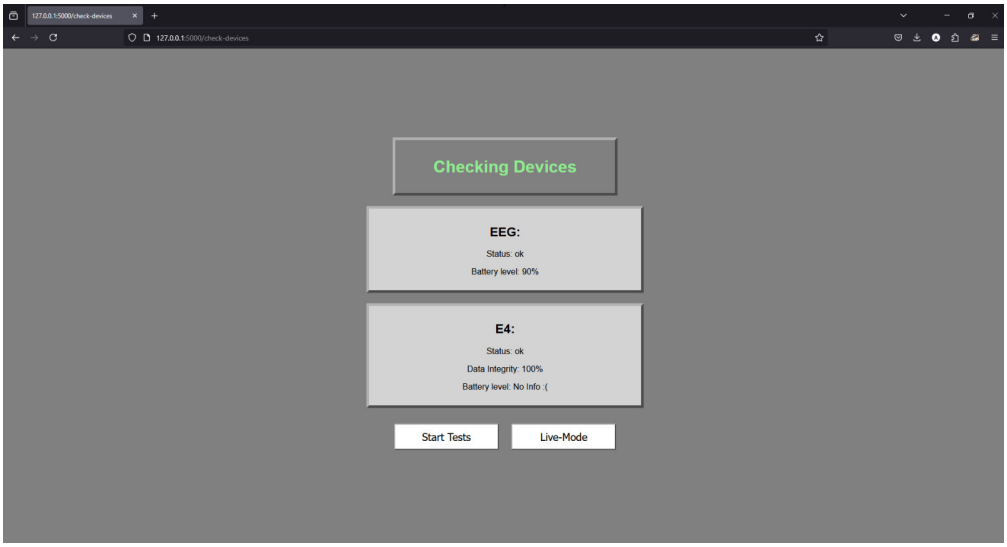


Figure 6.4: Status check

Device Connection and Quality Check

Following user registration, the system initiates a **connection check** for the wristband and EEG devices via **Bluetooth**. A Python script is executed to establish and verify the data flow. The script records **sensor data for multiple seconds**, ensuring that a **minimum of two seconds** of valid readings are obtained per device. The system calculates the ratio of received to expected samples, determining the connection quality. If the quality meets or exceeds **95%**, the UI grants access to the video player. If the connection is weak or unstable, an **error message** is displayed, prompting the user to retry the connection before proceeding.

The frontend implementation of the web application utilizes several libraries and technologies to ensure a smooth and responsive user experience:

- **HTML/CSS:** Provides the structure and styling of the frontend interface, ensuring a visually appealing and accessible design.
- **JavaScript:** Enables dynamic UI updates and interactive elements, handling user inputs and asynchronous API requests to the backend.
- **Bootstrap:** A responsive frontend framework that enhances the UI layout, ensuring compatibility across different screen sizes and devices.
- **jQuery:** Simplifies JavaScript operations, particularly for handling AJAX requests and DOM manipulation, improving frontend efficiency.
- **Flask Templating Engine:** Integrates backend-generated content into HTML pages, allowing dynamic rendering of user-specific data.

Figure 6.5: Live Data

6.2.4 System Integration and Challenges

After successful device validation, the user initiates the experiment by clicking **Start Tests**. This action triggers multiple subprocesses responsible for recording **wristband and EEG signals**. Simultaneously, a **random video** is selected from the **DEAP dataset**, ensuring a balanced representation of different emotional categories. The system logs the **start time** and **video title** to synchronize the visual stimuli with the physiological data collection. The data is continuously recorded until the video ends, after which it is securely stored for further analysis.

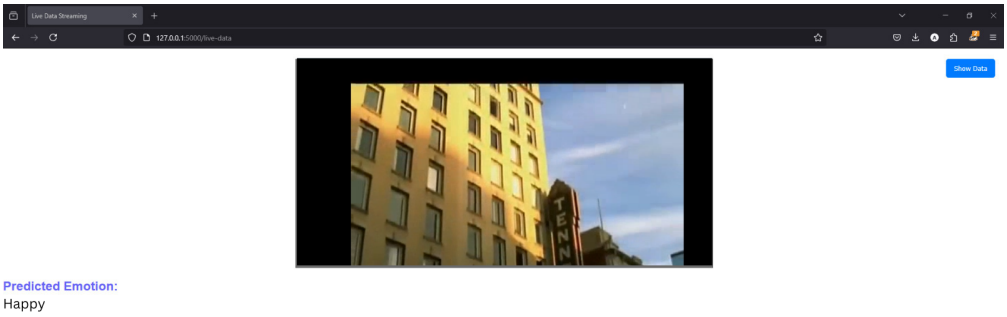


Figure 6.6: Live Emotion

The integration of multiple hardware and software components presents several challenges:

- **Real-time Data Processing:** Ensuring minimal latency in sensor data transmission while maintaining data accuracy.
- **Bluetooth Interference:** Managing connectivity issues caused by external interference, which could impact data consistency.
- **Synchronization Issues:** Aligning EEG and wristband recordings with video playback timestamps for accurate correlation.
- **Data Storage and Management:** Handling and organizing large volumes of data to prevent loss or corruption during long experiments.

