University of West Bohemia
Faculty of Applied Sciences
Department of Computer Science and Engineering

Master's thesis

Movement classification from EEG data

Plzeň 2024 Václav Hrabík

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Václav HRABÍK**
Osobní číslo:	**A22N0047P**
Studijní program:	**N0613A140040 Softwarové a informační systémy**
Téma práce:	**Klasifikace pohybu z EEG dat**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte možnosti detekce pohybu z EEG dat a principy souvisejících rehabilitačních postupů.
2. Seznamte se s protokoly měření EEG dat za účelem detekce pohybu realizovanými na KIV, souvisejícími experimenty a vzniklými datasety.
3. Prozkoumejte existující veřejné datasety, které vznikly aplikací obdobných protokolů jako v bodě 2.
4. Navrhněte a implementujte vhodný(é) klasifikátor(y) pro detekci pohybu z EEG dat nad vybranými datasety z bodů 2 a 3.
5. Výsledky klasifikátoru(ů) z bodu 4 porovnejte s již publikovanými výsledky jiných klasifikátorů použitých nad stejnými datasety.
6. Zhodnoťte dosažené výsledky.

Rozsah diplomové práce:	**doporuč. 50 s. původního textu**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování diplomové práce:	**tištěná/elektronická**
Jazyk zpracování:	**Angličtina**

Seznam doporučené literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce:	**Doc. Ing. Roman Mouček, Ph.D.** Katedra informatiky a výpočetní techniky

Datum zadání diplomové práce:	**8. září 2023**
Termín odevzdání diplomové práce:	**16. května 2024**

L.S.

Doc. Ing. Miloš Železný, Ph.D.	**Doc. Ing. Přemysl Brada, MSc., Ph.D.**
děkan	vedoucí katedry

V Plzni dne 11. října 2023

Declaration

Abstract

Patients with neuromuscular paralysis have a difficult road to rehabilitation. To make this rehabilitation more effective and faster, rehabilitation should not only exercise the muscles but also engage the brain. To achieve that, a Brain-computer interface (BCI) is used. The BCI tries to recognize and translate brain signals into the output commands for devices. For this, we use Motor Imagery (MI), where we imagine the movement without executing it, as it has been proven that these two states are almost identical. In this work, these signals represented by EEG are recognized by recent classifiers such as MLP, CNN, LSTM, and Transformer. This work has shown a large variation between people in the experiments. The most successful people achieved up to 75 % accuracy in classifying their motion.

Abstrakt

Pacienti s paralýzou nervosvalového systému mají těžkou cestu k rehabilitaci. K zefektivnění a zrychlení této rehabilitace je potřeba, aby při rehabilitaci nebyly procvičovány jen svaly, ale i mozek. Pro lepší zapojení mozku se používá Brain-computer interface (BCI). BCI se snaží rozpoznávat mozkové signály a převádět je do příkazů pro zařízení. Za tím účelem využíváme Motor Imagery (MI), kde si pohyb pouze představujeme bez jeho vykonání, protože je dokázáno, že tyto dva stavy jsou téměř totožné. Tyto signály reprezentované pomocí EEG jsou v této práci rozpoznávany aktuálními klasifikátory, jako jsou MLP, CNN, LSTM a Transformer. Tato práce ukázala velkou odlišnost mezi jednotlivými lidmi v experimentech. Nejvíce úspěšní lidé dosáhli až 75 % přesnosti klasifikace jejich pohybu.

Contents

1 Introduction **9**

2 Principles of movement recognition **10**
- 2.1 EEG 10
  - 2.1.1 SMR 11
  - 2.1.2 ERP 11
  - 2.1.3 ERD/ERS 12
- 2.2 Motor imagery 12
- 2.3 BCI 12
- 2.4 Rehabilitation 13
- 2.5 Summary 13

3 Previous experiments on DCSE **14**
- 3.1 Pavel Mochura thesis 14
- 3.2 Josef Yassin Saleh thesis 15
- 3.3 Jakud Kodera thesis 15
- 3.4 Summary 17

4 Public MI datasets **19**
- 4.1 Jeong-25 19
- 4.2 Averta-156 20
- 4.3 Liu-50 20
- 4.4 Farabbi-12 21
- 4.5 Jun-25 22
- 4.6 Dreyer-87 22
- 4.7 Peterson-10 23
- 4.8 Stieger-62 23
- 4.9 Summary 24

5 Classification techniques **26**
- 5.1 Support Vector Machine 26
- 5.2 Naive Bayes 26
- 5.3 Artificial neural networks 27
  - 5.3.1 Convolutional neural network 27
  - 5.3.2 Long short-term memory 28
  - 5.3.3 Transformer 29

5.4 K - Nearest Neighbors 30
5.5 Statistical classification 30
5.6 Summary 31

6 Data analysis **34**
6.1 Data structure 34
6.1.1 Kodera-29 34
6.1.2 Farabbi-12 36
6.1.3 Summary 37

7 Problem analysis **38**
7.1 Chosen classifiers 38
7.2 Types of classification 38
7.3 Feature vectors 39

8 Implementation **40**
8.1 Preprocesing 40
8.2 Classification 42
8.2.1 Statistic classification 42
8.2.2 MLP 43
8.2.3 CNN 43
8.2.4 LSTM 43
8.2.5 Transformer 43
8.3 Postprocessing 44
8.4 Visualization 44

9 Results **47**
9.1 Intra-subject results 47
9.1.1 Time-series representation and binary classification . 48
9.1.2 Time-series representation and multi-class classification 49
9.1.3 Frequency representation and binary classification . . 50
9.1.4 Frequency multi-class 52
9.1.5 Discussion 53
9.2 Kodera_29 results 54
9.2.1 Discussion 55
9.3 Farabbi_12 results 57
9.3.1 Discussion 59
9.4 Hrabik_41 results 60
9.4.1 Discussion 61
9.5 Discussion 63

10 Conclusion **64**

Bibliography **72**

A User Guide **78**

A.1 Structure of the repository 78

A.2 Installation 79

A.3 Work with the Project 80

A.4 Parameters of the Project 80

B Tables of results **82**

1 Introduction

Movement is a daily matter for every healthy person. However, for patients with paralysis of the neuromuscular system, the most common movement of the arm is difficult to impossible. Because of this, the idea is to help these patients with rehabilitation via a Brain-Computer Interface (BCI). This technology allows them to interact with the outside world by interpreting their movement intentions based on brain activity. Thus, the main goal of BCI is to recognize and interpret intended movement based on brain signals.

There are two possible ways how to produce these brain signals. The first way to really make the intended movement is the movement itself, which is, for some patients, impossible. The second way is Motor Imagery (MI). MI is a process that makes it possible to identify movement intentions based on brain activity alone, without a need for an external stimulus. MI is associated with activity in the motor cortex that is similar to that produced during actual physical movement. To work with brain signals, electroencephalography (EEG) is in use.

The aim of this work is to be a continuation of previous works on this topic at the University of West Bohemia. These works focused on the inter-subject classification of arm movement on gathered data, where data from each subject are merged together, thus creating one big dataset. This work takes a different approach through an intra-subject way, where each subject´s data is processed separately. This strategy proves vital because some subjects achieved results around 75 % accuracy of classification.

In Chapter 2, there are introduced principles of EEG. After that, in Chapter 3, there are shown works that were done on this topic by the neuroinformatics group at the University of West Bohemia. The overview of available free datasets is described in Chapter 4. In Chapter 5, there are described classification techniques. Chapters 6 and 7 describe analysis of gathered data. The implementation of the proposed tests is shown in Chapter 8. Accomplished results are presented in the Chapter 9.

2 Principles of movement recognition

This chapter explores the possibilities of movement recognition through Brain-Computer Interface (BCI). To do this, we use electroencephalography (EEG) signals. EEG signals can be represented in multiple ways. To get those signals, we use Motor Imagery (MI). Lastly, there is information on how all this can help people.

2.1 EEG

Recognition tasks require data. These data can vary from obvious images or videos to more complicated EEG signals. An EEG is a test that measures electrical activity in the brain. [1]. This activity is measured by electrodes on the scalp. Localization and designation of electrodes are done by 10-20 System [2]. They created a system of electrodes that measure signals in these parts. For example, parts of the brain controlling the left or right-hand movements are highlighted in Figure 2.1.

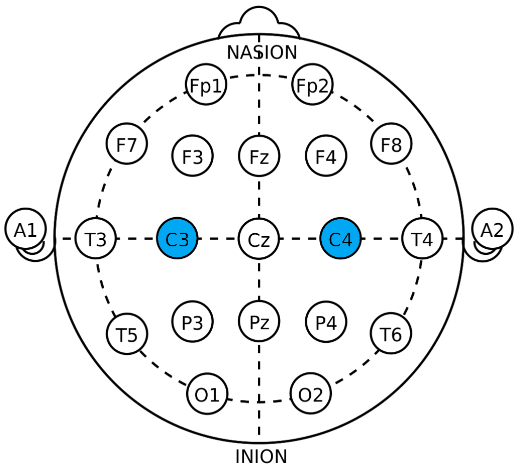


Figure 2.1: Electrode locations of International 10-20 system for EEG (electroencephalography) recording. Image source: [3] Highlighted electrodes C3 and C4 are on parts for controlling movements of the right and left hand.

EEG signals have great temporal resolution due to the speed of neural activity. Thanks to that, EEG signals have the ability to distinguish activities that are milliseconds away from each other. [4] Nowadays, EEG devices have low purchase prices, and they promise great portability. [5]

However, EEG suffers from low spatial distinction. This is because the acquisition values are affected by the bones and tissues of the head. [4] The other limitation of EEG sampling is the high variability of the tested subjects [4]. There are many other problems with the properties of the EEG signal, like the great risk of outer noise, etc. [5]

EEG signals are divided into frequency bands. Each frequency band owes rhythmic activity and transient phenomena. The most used frequency bands are Delta (0.5-4 Hz), Theta (4-7 Hz), Alpha (8-12 Hz), Beta (13-30), and Gamma (30-80 Hz) according to [6] and [7].

2.1.1 SMR

In the Alpha frequency band, there is a special rhythm called Sensory Motor Rhythm (SMR), also known as Mu Rhythm or Mu wave. This rhythm occurs in the central region of the head. There is a parallel with Figure 2.1 because electrodes C3 and C4 are located there as well. When engaging in motor activity or contemplating initiating it, Mu waves tend to diminish. [8]

2.1.2 ERP

Event-related potentials (ERPs) are small voltages generated in the brain structures in response to specific stimuli or events. These are changes in EEG that are time-locked to sensory, motor, or cognitive events. ERPs provide a safe and noninvasive approach to studying psychophysiological correlates of mental processes. They can be elicited by various sensory, cognitive, or motor events and are thought to reflect the summed activity of postsynaptic potentials produced when a large number of similarly oriented cortical pyramidal neurons fire in synchrony while processing information.