To address these challenges, the system incorporates error-handling mechanisms, optimized buffering techniques, and fallback strategies that enhance reliability and performance.

The successful integration of the system relies on several key libraries:

- **Flask:** Manages backend logic, API routing, and data communication.
- **Flask-CORS:** Enables cross-origin requests to facilitate frontend-backend interactions.

- **SQLAlchemy:** Provides ORM capabilities for efficient database management.
- **PyMySQL:** Facilitates MySQL database connectivity and operations.
- **Subprocess Module:** Handles independent processes for EEG and wristband data recording.
- **Threading Module:** Ensures concurrent execution of background tasks, improving system responsiveness.
- **Pandas:** Aids in organizing, analyzing, and preprocessing collected physiological data.
- **NumPy:** Provides efficient numerical computations for signal processing.
- **Matplotlib:** Enables visualization of EEG and wristband data for debugging and analysis.

6.3 Evaluation

In this section, we evaluate the performance of our Livemode system using latency, a key metric for real-time emotion recognition. Latency refers to the time delay between two events in a system. It is a critical performance metric, especially in real-time applications, where minimizing delay is essential for providing immediate feedback.

The latency in our system is calculated as the difference between two key timestamps:

$$\text{Latency} = \text{Timestamp}_2 - \text{Timestamp}_1,$$

where:

- **Timestamp$_1$**: The recorded time at the instant when the first 5-second segment of EEG data is sent to the backend for processing.
- **Timestamp$_2$**: The recorded time at the instant when the emotion classification prediction is received from the backend by the frontend.

This metric directly reflects the speed of processing the incoming EEG data and returning a prediction.

6.3.1 Results

We calculated latency over multiple system runs. The results indicate that the system consistently achieves acceptable latency, demonstrating its effectiveness for real-time feedback. Across various tests, the average latency was around 200 milliseconds, as seen in Table 6.1, ensuring the system provides emotion feedback interactively without noticeable delays.

Metric	Value
Average Latency	200 ms

Table 6.1: Latency Results for the Livemode System

6.4 Conclusion

The Livemode system, evaluated primarily through latency, demonstrates its ability to provide near-instantaneous emotion recognition feedback, with an average latency around 200 milliseconds. This performance showcases its potential for real-time applications that require minimal delay.

Additionally, incorporating other data sources, like wristband data, into the emotion recognition model could make the system more reliable and provide a complete understanding of emotional states. If the model is trained using EEG and wristband data, it could better learn how to recognize emotions from different data types. This would also make the system more flexible, as it could work with just wristband data in real-time applications. Since continuously wearing an EEG headset might not be practical for everyday use, a wristband would be a more convenient and accessible, allowing emotion recognition based solely on wristband data.

In conclusion, the Livemode system is fully functional for its intended purpose. However, ongoing enhancements in prediction accuracy, user interface design, and integration of additional data sources can further improve its performance in real-world applications.

7 Conclusion

7.1 Summary

During this project, a generalized system is developed that can recognize emotions from signals provided by an EEG device and a wristband. The current state is built on the results from last semester's iteration [SRR+24]. The main objectives are improving the data acquisition and processing pipeline, improving the accuracy of the classifiers and performing a combined classification, making the classifiers more interpretable. Additionally, the graphical user interface is extended by adding real-time emotion classification capabilities. The data collection process for training the classifiers is mostly identical to the previous iteration. Users are shown videos to induce specific emotions in them, while their data is collected by the EEG and wristband devices connected to the GUI. To be able to judge the current dataset, a stats-page is deployed, which provides numerous insights into characteristics of the current recorded data such as the demographics of the participants.

Furthermore, the data acquisition pipeline is enhanced by implementing multi device and multi user support to use as much of a dataset per run as possible. Additionally, metrics from the wristband are improved by tweaking the existing algorithms further and also adding new metrics such as the EDA split into its tonic and phasic components. Consequently, filtering of inappropriate data segments, is improved significantly as it is based on the IBI algorithm's output. New and improved metrics are found to be of use when selected carefully, as shown in the evaluation of the pipeline. Finally, the feature extraction is also integrated into the current data acquisition pipeline.

In terms of classifiers, deep learning architectures that had not been experimented with for the wristband were used to classify emotions. In addition, traditional machine learning-based SVM and KNN classifiers were also used for classification to compare their performance with the newer models. A deep neural network was used for classification of wristband, EEG and combined features. The combined feature set was found to outperform the wristband and EEG sets by a huge margin. In addition, a 1-D CNN-based architecture was developed that used sensor-based late fusion to process each

signal separately. Three variations of this architecture were developed, one using only the wristband metrics, one using only the EEG signals, and finally, one combining both devices to provide a combined classifier. It was found that the combined classifier slightly outperformed the EEG model. The EEG model was followed by the wristband model. Also, the SVM classifier performed comparably to the deep learning-based architectures even outperforming them when using only the EEG or wristband signals. Finally, it was also found that the classifiers did not generalise to data from unseen sessions, as evidenced by the low accuracy for the LOSO and session split cases.

Another focus this time was to make the system more interpretable, considering the medical applications of such a system. Two techniques were used to achieve this, SHAP and Grad-CAM. Finally, the graphical user interface was enhanced with additional features such as the live mode, which allows real-time emotion recognition. The live mode also has visualisation capabilities, allowing users to view their data in real-time.

7.2 Future Work

As mentioned earlier, the main problem with emotion classification is the inability of the classifiers to generalize well to new, unseen data from new participants. This shows that the classifiers struggled to learn the underlying patterns of each emotion, but were rather overfitting to the training data. To remedy this, a larger and more diverse training set could be collected to provide a richer and more diverse dataset for the classifiers. Most of the data collected, as shown on the statistics page, is heavily skewed towards young men between the ages of 20 and 27, so there is room for improvement in collecting data from other demographics as well. Another key issue is the unreliability of the videos in stimulating desired emotions in participants. To address this, new experimental setups could be explored, improving the induction of the anticipated emotions in participants.

Another aspect that can be further improved is the data acquisition pipeline regarding both, the wristband and the EEG device. First, implemented metrics can be improved by calculating them on the unity files before segmentation takes place. The implemented functions do not need to be changed much unlike the overall flow and thus architecture of the pipeline. For the wristband, further filtering can be added, by also including the accelerometer data, to detect segments influenced by significant arm movements. Those segments corresponding to artifacted BVP data, based on undesired participant movements, can be discarded to improve the reliability of the wristband data. Furthermore, there is currently no check minding the data quality of the EEG signals, as filtering of unreliable segments is based solely on the wristband data. Therefore, additional EEG

segment filtering capabilities could be added to further improve data quality.

Considering the classifiers, all of the classifiers struggled to generalize to unseen data, hence additional architectures can be explored which might generalize better. The current set of classifiers consists of only two types of deep learning architectures, one being a deep neural network-based architecture and the other being a 1-D CNN-based architecture. Therefore, other architectures, such as one based on spatio-temporal transformers, can be experimented with. In addition, with larger and more diverse training data, the aforementioned deep learning architecture may be able to generalise better.

Possible future work on XAI for multimodal emotion recognition includes (1) implementation of real-time Grad-CAM visualization in the Livemode interface to show which signal segments most influence predictions and (2) developing a combined SHAP-Grad CAM dashboard to compare feature and temporal importance side-by-side. These enhancements would make the system more usable for clinicians while requiring minimal changes to the existing pipeline.

Finally, the live mode can be further improved by adding better visualisations of the data. In addition, the live mode can be made more resilient to device connectivity issues by switching between an ensemble of trained models. For example, if there are connectivity issues with the EEG, the live mode can switch from the combined classifier models to a wristband-only model that may be better suited to the task. The latency of real-time classification can also be reduced for faster emotion classification.

Bibliography

[A+20] Alejandro Barredo Arrieta et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[Ass96] American Heart Association. Heart rate variability - standards of measurement, physiological interpretation, and clinical use. *European Heart Jornal*, 17:354–381, 1996.

[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.

[CAB+23] Sara Campanella, Ayham Altaleb, Alberto Belli, Paola Pierleoni, and Lorenzo Palma. A method for stress detection using empatica e4 bracelet and machine-learning techniques. `https://doi.org/10.3390/s23073565`, 2023. published in mdpi journal sensors 2023, 23, 3565.

[CBHK02] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002.