ERPs in humans can be categorized into two types. The early waves or components, which peak roughly within the first 100 milliseconds after the stimulus, are called the "sensory" or "exogenous" as they depend mainly on the physical parameters of the stimulus. In contrast, the later parts of ERPs reflect the way the subject evaluates the stimulus and are called "cognitive" or "endogenous" ERPs as they examine information processing. The waveforms are described according to latency and amplitude. [9]

2.1.3 ERD/ERS

Event-related desynchronization/synchronization (ERD/ERS) is a relative power decrease/increase of EEG in a specific frequency band during physical motor execution and mental motor imagery. Thus, it is widely used for the purpose of brain-computer interface . [10]

2.2 Motor imagery

Movement is essential every day for all of us. Our body is directed by the brain, which controls it via signals. These signals come from different parts of the brain. Each part of the brain has control of one part of the body. Localization and designation of these parts in the brain were done in [2].

It is a common fact that brain signals are produced right before movement and during movement. The movement leads to changes in neuronal rhythms and thus can be recognized in signals on electrodes. These changes can be seen as a decrease in the case of ERD or an increase in the case of ERS in amplitude in frequency bands Alpha and Beta (8-30 Hz). [11] Motor imagery (MI) benefits from the fact that all things described above are done during the imagining movement except movement itself. During the imagining, movements are in action in very similar areas of the brain, and it produces almost the same ERD or ERS as movement. [12] MI is one of the Brain-computer Interface (BCI) paradigms.

2.3 BCI

A Brain-Computer Interface is a technology that translates brain activity into signals that can control external devices. These signals can help restore, replace, enhance, or supplement natural neural output, thereby changing the interaction between the brain and its environment. BCIs can be invasive or non-invasive, depending on how they measure brain activity.

In invasive systems, electrodes are positioned on the surface or implanted in the cortex of the brain. In non-invasive systems, electrodes are placed on the scalp or near-infrared spectroscopy.

A typical non-invasive BCI system uses electroencephalography (EEG) to decode the user's intention (e.g., motor imagery or execution) in real time. The system extracts relevant features from the ongoing electrical activity of the brain to detect the user's movement intention. When the system detects the intention, it triggers sensory feedback to the user. This feedback can be delivered in an abstract form (e.g., a moving cursor on a computer screen) or as embodied feedback. Embodied feedback reproduces the intended movement through visual or somatosensory representations. This feedback is shown to enhance motor learning.

Brain-computer interfaces (BCIs) are currently being explored in two clinical applications. The first application is assistive technologies that aim to restore lost functions, such as communication in locked-in syndrome or movements in paralysis, such as eating and drinking despite quadriplegia, using robotic actuators or functional electrical stimulation systems. The second application is rehabilitation technologies, also known as neurofeedback or rehabilitative BCIs, that foster neuroplasticity by manipulating or self-regulating neurophysiological activity to facilitate motor recovery. [13]

2.4 Rehabilitation

According to WHO [14], Rehabilitation is “a set of interventions designed to optimize functioning and reduce disability in individuals with health conditions in interaction with their environment.” From the previous sections, we can derive that BCI, alongside EEG, has great importance in rehabilitation.

So rehabilitation means a learning process where the patient tries to regain old and new skills based on practice. Actively exercising again creates a flow of afferent (i.e., centripetal, i.e., leading from peripheral nerves to the central nervous system) information. Additional mental exercise, such as MI, is crucial in improving motor performance, not just physical exercise.. [15], [16], [17]

There have been several studies using MI that have shown improvements in the speed, strength, and accuracy of performing a motor task. In the learning phase, a similar asymptotic learning curve was found for both MI and physical performance of movements. This therefore further supports the fact that MI is not only epiphenomenal (incidental and indirect) but plays a vital role in cortical plasticity related to the execution of movements. [17], [15]

2.5 Summary

This chapter shows some examples of the representation of EEG signals and how these signals are gathered. Alongside this, there is a description of MI and how it can be used for BCI. Lastly, it is shown how MI and BCI can help people with rehabilitation.

3 Previous experiments on DCSE

This chapter summarizes previous works on this topic at the Department of Computer Science and Engineering (DCSE) of the University of West Bohemia (UWB). The first work was from Pavel Mochura, named Movement Detection from EEG Signals during Exercise on a Rehabilitation Robot [18] in the year 2021. Next year, Josef Yassin Saleh published a work named Design of Movement Detector of measured EEG data [19], which was a continuation of the work from P. Mochura. The last addition to this topic was Jakud Kodera with the work Motion Detection from EEG Data [20].

3.1 Pavel Mochura thesis

Pavel Mochura designed and created the first EEG data set on arm movements. This data set contains data from 14 people; nine of them were women, and five were men. In his thesis, chapter 8 fully describes the entire process.

When he had data, he moved to create feature vectors from it. His idea was to use ERD and ERS to create these vectors. There were created seven feature vectors. The first two were raw ERD and ERS in different setups. The other five vectors were filled with statistical metrics computed from ERD/ERS. The computation of ERD and ERS is in Chapter 9. Statistical metrics are depicted in Chapter 10.

In the classification part of his work, he tried to find the right setting for MLP for the best results. He was tuning the number of neurons or layers in MLP and activation function in neurons. Each test run was done 100 times for some reliability to get enough statistical data about the setting. From the results, the activation function Sigmoid is more reliable than the Tanh function. It has better performance across results. The most successful setting is with three hidden layers and Sigmoid. Layers have a composition of 400 neurons in the first, 200 neurons in the second, and 100 neurons in the last layer. This setting has achieved 90.05%, which is excellent. This result comes from Table 13.3 on page 64 in [18].

Accuracy was not the only goal in the thesis scope. The other goals are the time of classification and the fact that rehabilitation robots could classify only from ERD. In the previously mentioned setting, the time of classification was 2.74s. Compared to other results, which can go up to 10s, this is alright, but it was from a feature vector composed of both ERD and ERS. The right vector composed from ERD only has an accuracy of 86.30% and a classification time of 1.96s. This result comes from Table 13.7 on page 66 in [18].

There are many more results in this thesis, but these two findings are the main results of this work.

3.2 Josef Yassin Saleh thesis

The second is the thesis from Josef Yassin Saleh. This work aimed to classify movement as a thesis before but from a different approach. Mochura used ERD/ERS from 2.1.3. Saleh chose to work with SMR from 2.1.1.

This was one of many goals of this work. The second goal was to add more data for testing. In that manner, other measurements of EEG were performed. The scenario was pretty much similar to the previous one. In this case, the task was performed by 10 people; 6 of them were women, and 4 were men. The process of measurement is described in Chapter 2, Section 4 in [19]. With data from his work and Mochura's work, Saleh has data about 24 people.

For the classification of movement, Saleh chose Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA) (for more information on SVM and LDA: [21]). These classifiers were used with and without Common Spatial Patterns (CSP). So overall, tests were performed on four classifiers.

The results of this work could be more transparent. The first thing that makes me suspicious is that he did not show all the gathered data but only fragments of them. Then, he makes the point that it depends on the proper split of data in training and testing. This is a big problem because primary conditions can not be guaranteed. Lastly, he stated that accuracies are in the range of 20% to 90%, which is interesting. From all that, this work proves nothing. It shows that the process needs to be more general and thus can not be used in everyday practice.

3.3 Jakud Kodera thesis

The last work on this topic was from Kodera. The aim was to try different classifiers on movement EEG alongside data augmentation and various kinds of data representation.

Firstly, he performed the exact measurement of EEG as Mochura to gather more data. In this case, data from five people was collected through EEG. All of them were men. So, in total, EEG data about movement are composed of 29 people.

The data were represented in three ways. The first is basic time series. This is a standard representation of data that came in some sequences. The second representation is the Frequency spectrum. This method promises that there could be some patterns that time series representation can not express. So, by Discrete Fourier Transformation, the signal can be transformed from a time domain to a frequency domain. On the other hand, this signal lost information about time. Because of this, there is the third option, the Time-frequency spectrum. This is a combination of two previous approaches together. Several methods, such as the short-time Fourier transform, wavelet transform, or Hilbert transform, can do it.

Three different methods were used to augment the data. The first method was Noise adding, which simply adds some noise (Gaussian) to the base signal. The second method was Conditional Variational Autoencoders. This method is composed of an encoder and a decoder. Both of these parts are simple MLPs with only one hidden layer. This method generates samples from some conditions in the input. The last method was a Conditional generative adversarial network with a Wasserstein cost function and penalty gradient. This method is composed of two MLPs: one is a generator, and the other one is a discriminator. The generator generates samples from a noise. These samples are evaluated by the discriminator if they are original or artificial. This setup requires more work to configure properly. This method had poor results in this work. The amount of generated samples is equal to 80 % of the original size of the dataset. There are metrics to depict the resemblance of original data and artificial data.

The classification uses five different methods. The first two were the Support Vector Machine and the LDA, as mentioned before. The third was MLP. The fourth was the Convolutional neural network (CNN). The last method was the Long short-term memory (LSTM) type of ANN. The training of these methods used cross-validation tenfold.

The results are shown for all tested setups of augmented data, data representation, and classifiers. Each data representation has its table of results of the classification and a table about metrics, and all are supported by the graph. For each setup, two tests were performed. The first is binary classification, which represents the arm movement state and the state without movement. The second test with multi-class classification represents movement with the left or the right arm and state without movement. Overall, 120 tests were performed. For each test run, classic metrics such as accuracy, precision, recall, and F1 score were computed. The time of learning and the time of classification of one sample was also measured. In terms of overall accuracy, the CNN classifier has the best result of 76% on data without augmentation and using time series representation. The time of classification was hundreds of ms, which can be used for online classification. There is a vast difference in the accuracy of binary and multi-class classification. The best multi-class classification got only 58.57% accuracy, which is nearly a 20% difference. From the results, it can be derived that CNN is suitable for this classification task. It has two reasons. CNN had the best overall result and achieved the best results in 4 out of 6 groups of tests.

3.4 Summary

All previous works have done a good job describing the topic and their method and creating useful information for future works. Table 3.1 shows the summary of results, used methods, and size of the dataset from the works mentioned previously. Across these works, EEG data about arm movement was created from 29 people. There were 15 women and 14 men. From this data, feature vectors such as ERD/ERS, SMR, Time series, Frequency spectrum, and time-frequency spectrum were created. These vectors were put into multiple setups of MLPs, SVMs, LDAs, CNN, and LSTM to train these classifiers. The accuracies range from the worst 20% from Saleh [19] to the best 90.05% in Mochura [18]. This dataset is for further work called with alias **Kodera-29**. This dataset can be downloaded at [22].

Each of those works tries its best to portray its solution and results. Although work from Saleh [19] had several drawbacks described before, it did its job of exploring one possible way of classifying EEG data. The other two works from Mochura [18] and Kodera [20] were more successful.

Author	Number of people	Classification method	Best accuracy
Mochura	14	MLP	90.05%
Saleh	24	SVM, LDA	90%
Kodera	29	SVM, LDA, MLP, CNN, LSTM	76%

Table 3.1: Overview of information from individual works. It contains the name of the author, the number of people in the dataset, the method of classification used, and the best-achieved result.

From these works, several thoughts can be derived. The best feature vectors represented the whole EEG signal. The performance of classifiers on aggregated feature vectors is worse. The worst classification technique in nearly all cases was LDA. The performance of SVM is stable across all test runs. Different kinds of ANN performed well in all cases, such as SVM. In most cases, SVM was even outperformed by ANNs despite the fact that ANNs usually need huge amounts of data to work properly.

4 Public MI datasets

For each dataset, aliases were created, which can be seen in Table 4.1. Alias comprises the name of one author and the number of people included in the data set.

Name	Alias	Reference
Supporting data for "Multimodal signal dataset for 11 intuitive movement tasks from single upper extremity during multiple recording sessions"	Jeong-25	[23]
U-Limb	Averta-156	[24]
EEG datasets of stroke patients	Liu-50	[25]
Motor-Imagery EEG Dataset During Robot Arm Control	Farabbi-12	[26]
A large EEG dataset for studying cross-session variability in motor imagery brain-computer interface	Jun-25	[27]
A large EEG database with users' profile information for motor imagery Brain-Computer Interface research	Dreyer-87	[28]
Motor Imagery vs Rest - Low-Cost EEG System	Peterson-10	[29]
Human EEG Dataset for Brain-Computer Interface and Meditation	Stieger-62	[30]

Table 4.1: Assigning the aliases of obtained data sets.

4.1 Jeong-25

The name of the dataset is Supporting data for "Multimodal signal dataset for 11 intuitive movement tasks from single upper extremity during multiple recording sessions". This dataset includes data about 25 healthy subjects. The data are about 11 different movement tasks of the upper limbs. 60-channel EEG, 7-channel electromyography (EMG), and 4-channel electrooculography (EOG) were recorded during 3-day of measurement. For each subject, 85,500 trials were done. This dataset was released in September 2020. [23]

There is a study [31] from which this dataset originates. The researchers assert that the dataset is appropriate for three main purposes: (i) comparing brain activities linked with actual movement and imagination, (ii) enhancing the decoding accuracy, and (iii) examining variations among recording sessions. Consequently, this study has concentrated on gathering data required for future developments in BCI technology.

On Google Scholar, this study has 46 citations.

4.2 Averta-156

The name of the dataset is U-Limb. The dataset contains information from a total of 156 participants, 91 of them being able-bodied and 65 of them having suffered a stroke. This dataset was released in July 2020. The data is organized into three categories [24]:

(i) Daily living activities of the upper limb, where physiological signals (such as EMG, EEG, and electro-cardiography (ECG)) and kinematic data were recorded.

(ii) Force-kinematic behavior during precise manipulation tasks, where a haptic device was used to record the data.

(iii) Neural hand control, which was measured using functional magnetic resonance imaging.

This study has 17 citations on Google Scholar. The number of downloads is 1800. [24]

4.3 Liu-50

The name of the dataset is EEG datasets of stroke patients. This dataset contains Electroencephalography (EEG) data from 50 individuals who experienced acute ischemic stroke. The participants ranged in age from 30 to 77 years, with 39 males and 11 females included in the study. The time elapsed after a stroke ranged from 1 day to 30 days. Among the participants, 22 experienced right-hemisphere hemiplegia, while 28 experienced left-hemisphere hemiplegia. It is worth noting that all participants were originally right-handed. The first version was released in December 2022. The latest version was released in September 2023.

During the experiment, the participants were seated in front of a computer screen with their arms resting either on a pillow on their lap or on a table. They followed the instructions that were presented on the computer screen. At the start of each trial, a picture with a text description was displayed for two seconds. The participants were asked to focus their minds on the hand motor imagery that was instructed while a video of ipsilateral hand movement was shown on the computer screen for four seconds. After that, there was a 2-second break.

This dataset has zero citations on Google Scholar, the number of downloads is 1200, and the number of views is 3644. [25]

4.4 Farabbi-12

The name of the dataset is Motor-Imagery EEG Dataset During Robot Arm Control. This experiment involved 12 healthy subjects who had no prior experience with neurofeedback or BCI and did not have any known neurological disorders. All participants were right-handed except for one who was ambidextrous (participant number 5). The latest version was released in April 2020.

The experiment was conducted in a laboratory environment under controlled conditions. The subjects underwent three sessions, each lasting a maximum of two hours, over three consecutive days, at approximately the same time each day.

During each session, participants were exposed to three different conditions. The first condition was always the "resting state," during which the user was asked to keep their eyes open for two minutes while staring at a screen with a green cross and a red arrow pointing up and then to close their eyes for the next two minutes. After this, two more conditions related to Motor imagery (MI) tasks were followed and performed in a randomized order between left- and right-hand movements. The two MI conditions consisted of two phases each: a training phase and a test phase. The general experimental routine for both of them was the same. Each trial lasted 6 seconds (2 seconds baseline and 4 seconds MI), forewarned by the appearance of a green cross on the screen and a concomitant beep sound a second before the onset of the task.

Then, an arrow appeared pointing left or right, and the subject had to imagine the movement of the corresponding arm reaching an object in front of the Baxter Robot (Rethink Robotics, Bochum, Germany). For both phases, 20 trials for left and 20 trials for right MI were generated in a randomized order for a total of 40 trials. Finally, there was an inter-trial interval that extended randomly between 1.5 and 3.5 seconds.

There are metrics for the number of downloads, 216, and the number of views, 577. There are zero citations on Google Scholar. [26]

4.5 Jun-25

The name of the dataset is A large EEG dataset for studying cross-session variability in motor imagery brain-computer interface. The dataset comprises data about 25 individuals across five sessions conducted on different days, with a gap of 2-3 days between each session for every subject. Each session encompasses 100 trials of left-hand and right-hand Motor Imagery (MI). The latest version was released in September 2022.

The MI experiment was conducted in three stages. The initial stage, 0-2s, was the preparatory rest period, during which subjects were allowed to rest, make small body movements, and blink. The second stage, 2-4s, was the prompt stage. Here, a left-hand or right-hand movement animation was displayed on the monitor to remind the subjects of the upcoming hand task. The third stage, 4-8s, was the MI process. During this period, subjects performed MI tasks of hand movement in the corresponding direction as per the arrow prompt on the monitor. [32]

There are metrics for the number of downloads 2260, and the number of views 3423. [27] The number of citations is three on Google Scholar.

4.6 Dreyer-87

The name of the dataset is A large EEG database with users' profile information for motor imagery Brain-Computer Interface research. The database contains electroencephalographic signals from 87 human participants, amounting to over 20,800 trials in total and representing about 70 hours of recording. The data was collected during brain-computer interface (BCI) experiments and organized into three datasets (A, B, and C). All three datasets were recorded following the same protocol, which involved right and left-hand motor imagery (MI) tasks during a single-day session.

The database includes the performance of the associated BCI users, detailed information about their demographics, personality, and cognitive profiles, as well as the experimental instructions and codes executed in the open-source platform OpenViBE.

This database could be helpful for various studies, including but not limited to 1) exploring the relationship between BCI users' profiles and their BCI performance, 2) examining how EEG signal properties vary for different users' profiles and MI tasks, 3) leveraging a large number of participants to design cross-user BCI machine learning algorithms, or 4) incorporating user profile information into the design of EEG signal classification algorithms. [28]

There are metrics for the number of downloads 390, and the number of views 1000 [28]. [33] There are two citations on Google Scholar. The latest version was released in January 2023.

4.7 Peterson-10

The name of the dataset is Motor Imagery vs Rest - Low-Cost EEG System. This dataset contains electroencephalography (EEG) signals obtained using an inexpensive consumer-grade device. Ten individuals with no prior experience with the brain-computer interface (BCI) participated in the study. The BCI protocol involved two conditions: kinesthetic imagination of grasping movement (MI) of the dominant hand and a rest/idle condition. The participants were asked to perform five protocol runs, with the first involving natural grasping movement to explain the protocol better and help the subject focus on the sensation of moving. The remaining runs were identical, consisting of MI vs. Rest conditions. EMG signals of the dominant hand were acquired for protocol control. During acquisition, the EEG signals were filtered using a third-order Butterworth bandpass filter between 0.5 and 45 Hz. The latest version was released in November 2021. [29]

There are no metrics for the number of downloads, the number of views, and the number of citations directly in the data portal. However, at the Google Scholar [34], it has three citations.

4.8 Stieger-62

The name of the dataset is Human EEG Dataset for Brain-Computer Interface and Meditation. This database contains EEG data that has been de-identified from 62 healthy individuals who took part in a brain-computer interface (BCI) study. Each individual underwent 7-11 sessions of BCI training to control a computer cursor in one-dimensional and two-dimensional spaces using their "intent." EEG data was recorded with 64 electrodes. Additionally, behavioral data, including the online success rate of BCI cursor control, is also included in the database. Their aim is to characterize how individuals learn to control SMR-BCI systems. The latest version was released in February 2021. [30]

There are metrics for the number of downloads 49244 and the number of views 5348. [35] The number of citations is five on Google Scholar.

4.9 Summary

In this chapter, available data sets were described. For each data set, a short description of the experiment, some basic information like the number of participants or the number of channels, and metrics of citations, downloads, or views were gathered.

There could be several drawbacks. The first is the fact that two data sets (Averta-156 and Liu-50) include patients who suffered a stroke. This could lead to different EEG samples, and thus, this data can be discarded from further usage because the volume of these samples could be too much different from healthy ones. On the other hand, this data can be a useful addition because it can help to cover a wider area. The second drawback is that data sets are created with totally different experiments; therefore, data samples can be too different from each other. However, we want to get quality data from experiments or situations that are most similar.

The overall summary of metrics is in Table 4.9. These metrics are the number of people, the number of downloads, the size of the data set, and the license as assurance that the data set can be used. Alongside metrics for each data set, there is the sum of these metrics. Sizes of datasets range from dozens of MB to hundreds of GB. If the size of the dataset is too much different from the original dataset, there could be two outcomes. The first is when the new dataset is too small compared to the original dataset. This could lead to samples from the new dataset having little to no impact on accuracy. The second outcome is when the new dataset is too large. This could lead to samples from the new dataset having a great numerical advantage; thus, classifiers will fit more into them.

The information about datasets was gathered in February of 2024.

Alias	Number of people	Number of downloads	Size	Licence
Jeong-25	25	-	226.32 GB	CC0
Averta-156	156	1800	35.8 GB	CC0 1.0
Liu-50	50	1200	2.18 GB	CC BY 4.0
Farabbi-12	12	216	2.5 GB	CC BY 4.0
Jun-25	25	25	4.22 GB	CC BY 4.0
Dreyer-87	87	390	27.5 GB	CC BY 4.0
Peterson-10	10	-	67.01 MB	CC0
Stieger-62	62	49244	351.01 GB	CC BY 4.0
Sum	427	52875	649.6 GB	-

Table 4.2: Comparison of available datasets.

5 Classification techniques

There are many classification techniques in the machine learning field. It starts from the simplest Logistic regression to the most complex artificial neural networks. In this chapter, techniques that were chosen for future testing are introduced. All mentioned techniques are supervised machine learning algorithms.

From a statistical point of view, there is no way to tell whether the classifier will produce good or bad results; it must be tested in the actual situation.