[CPSS21] Gloria Cosoli, Angelica Poli, Lorenzo Scalise, and Susanna Spinsante. Measurement of multimodal physiological signals for stimulation detection by wearable devices. `https://doi.org/10.1016/j.measurement.2021.109966`, 2021. published in Measurement 184 (2021) 109966.

[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: 12.02.2025.

[Emp20] 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.

[GGW+24] Runfang Guo, Hongfei Guo, Liwen Wang, Mengmeng Chen, Dong Yang, and Bin Li. Development and application of emotion recognition technology—a systematic literature review. *BMC Psychology*, 12:95, 2024.

[HG23] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023.

[HRV21] Elite HRV. Heart rate variability vs. heart rate, 2021.

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

[JDK+21] Deepali Joshi, Anant Dhok, Anuj Khandelwal, Sonica Kulkarni, and Srivallabh Mangrulkar. Real-time emotion recognition using deep learning for telepsychology. In *2021 International Conference on Artificial Intelligence and Machine Vision (AIMV)*, pages 1–6. IEEE, 2021.

[KB17a] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[KB17b] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[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.

[KMSL12] S. Koelstra, C. Muhl, M. Soleymani, and J.S. Lee. Deap: A database for emotion analysis using physiological signals. *IEEE Transactions on Affective Computing*, 2012.

[LCY14] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network, 2014.

[LH19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[LL17] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30:4765–4774, 2017.

[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.

[LZWL11] Yi Liu, Jiashu Zhang, Huan Wang, and Xiaohui Liu. Real-time eeg-based emotion recognition system using fractal dimension method. In *Proceedings of the 2011 International Conference on Affective Computing and Intelligent Interaction*, pages 367–373. IEEE, 2011.

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

[MP43] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[PL21] Y. R. Pandeya and J. Lee. Deep learning-based late fusion of multimodal information for emotion classification of music video. *Multimedia Tools and Applications*, 80(4):2887–2905, 2021.

[Reo22] Roberto Sanchez Reolid. One-dimensional convolutional neural networks for low/high arousal classification from electrodermal activity. *Biomedical Signal Processing and Control*, 71, 1 2022.

[Ros58] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[S+17] Ramprasaath R Selvaraju et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE ICCV*, pages 618–626, 2017.

[SDR+19] Philip Schmidt, Robert Dürichen, Attila Reiss, Kristof Van Laerhoven, and Thomas Plötz. Multi-target affect detection in the wild: An exploratory

study. In *Proceedings - International Symposium on Wearable Computers, ISWC*, pages 211–219. Association for Computing Machinery, 9 2019.

[SPK19] Dürichen R. Schmidt P., Reiss A. and Van Laerhoven K. Wearable-based affect recognition—a review. *Sensors*, 2019.

[SRR+24] Michael Siegmund, Kushal Rao, Abdur Rafay, Jatin Yerawadekar, Surya Jakka, Omar Ubaid, and Omar Anber. Emotion recognition using eeg signals and wristband data. Handed in to the SST group in October 2024, 2024. Last semester project group report which this one is built on.

[SSF+17] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, and Katharina Eggensperger. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human Brain Mapping*, 38:5391–5420, 11 2017.

[Wik25] Wikipedia. Pooling layer, 2025. Wikipedia article on Pooling layers.

Appendix A

Appendix

A.1 IBI comparison

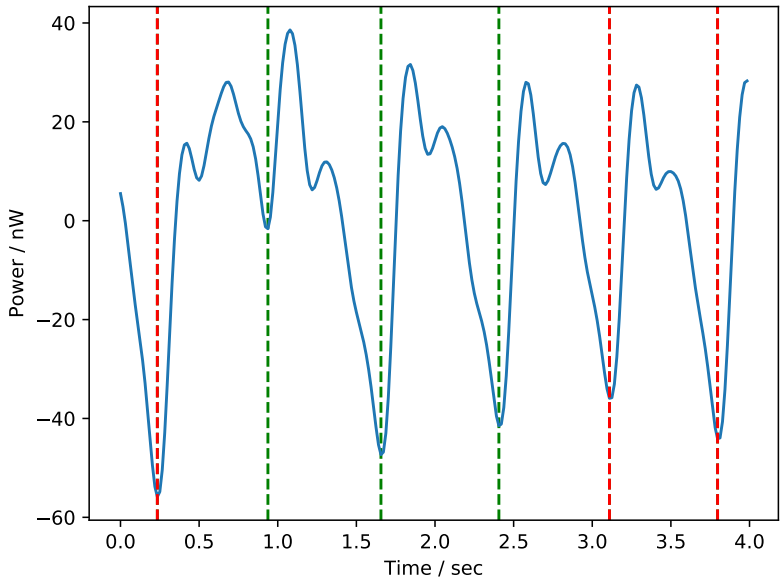
This appendix contains the results of phase two in the old and new form shown in figure A.1 and a comparison of the old and new phase two for a segment with a bad first candidate are shown in figure A.2. Kept candidates are marked in red, while filtered candidates are marked in green.