5.1 Support Vector Machine

Support Vector Machine (SVM) is the first chosen technique. This algorithm finds a hyperplane between classes in N-dimension space. An indefinite number of hyperplanes can exist between classes. Therefore, SVM aims to find a hyperplane that has the biggest margin between samples of different classes. Maximizing the margin distance provides some reinforcement so that future samples can be classified with more confidence. Hyperplanes are decision boundaries that help distinguish the samples. Instances that fall on either side of the hyperplane can be assigned to distinct categories.

In SVM, support vectors refer to the data points that are closest to the hyperplane. These points play a crucial role in determining the orientation and position of the hyperplane. By utilizing these support vectors, we can effectively maximize the margin of the classifier. Removing any of these support vectors will significantly affect the position of the hyperplane. Therefore, these data points are vital in constructing our SVM. [36]

5.2 Naive Bayes

The second machine learning algorithm is the Naive Bayes classifier. It is part of a family of generative learning algorithms that seek to model the distribution of inputs in a given class or category. Unlike discriminative classifiers, like logistic regression, it does not learn which features are most important to differentiate between classes. Instead, it learns the probabilistic distribution of features in each class.

Naive Bayes has several main assumptions. The first is that features in data are conditionally independent or unrelated to any of the other features. The classification model assumes that all features have an equal contribution to the outcome, even though this may not be true in real-life situations (e.g., in an email, the next word depends on the previous one). However, this simplifies the classification problem and makes it easier to compute since only one probability is needed for each variable. Despite this unrealistic assumption of independence, the classification algorithm performs well, especially with small sample sizes. [37]

5.3 Artificial neural networks

Artificial neural networks (ANN) are a subset of machine learning and deep learning algorithms. The name of a neural network is like the structure taken from a human brain. They are trying to mimic the human brain's behavior with a certain level of abstraction. As the human brain, they are composed of nodes called *neurons*. The neuron consists of two parts. In the first part, inputs are multiplied by the weights and then summed. In the second part, the sum is passed into the activation function, where it is decided if the neuron will pass information further. The structure of the neuron is shown in Figure 5.1.

These neurons are combined into layers. Between layers, there are links called *synapses*. There are three main types of layers. First is the input layer. The input layer reads the input data. The second one is the hidden layer. There can be many but also none hidden layers. A number of hidden layers can differ from one experiment to another experiment. If in the neural network, there are two or more hidden layers, then it is called the *deep* neural network. The last one is the output layer. The common structure of an Artificial neural network is shown in Figure 5.2. The output layer provides the vector as an output. For example, in the recognition tasks, this vector contains the numbers whose sum gives one. The highest value in that vector is the wanted outcome. [38] This basic scheme is called Multilayer perceptron (MLP) [39]. There are various types of ANN. Each type adds some new way of classification.

5.3.1 Convolutional neural network

A convolutional neural network (CNN) is a type of ANN that has a classic feed-forward structure. Besides that, CNN adds new types of layers to its structure. These new types are the Convolutional layer and Pooling layer. These two layers are in tandem. In short, the Convolutional layer computes

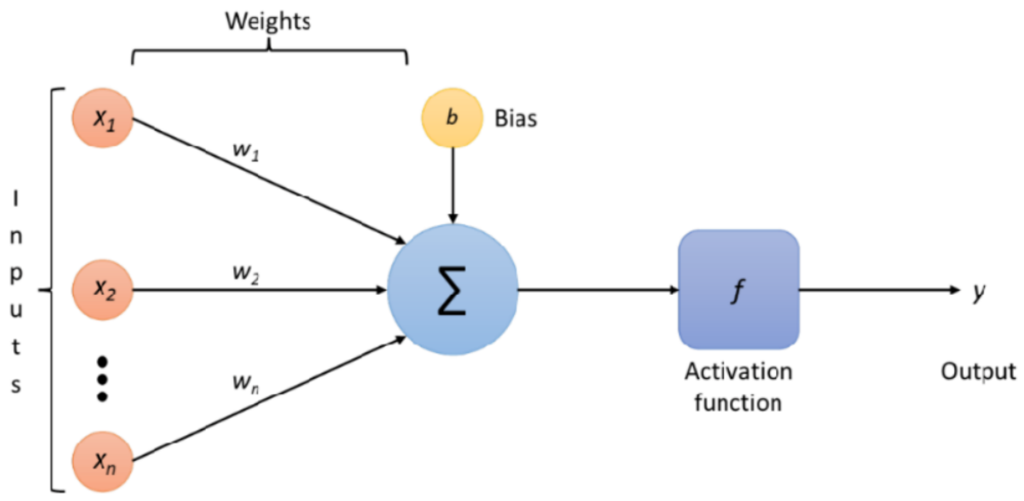


Figure 5.1: Example of neuron structure. The inputs with weights are on the left side. In the middle is the sum. The activation function with the result is on the right side. Source: [40]

the representation of the point by matrix multiplication of its surroundings and chosen kernel. This is done for all points in the input. The output of this operation is called the Convoluted feature. This feature is then passed to the Pooling layer as input. The Pooling layer performs dimension reduction by down-sampling. Together, these two layers perform feature extraction before the output is flattened to pass it into fully connected layers with the same structure mentioned above. An example of the whole CNN is in Figure 5.3, which shows how the Convolutional and Pooling layers are between raw input and classic ANN structure. [41] [42] [43]

5.3.2 Long short-term memory

Long short-term memory, mostly referred to as LSTM, is a special type of Recurrent neural network (RNN). RNN is a neural network where the output of the cell is used in the next steps to enhance the knowledge of the neural network with past experience. These RNNs are great for text or series classification tasks. The reason for that is their memory. Classic feedforward ANN mentioned before learn from its mistakes. RNNs also learn from mistakes but add the memory of previous experiences. RNN has one problem called *vanishing gradient*. This vanishing gradient causes forgetting information over time. The LSTM resolved this problem by changing the inner structure of the cell. [44]

The structure of LSTM is pretty much the same as ANN (see Figure 5.4)

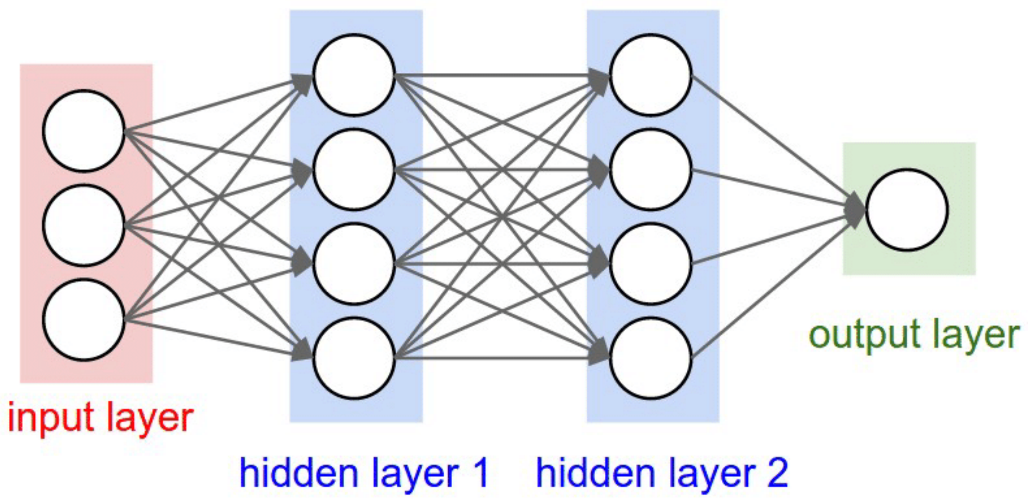


Figure 5.2: Example of ANN structure. The input layer is on the left side. In the middle are hidden layers. The output layer is on the right side. Source: [38]

with one change. The change is the recurrent connection in hidden layers.

More changes are in the neuron, which is called a *cell.* The structure shown in Figure 5.5 is totally different from the structure of the neuron shown in Figure 5.1. Each cell works with three values. The first value is the *input* X_t shown in Figure 5.5 . Other values are *output* and *hidden state* from the previous computation. These two values form the memory of LSTM. For more depth information see [45], [44], or [46].

5.3.3 Transformer

The Transformer is a new type of ANN. It was published in the year 2017 by Google [47]. It is the next step in machine learning with memory. The predecessors like RNN or LSTM address this problem with recurrent approaches. With this approach, RNN and LSTM can take into account the context of other samples. However, the recurrence is a highly time-consuming process. So, the Transformer finds different ways how to take context into computation. [47]

The Transformer structure is shown in Figure 5.6. The Transformer consists of the encoder and decoder parts. The encoder takes the input vector and transforms it into a representative vector. The decoder generates the output sequence from this representative vector and the previous output. [48], [49], [50]

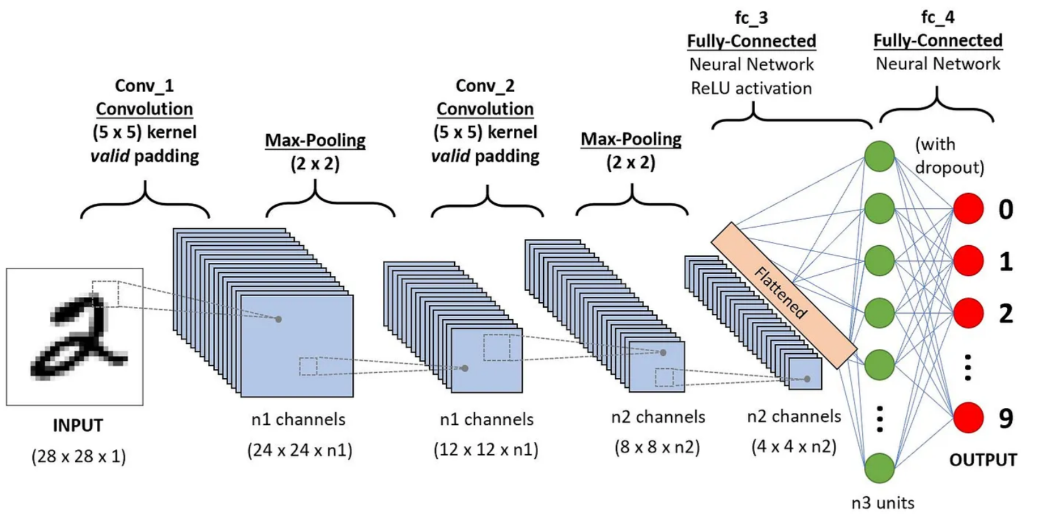


Figure 5.3: Example of CNN structure. The input layer is on the left side. In the middle, there are Convolutional layers and Pooling layers. The output layer is on the right side. Source: [43]

5.4 K - Nearest Neighbors

K - Nearest Neighbors alias KNN is a non-parametric classifier. Although it can be utilised for regression or classification, it is commonly applied as a classification algorithm. The algorithm functions based on the assumption that similar points exist close to each other. KNN uses distance as a metric for classification or prediction tasks. Basically, the method of classification computes distances from all data points in the dataset and looks for K points that are closest to the original point. Based on these points, the classification of the given point can be determined.