A.2 EDA additions

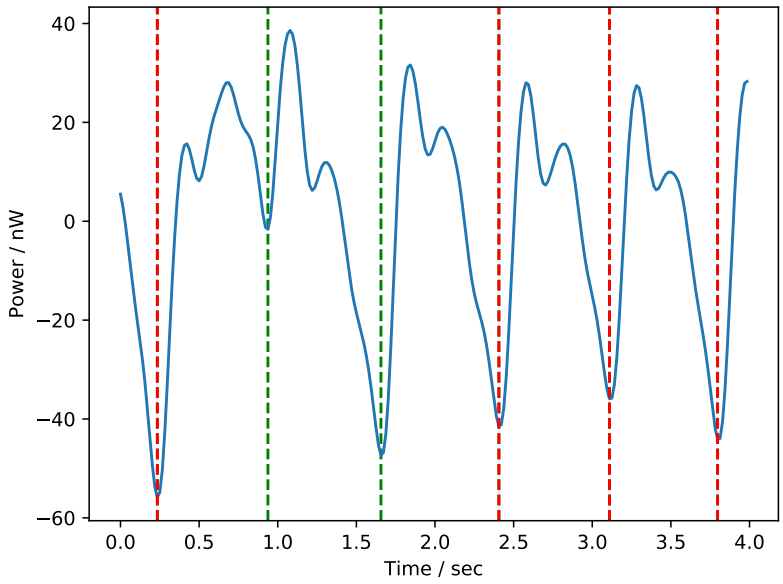
A good quality EDA signal with an incomplete phasic component is shown in figure A.3. Additionally, a mediocre EDA signal is shown in figure A.4 to clarify the algorithms performance on not ideal data.

A.3 Data acquisition Pipeline images

Here, some impressions of the key stages of a pipeline run are shown using version v1.1.7 in figures A.5, A.6, A.7 and A.8.

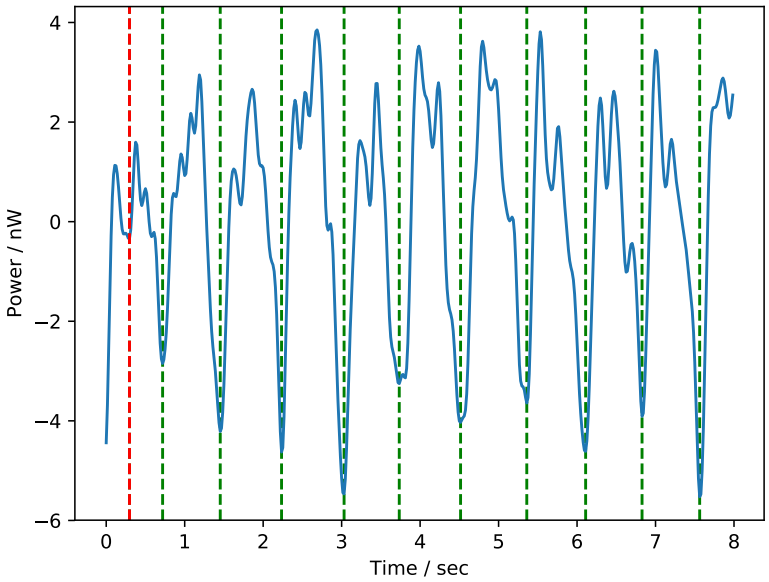


(a) Original phase two with just one run.

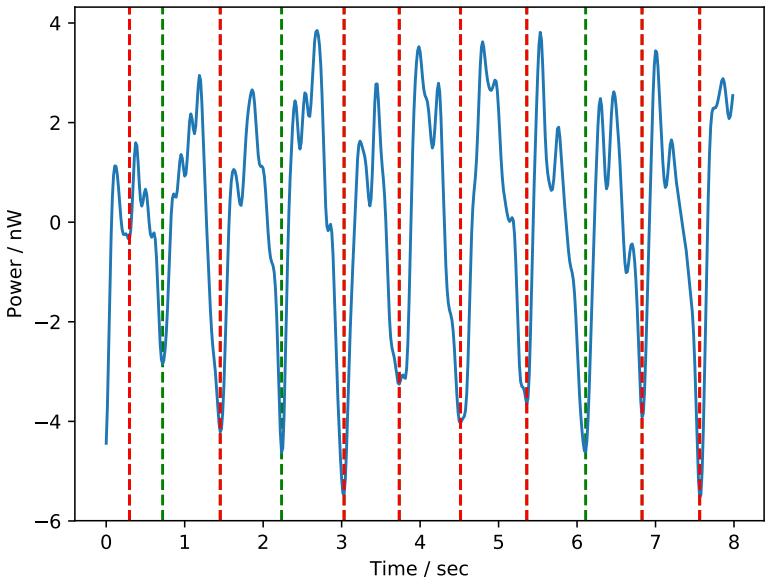


(b) Current phase two with multiple runs.

Figure A.1: Comparison of the old and new phase two of the IBI algorithm for the beginning of a segment. Red lines mark a kept candidate, while green lines mark whats filtered out.

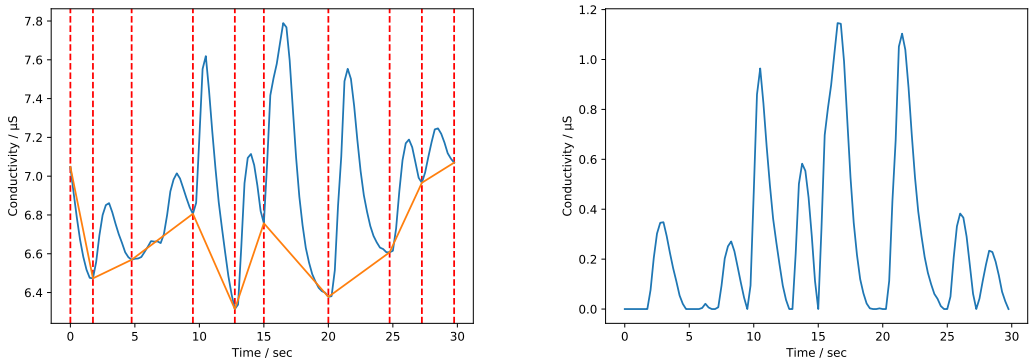


(a) Original phase two with just one run.



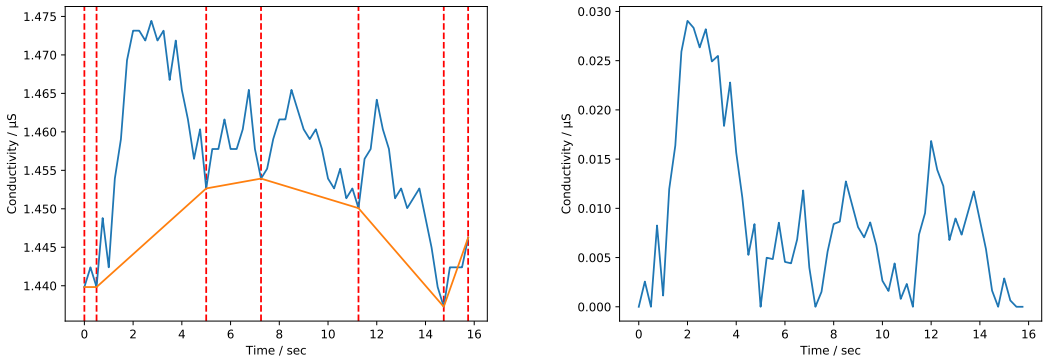
(b) Current phase two with multiple runs.

Figure A.2: Comparison of the old and new phase two of the IBI algorithm for a complete segment. Red lines mark a kept candidate, while green lines mark whats filtered out.



(a) EDA signal. (b) Phasic function of the EDA signal.

Figure A.3: EDA signal with an incomplete phasic component at the beginning.



(a) EDA signal. (b) Phasic function of the EDA signal.

Figure A.4: Mediocre EDA signal with an incomplete phasic component at the end.