The metric determines the distance between two points. There are a lot of different metrics. For instance, the most common Euclidean distance measures a straight line between the query point and the other point, or Manhattan distance measures the absolute value between two points; also, other different distances can be found. [51]

5.5 Statistical classification

This approach uses basic paradigms of statistics like Mean or Median. It takes samples represented as vectors and labels of these samples. For each class from labels, it computes one representative called the pivot. This is done via Mean, Median or other techniques for centroid computation. So,

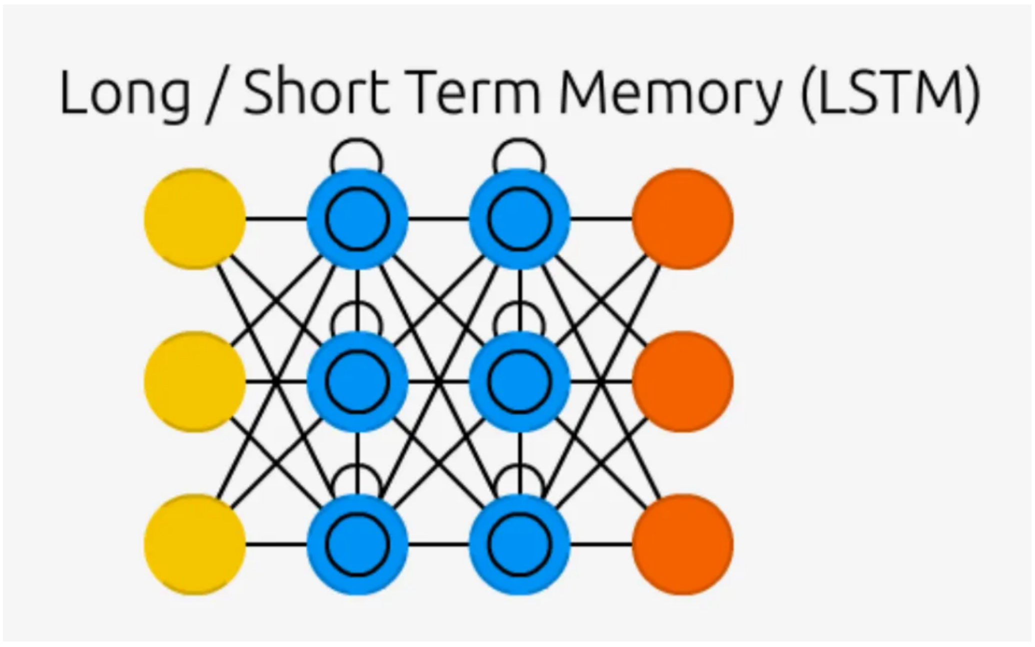


Figure 5.4: Structure of LSTM. The recurrent connection of the cells is shown. Source: [45]

each class is represented by one pivot. The distances between the new sample and the pivots are computed during the classification of the new sample. The new sample belongs to the class with the least distance from its pivot.

5.6 Summary

This chapter briefly introduced some classification techniques. It gives little insight into these methods for further understanding of the topic. For more deep knowledge on classification techniques, you can see [52].

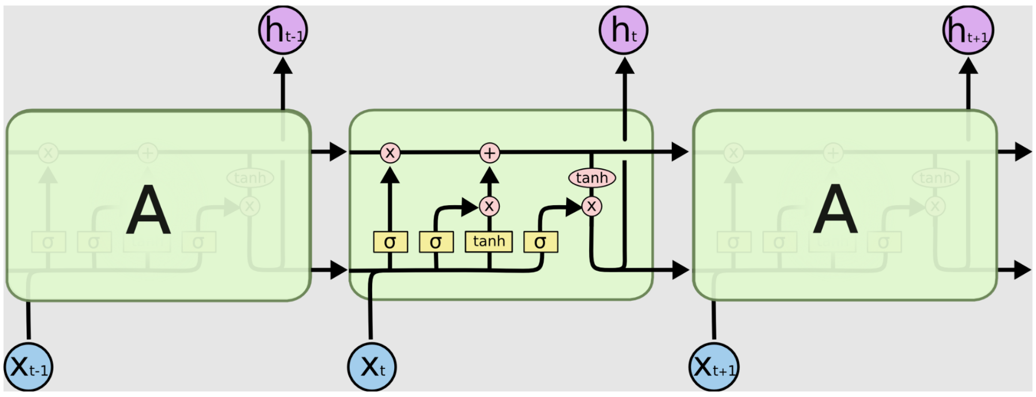


Figure 5.5: The structure of the cell of the LSTM. The same cell is shown in the three different stages. Source: [44]

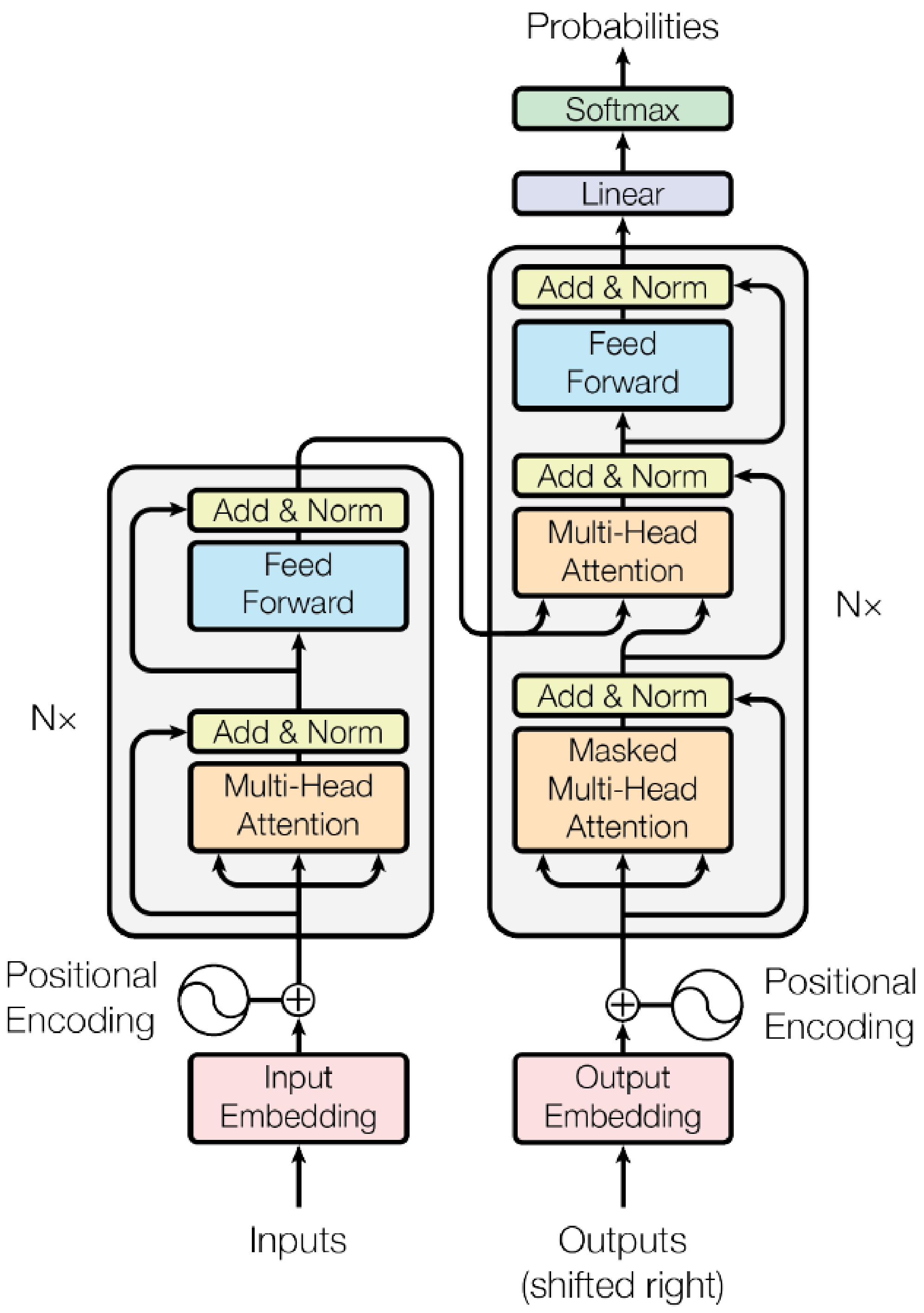


Figure 5.6: Structure of the Transformer. The encoder is on the left side, and the decoder is on the right side. Source: [47]

6 Data analysis

In Chapter 4 there were listed several open access datasets on the same topic as this thesis deals with. Works that created the first dataset at the University of West Bohemia were shown in Chapter 3.

From Chapter 4 was chosen dataset Farabbi-12. This dataset was chosen because the experiment behind it has the most similar description to the experiment done on DCSE. The description of this dataset is the most similar to the dataset Kodera-29. Farabbi-12 has the same stages in the experiment: rest stage, left-hand movement, and right-hand movement. Even though these datasets have the same number of channels (in the system 10-20), the movement of the upper limbs can be recognized in channels Cz, C3, and C4, so there is no need to use all 32 available channels. So, the total count of subjects in the final dataset is 41. All basic parameters for both datasets are in Table 6.1. Datasets have different sampling rates. To merge these datasets together, Kodera_29 must be resampled in 250 Hz.

	Kodera_29	Farabbi_12	Hrabik_41
Number of subjects	29	12	41
Number of channels	32	32	3
Sampling rate	500	250	250

Table 6.1: Table of parameters of datasets alongside their combination.

6.1 Data structure

Both datasets have different structures in terms of the directory structure, file naming, and file structure.

6.1.1 Kodera-29

This is the dataset from previous experiments on DCSE described in Chapter 3. This dataset consists of 11 directories. Each directory represents one measurement with multiple subjects. The name of the directory is the date when it was measured. There are three groups of directories. The first group is in directories from 1. 12. 2020 to 28. 1. 2021. This group consists of 5 directories with EEG data about 14 subjects and was measured by Mochura [18]. The second group is in directories from 2. 9. 2021 to 14. 10. 2021.

This group consists of 5 directories with EEG data about ten subjects and was measured by Saleh [19]. The last directory contains EEG data about five subjects, was measured by Kodera, and was measured on 3. 4. 2023 [20].

There are two naming conventions in those directories. The first convention is used in the first group of measurements from Mochura [18] and the third group of measurements from Kodera [20] (in 6 directories). This convention looks like this:

$$< ID > \_ < sex > \_ < date > \_ < movement > \_ < trial > . < format >$$

- ID - Identification of a subject.
- sex - Sex of the subject.
- date - Date when the measurement was performed.
- movement - Information is about which hand was moved.
- trial - Number of trials for each subject.
- format - Format of the File.