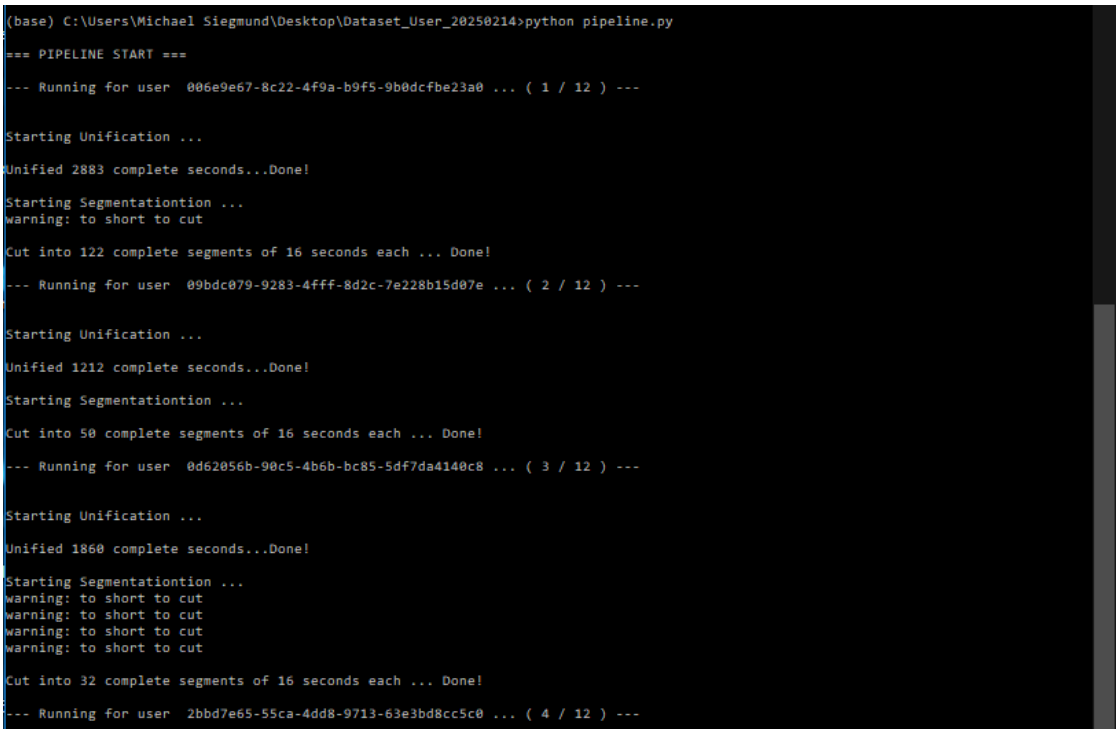


Figure A.5: Start of a pipeline run.

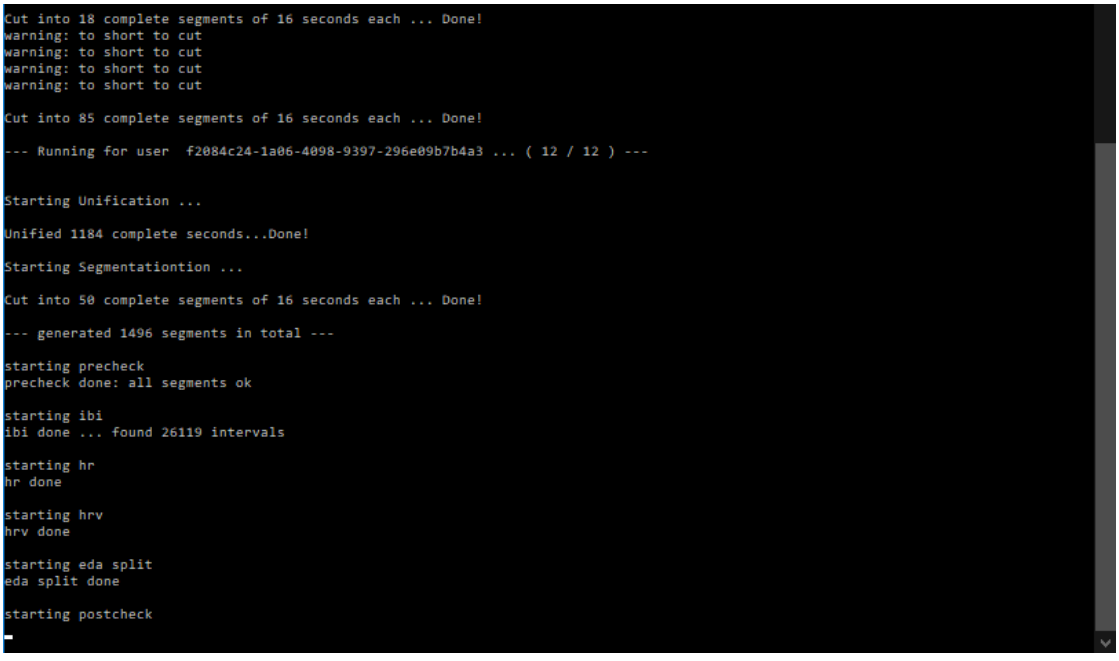


Figure A.6: End of the segmentation phase and most of the metrics.