The second convention is used by the second group of measurements from Salech [19]. This convention looks like this:

$$HR\_ < date > \_ < ID > \_ < haptic > \_ < movement > . < format >$$

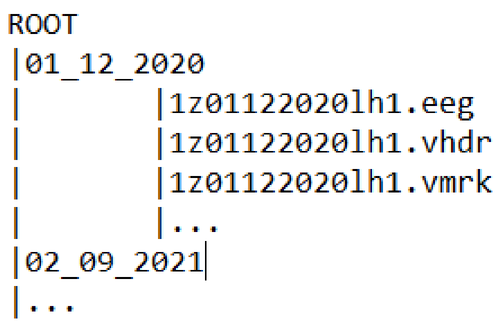
- HR - Prefix. All files have the same prefix.
- date - Date when the measurement was performed.
- ID - Identification of a subject.
- haptic - Information on whether there was a haptic device with vibration during the experiment.
- movement - Information is about which hand was moved.
- format - Format of the File.

Both conversions use the same Brain Vision standard, which consists of 3 separate file formats.

- *.vhdr* - This file is a header file with metadata about EEG data.

- *.vmrk* - This file is a marker file with information about events in EEG data.
- *.eeg* - This file contains the voltage values of the EEG.

To sum up this dataset, a structure is chosen based on each measurement. This means that to get all the data about one subject, we need to go through all the measurements. Figure 6.1 is an example of the structure of directories and files in the file system.



```
ROOT
|01_12_2020
|        |1z01122020lh1.eeg
|        |1z01122020lh1.vhdr
|        |1z01122020lh1.vmrk
|        |...
|02_09_2021
|...
```

Figure 6.1: This is an example of the structure of the Kodera-29 dataset. Within the root folder are directories of each measurement labelled by the date of measurement. Inside these directories, there are sets of three data files corresponding to each subject involved in the measurement.

6.1.2 Farabbi-12

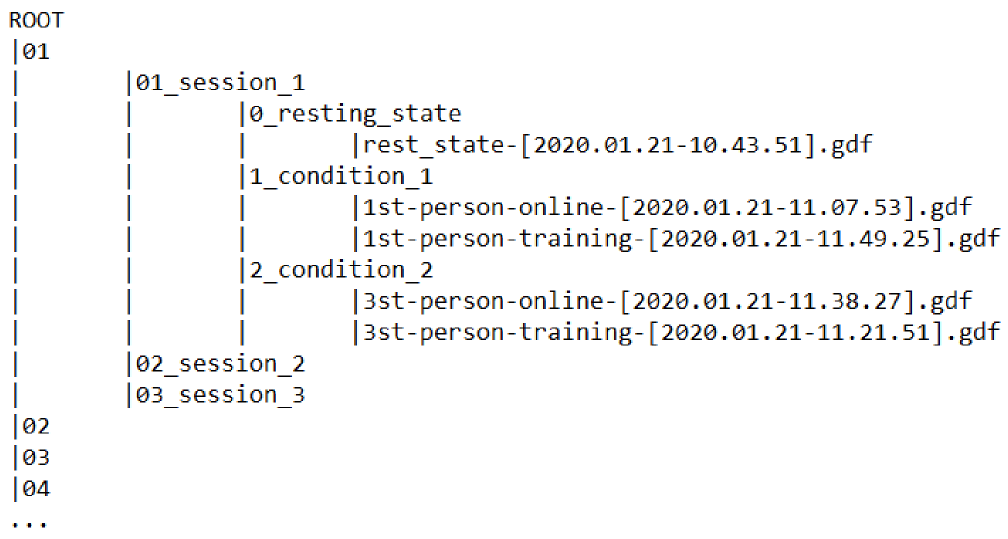
It was got through [26] Zenodo storage. The dataset consists of 12 directories. Each directory has the ID of the subject in its name and represents one subject. In each directory, three directories represent sessions for each subject. In each session, there were three conditions. The conditions are rest state, movement from the first-person view, and movement from the third-person view. The first-person view means the subject is trying to imagine movement like he could. The third-person view implies the subject is trying to imagine the puppet's movement before him. For example, it can be ordered as "Rise right-hand puppet." Each condition has a separate directory. In the rest state directory, there is only one file named in the format:

$$rest\_state - [< date > - < time >].gdf$$

In the other two conditions, there are two files in the format:

$$< view > -person- < phase > [< date > - < time >].gdf$$

The view can become 1st or 3rd, as mentioned before. The phase is information about whether it is a final (called online) or a training trial of the subject. Both phases have the same experiment structure and thus can be merged. The whole structure is shown in Figure 6.2.



```
ROOT
|01
|       |01_session_1
|       |       |0_resting_state
|       |       |       |rest_state-[2020.01.21-10.43.51].gdf
|       |       |1_condition_1
|       |       |       |1st-person-online-[2020.01.21-11.07.53].gdf
|       |       |       |1st-person-training-[2020.01.21-11.49.25].gdf
|       |       |2_condition_2
|       |       |       |3st-person-online-[2020.01.21-11.38.27].gdf
|       |       |       |3st-person-training-[2020.01.21-11.21.51].gdf
|       |02_session_2
|       |03_session_3
|02
|03
|04
...
```

Figure 6.2: Example of the structure of data for one subject.

The format of files *.gdf* [53] is the acronym for General data format. This format envelops all EEG data alongside metadata and markers. The information about left-/right-hand movement is inside those files in the marker part.

6.1.3 Summary

Datasets have different approaches to how they store data. Kodera-29 has files with the movement of a specific hand (the file containing data from the right hand and the file for the left hand), and Farabbi-12 has files with data about the movement of both hands mixed. Besides this, both datasets have the following structure of the experiment with minor changes at the start and the end of measurements.

- rest stage
- movement stage

7 Problem analysis

Chapter 3 presented various works and open-access datasets related to a topic, yet our understanding of the subject remains insufficient. Rather than blindly tuning classifiers, we took a different approach and turned to basic statistics, learning some valuable insights. Given the complexity of the problem, we are determined to cover most outcomes and then try to tell which approach could lead to better results.

The main aim of this thesis is to decide if we are able to enhance our dataset with data from different sources, and the second goal is if it is good to have one general model for all subjects or a specific model for each subject.

7.1 Chosen classifiers

With this in mind, we chose mostly standard classifiers in the field. The chosen classifiers are:

- Statistical classification (SC)
- MLP
- CNN
- LSTM
- Transformer

MLP, CNN, LSTM, Transformer, and Statistical classification are described in Chapter 5.

7.2 Types of classification

Classification can be based on the nature of the dataset, such as binary or multiclass, and EEG specifications that can be either inter-subject or intra-subject. Both are used for comparison to determine if it is better to have a general model or a subject-specific model.

7.3 Feature vectors

Before classification, proper input feature vectors are needed. In the Chapter 3, there are several examples of these vectors. In this work, the **Time series** and **Frequency** representations were chosen. For this, there were a few ideas. The first idea is the desire to keep the most simple flow of the data. The second reason is that if another representation were chosen, it would only mean data transformation. However, the information contained in the data stays the same. The last reason is that most chosen classifiers natively want to do their feature extraction from raw data.

The data has an uneven distribution of samples among different classes, making it inconvenient for cross-validation. To mitigate this, the data needs to be split into class-specific parts and then merged back together. This issue is not seen in inter-subject classification because there is a great volume of data, and both training and testing sets are all types of samples.

So, there are basically two types of feature vectors for each representation. In the first, data is unsorted, and in the second, data is sorted in the way described above. This applies to multi-class classification. For binary classification, there is only one unsorted feature vector for each representation.

For instance, these are labels for subject 09:

$$2, 5, 2, 5, 5, 5, 2, 5, 5, 2, 5, 2, 5, 5, 2, 6, 6, 2, 6, 2, 6, 6, 2, 6, 6, 6, 2, 6$$

As seen from the star, there are only twos and fives. On the other hand, from the middle, there are only twos and sixes (for context, two is rest, five is left-hand, six is right-hand). To eliminate this, this vector must be reordered as described before.

8 Implementation

The project is written in Python version 3.10.4. The whole project is on GitHub: https://github.com/Hrabikv/Diploma_thesis as a public repository. The project's overall structure is shown in Figure 8.1. The project is divided into four main parts. These parts are:

- Preprocessing
- Classification
- Postprocessing
- Visualization

The main entry point is *main.py*, which delegates all work to the proprietary part.

The used libraries are SciPy, NumPy, Plotly, MNE, Scikit-learn, Keras, Matplotlib, TensorFlow, and Pandas.

8.1 Preprocesing

This part is responsible for loading the data from various sources that are described in Chapter 6 and transforming these raw data into a structure that can be used for classification purposes.

The data loading is simple because both data formats have open structures, and there are already functions for loading them. The only problem is to group loaded data from one subject.

The more complicated process was the transformation of the data. After loading, the minimal sampling frequency for both datasets must be found. With this sampling frequency, epochs can be computed. Epochs and their events are created from this process. From these pairs must be chosen pairs with proper events.

In the end, the shape of the loaded and transformed data is as follows:

$$[number\_of\_subjects, number\_of\_events, channels, timestamps]$$

The shape of the labels looks like this:

$$[number\_of\_subject, number\_of\_events]$$

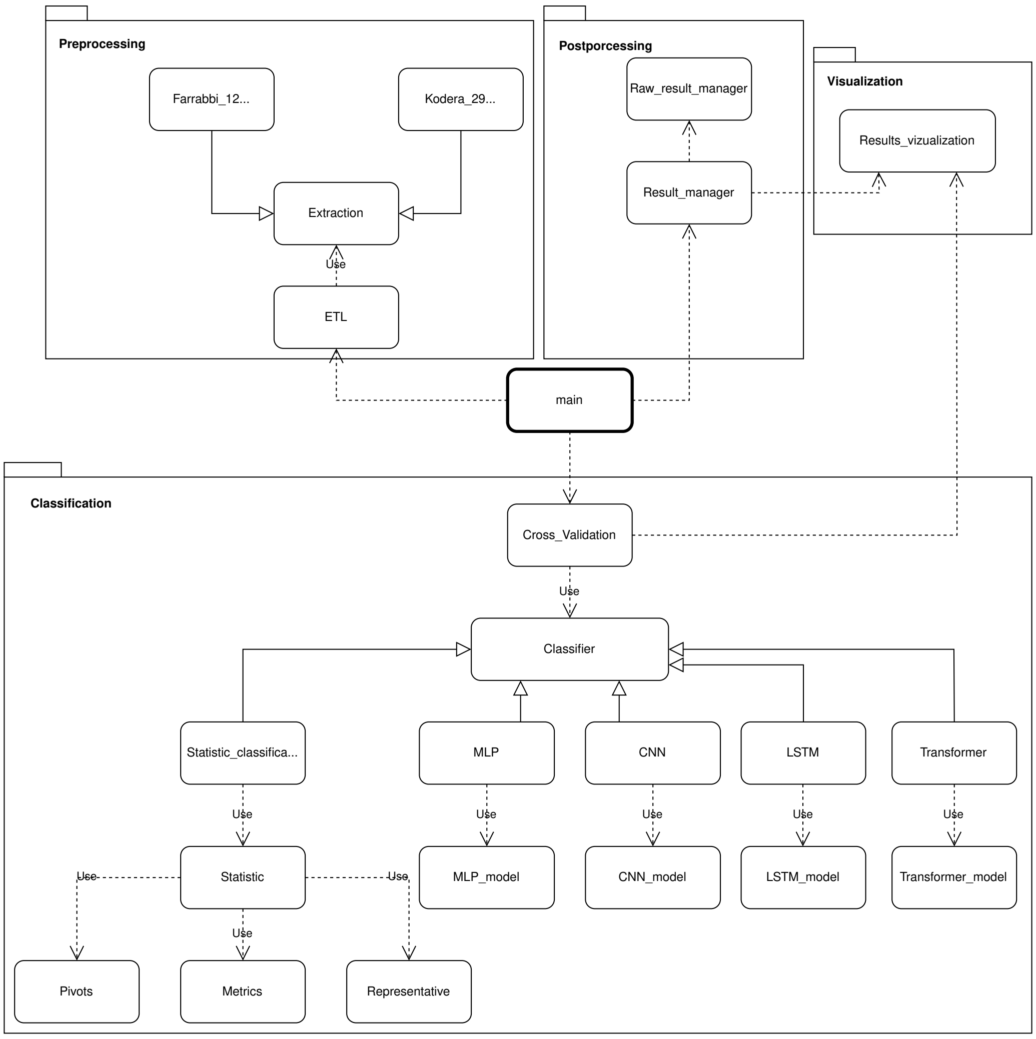


Figure 8.1: The structure of the project.