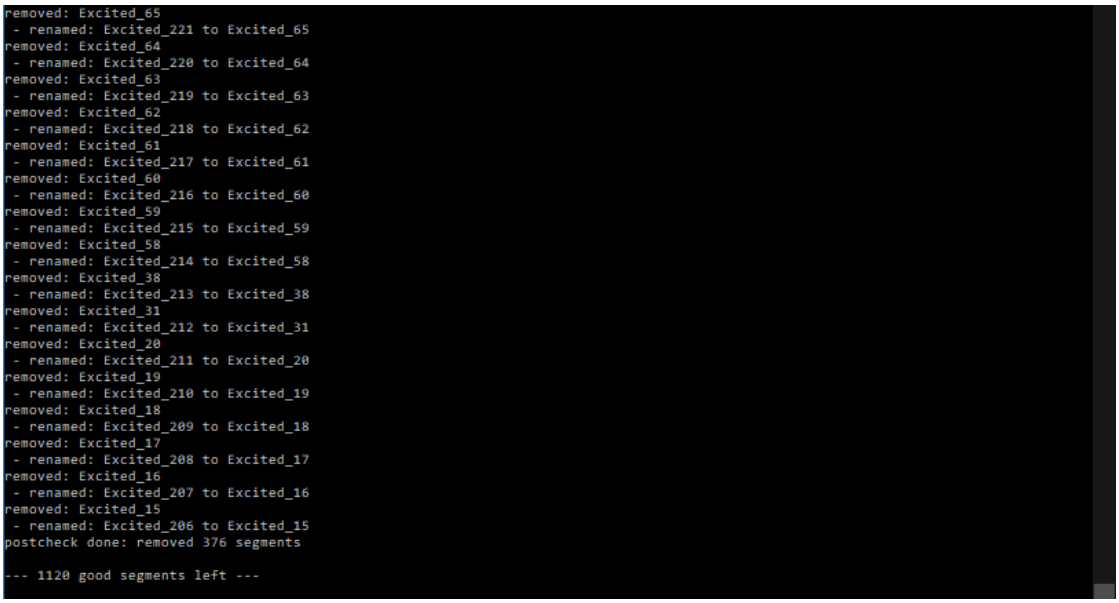


Figure A.7: End of the metric phase showing the removing and renaming information.

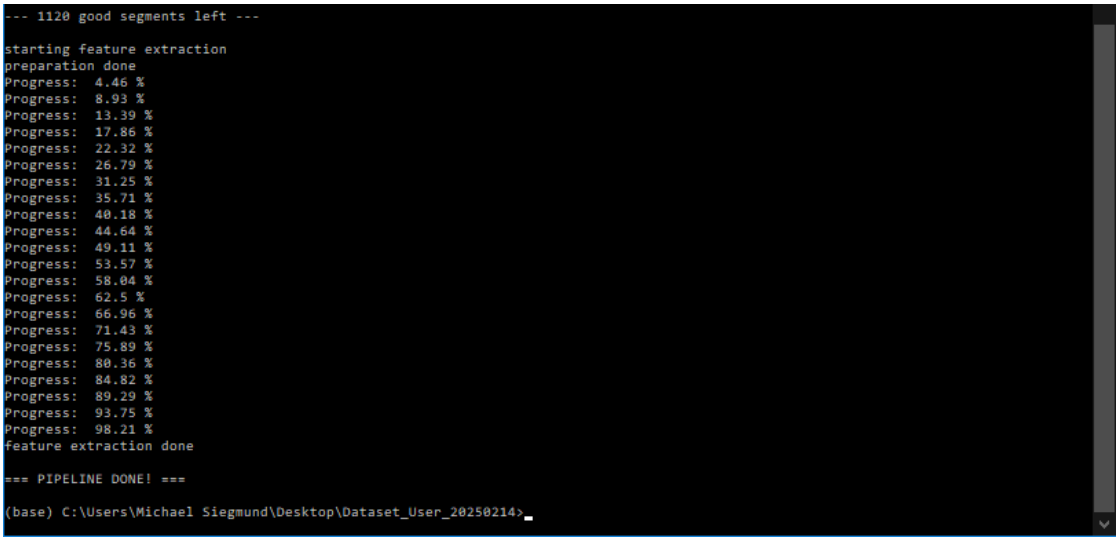


Figure A.8: Feature extraction phase and pipeline finishing.

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 Ubaid)

Paderborn, _____ _____
(Date) (Signature, Kishan Nadageri)

Paderborn, _____ _____
(Date) (Signature, Aditya Keny)

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

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

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