- number of subjects - It is the number of subjects in a dataset. For example, in Kodera_29, there are 29 subjects.
- number of events - It is the number of rest, left-hand, and right-hand events together. Each subject has a different number of events
- channels - Number of used channels. There are three (C3, CZ, C4).
- timestamps - The length of the signals from the channel. For the used sampling rate of 250 Hz across 4 s, it is 1000 values.

8.2 Classification

The classification uses a cross-validation method to test created models. This cross-validation tries different sizes of folding. Each classifier has a separate class. All classifiers implement abstract class *Classifier* to have methods *train* and *validate*. Each classifier transforms general input data to specific needs. Classifiers produce raw results and other metrics from cross-validation. These raw results and other metrics are then saved during cross-validation in *.csv* in directory *results/raw_results* files for further work.

8.2.1 Statistic classification

This method is very similar to K - means. This approach computes Pivot for each class. Each pivot represents each class during classification. The example of computed pivots with data classes is shown in Figure 8.2. For classification, there are defined metrics that compute the distance between each pivot and the sample that is classified. The sample is then included in the class with the least distance.

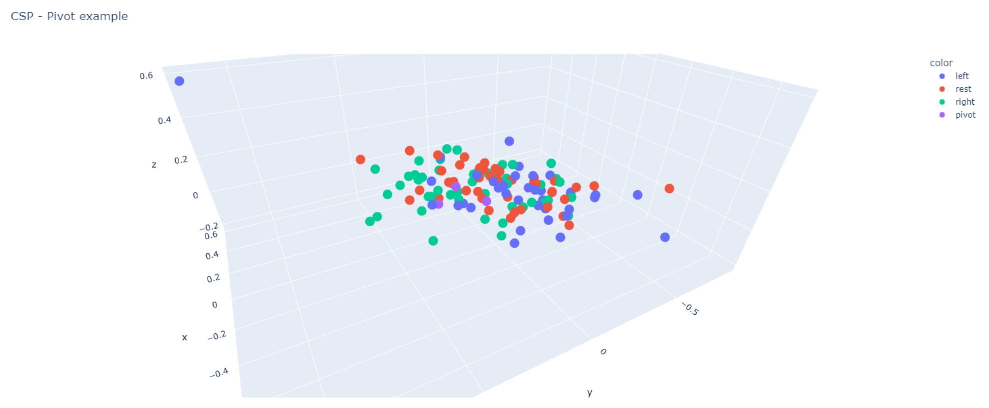


Figure 8.2: Example of computed pivots. Each point represents an EEG signal fingerprint constructed by dimensionality reduction Common Spatial Patterns (CSP) algorithm [54]. The blue color points represent left-hand samples, the red color points represent the rest, and the green color points represent right-hand samples. The purple color points are the pivots of each class.

8.2.2 MLP

MLP consists of 4 layers. The Dense layers have a variable number of neurons. It depends on the size of the input feature. The input layer has the size of the input feature neurons with a ReLU activation function followed by Batch normalization. The core of MLP is two hidden layers. The first hidden layer has two-thirds of the size of the input feature neurons, and the second has a fifth of the size of the input feature neurons. Both layers have ReLU and Batch normalization. The number of neurons corresponds with the number of classification classes in the output layer. Here is Softmax as an activation function. Figure 8.3 shows this structure.

8.2.3 CNN

The created CNN model consists of 5 layers. The first three layers are convolution layers with filter size set to 100 and kernel size set to 5. All three layers have ReLU as an activation function and are followed by Batch normalization. After the convolution block is the Global average pooling layer. The output layer is a dense layer with a softmax activation function and a variable number of neurons, such as MLP. The whole structure can be seen in Figure 5.3.

8.2.4 LSTM

The LSTM model is the smallest of all NNs. It consists of 3 layers. Two of these layers are LSTM layers. The first is the input layer, which has 250 LSTM units, and the second is the hidden layer, which has 150 LSTM units. Both LSTM layers use ReLU as an activation function. The output layer is a dense layer with a number of neurons corresponding with a number of classes and a softmax activation function. LSTM structure is shown in Figure 8.5.

8.2.5 Transformer

The Transformer model is the most complex one as it consists of 15 layers. The Encoder has 12 layers, and the last three layers are for classification. Furthermore, the Encoder consists of four blocks, each with its own three layers. The first layer is the MultiHeadAttention type with four heads. Each head has a size of 250, and each layer has Dropout and Layer normalization. The second and the third layers are the same. Moreover, both have four filters with kernel size 1, an activation function is ReLU, and have Dropout.

After these blocks, the Global Averaging pooling layer is created. The last two layers are fully connected. The first of the¨two has 125 neurons, ReLU as an activation function, and has Dropout. In comparison, the output layer is the same as before. The whole structure is displayed in Figure 8.6.

8.3 Postprocessing

This part of the project loads raw results from the classification. From them, it computes these metrics:

- the average accuracy of classification for tenfold cross-validation
- the best accuracy from all folds
- the average time of training
- the average time of classification of one sample

The results are saved in *.xlsx* file by the Pandas framework.

8.4 Visualization

This part of the program is for printing graphs of raw data, raw results, and results. There is a function to print graphs of raw data with the help of CSP reduction dimension [54]. It is responsible for creating graphs of raw results of accuracies from cross-validation. Lastly, it has a method to print a graph for each metric described in the previous Section 8.3.

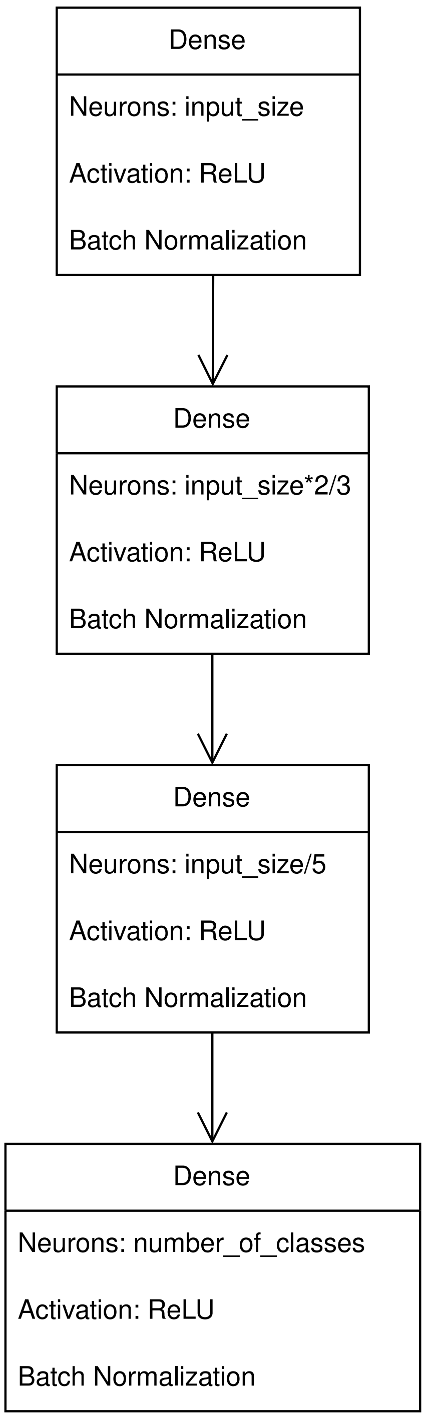


Figure 8.3: Structure of MLP.

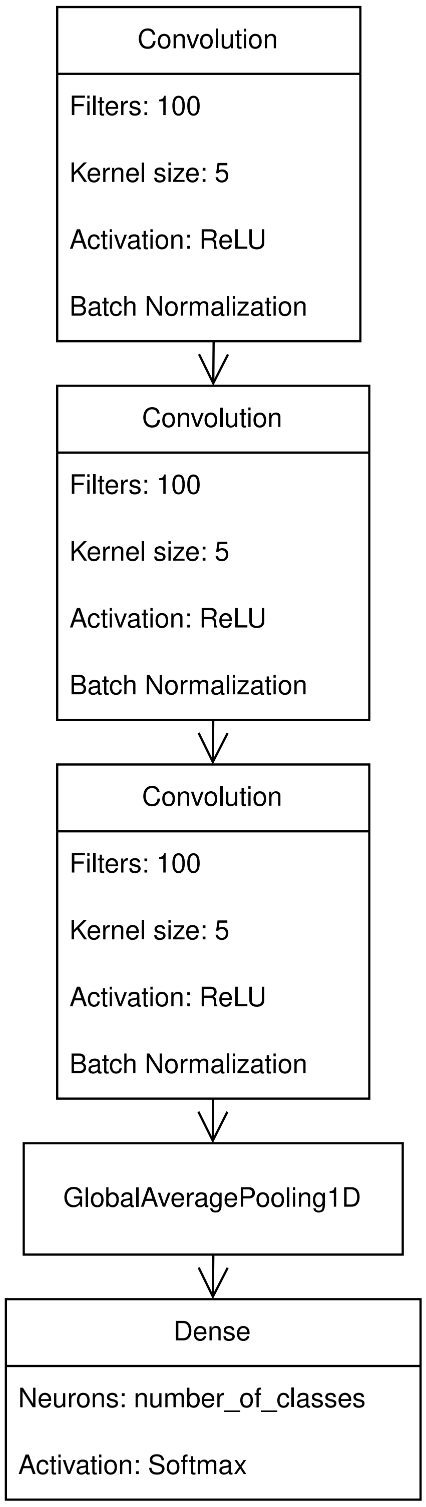


Figure 8.4: Structure of CNN.

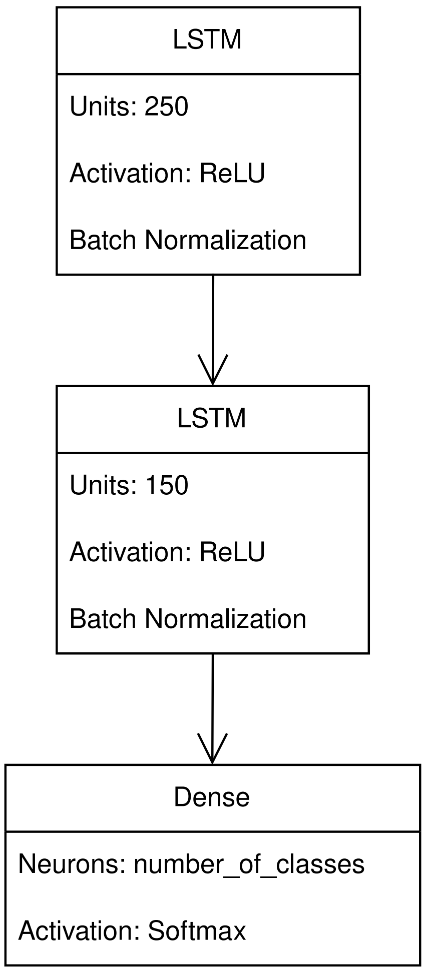


Figure 8.5: Structure of LSTM.

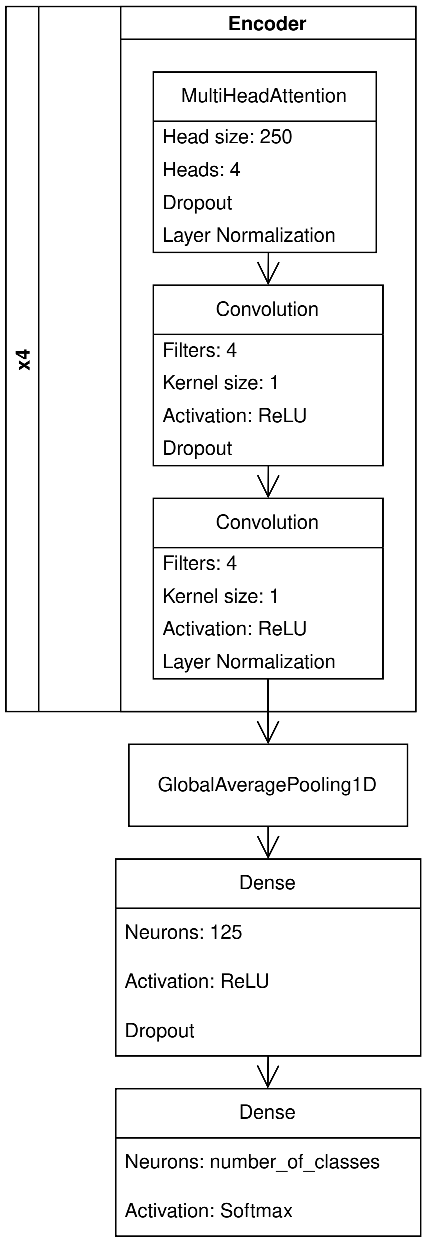


Figure 8.6: Structure of Transformer.

9 Results

This chapter conducts results from all possible combinations of the parameters (types of classification, feature vector) described in Chapter 7. A total of 4 different tests on each possible dataset were performed. The results are focused on average accuracy from cross-validation. However, for completeness, there are data about the time of the training of each model alongside the time of classification of one sample.

These metrics are computed from 10-fold cross-validation. The other folds were tested for clarification on whether the classifiers were over-trained. These results were only for testing purposes and are not shown in the following sections.

All experiments were performed on the Notebook with 16GB of RAM DDR4, 8-core processor AMD Ryzen 7 5800H (3.2GHz, TB 4.4GHz, 16 thread), and graphics card NVIDIA GeForce RTX 3050 Ti 4GB GDDR6.

9.1 Intra-subject results

The volume of these results is huge. For each subject, 26 models of each classifier were trained. This means that a total of 1066 models were trained for each scenario. The total computing time for all scenarios is 85 hours.

Each subject has a different size of data. The number of samples ranges from 18 (subject 24) to 224 (subject 23) for binary and from 14 (subject 24) to 216 (subject 36) for multi-class classification. The difference between binary and multi-class is caused by the implementation of the MNE library. During dropping samples in the multi-class part, some samples are cut out. Subject 24 has 0 samples which is the minimum and can be seen in Tables B.1, B.2, B.3, B.4, B.5, B.6, from the Appendix B. It is because data could not be adequately split into training and testing parts for 10-fold.

Tables B.1, B.2, B.3, B.4, B.5, B.6, in Appendix B, have the following structure. One table has 6 data columns and 41 data rows. Each row represents the accuracies of classifiers on data from one subject and, in the last column, the number of samples for that subject. As stated in Chapter 7, the binary classification has only one type of feature vector, so the results in this case have one table. On the other hand, the multi-class classification has two feature vectors, so it has two tables. Each table is subsequently divided into four different groups. Each group represents one performed dataset. In bold are the best accuracies for each group and each classifier. If there are two or more best accuracies, all are in bold.

For better referring to groups in tables, there is implemented a color-coding:

- 1. group Blue - 14 subjects from Mochura Thesis [18].
- 2. group Gray - 10 subjects from Saleh Thesis [19].
- 3. group Orange - 5 subjects from Kodera Thesis [20]
- 4. group Green - 12 subjects from Farabbi paper [26].

This partition was done to determine whether data from multiple measurements performed similarly or if there were some differences.

9.1.1 Time-series representation and binary classification

Table B.1 presents the accurate classification results using time-series representation and binary classification parameters. Table 9.1 displays all subjects' average accuracy, training times, and classification time to provide further details.

In the blue group, subject 09 had the best performance with 75 % accuracy using MLP, resulting in the best outcome. This subject has only 36 samples in the data, which implies that the result of this test was just luck, but interestingly, cross-validation could not negate this. Subjects 10 and 12 have slightly lower accuracy than subject 09. Their performance is not as great as subject 09. Both have around 61 % with the MLP classifier. These results thus can be supposed to be more trustworthy ones because subjects 10 and 12 have a higher number of samples.

The gray group has pretty much the same pattern as the blue group. There is subject 22 with 72,50 % accuracy from the MLP but with 40 samples. But the other results of this subject are not that good. Subject 23 has better average accuracy across all classifiers on 224 samples.

The orange group is the smallest, but all subjects have many samples. The best-performing subject, 27, has an accuracy of 65,91 % from the LSTM and has almost all other best scorers from the group except one. All is achieved with the second highest number of samples of 220.

The last green group has the best results of all groups. The best result is subject 35, with an accuracy of 75,71 % on the Transformer. However, as previously mentioned, this subject has only 70 samples. Subjects 37, 39, and 40 have great results with a high number of samples. Even subjects 37 and 40 have an average accuracy above 68 %.

There are no significant differences within Kodera_29 (Blue, Gray, and Orange group). Subjects from Farabbi_12 have slightly better results than subjects from Kodera_29.

On the average, this test performed poorly. Accuracies from Table 9.1 show a maximum average accuracy of 54,80 % for the MLP classifier, which is not more of a coin flip. In the training time, the statistical classification excels with 9 ms of training. This is a huge difference when compared to Transfomer performance with 58,9 s of training time. Nevertheless, both classifiers have similar classification results. The CNN, LSTM, and MLP classifiers used a comparable amount of resources during training and provided similar results.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	52,77	52,22	54,80	51,05	51,88
Time of training (s)	1,390	3,049	4,912	0,009	58,902
Time of classification (ms)	42,184	37,995	43,542	0,026	75,211

Table 9.1: Table shows the model's average accuracy from all subjects, training time, and classification time for each classifier. The combination of parameters is time-series representation and binary classification.

9.1.2 Time-series representation and multi-class classification

The tables representing classification accuracy with time-series representation and multi-class classification parameters are shown in Tables B.2 and B.3. Table 9.2 displays the average accuracy, time taken by each classifier for training all subjects, and classification time with unsorted feature vectors, while Table 9.3 shows the same metrics for all subjects with sorted feature vectors.

The blue group's best performer was subject 3, with an accuracy of 56,25 % for unsorted vectors and an accuracy of 65,71 % for sorted vectors for the LSTM classifier on data with 82 samples. The other results are bad except for subject 10. This subject has achieved, with a sorted vector, an accuracy of 50 % from the CNN and the LSTM but on only 51 samples in the dataset.

The gray group has exciting results. For unsorted vectors, subject 22 was the best, with an accuracy of 53,33 % from the statistical approach. For sorted vectors, it was subject 20 with an accuracy of 45,45 % from the

LSTM, the MLP, and the statistical approach. Both subjects have a smaller pool of samples, 31 samples for subject 22 and 38 samples for subject 20. The overall best subject from this group is subject 15, which showed an accuracy of 36,73 % for unsorted vectors and an accuracy of 42,80 % for sorted ones. Another indication is his number of samples, which is 115.

The orange group is the exact opposite of the gray group. Sorting the vectors leads to lower overall results. The best accuracy of subject 26 goes from 48,46 % from the CNN to 42,50 %. This group has two candidates for the best results. Subject 26 has high accuracies with fewer samples, and his results are not stable across all classifiers. Subject 27 achieved a lower best accuracy with a higher number of samples, and his results achieved better accuracies on average.

The green group shows resistance to the sorting of vectors. Sorting helps subject 30 to increase the accuracy of the CNN from 56,25 % to 61,11 %, but the performance of other classifiers is too bad on his 48 samples. It seems subjects 37 and 40 again performed great across classifiers with their high number of samples. Subject 37 is the best classifier for the Transformer, with 58,00 % on unsorted vectors and 57,00 % on sorted vectors. The subject has the best classifier, the MLP, with an accuracy of 58,00 % on unsorted vectors, and the Transformer with 56,32 % accuracy on sorted vectors.

There is no big difference between groups. However, there are differences across subjects.

On the average, this test performed poorly. Accuracies from Tables 9.3 and 9.2 show a maximum average accuracy of 35,87 % for the MLP classifier with sorted vectors. In the training time again, the statistical classification excelled with 9 ms of training. The overall performance of this approach is lower than that of other classifiers, but there are considerable differences in the resources used (0,009 ms of training time). The Transformer performance can be comparable with other classifiers, but the number of resources used (58,902 s training time) is vastly different. The CNN, LSTM, and MLP classifiers used a comparable amount of resources during training and provided similar results.

9.1.3 Frequency representation and binary classification

Tables B.4, 9.4 show the classification accuracy and average accuracy and average training and classification times for each classifier using frequency representation and binary classification.

Subject 09 is the best-performing with 66,67 % accuracy from the Trans-

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	34,58	32,25	34,69	32,12	34,04
Time of training (s)	1,439	3,088	4,403	0,009	55,704
Time of classification (ms)	42,09	39,17	44,19	0,04	75,02

Table 9.2: Table shows the model's average accuracy, training time, and classification time for each classifier with unsorted feature vectors. The combination of parameters is time-series representation and multi-class classification.

Metrics	CNN sorted	LSTM sorted	MLP sorted	SC sorted	Transformer sorted
Average accuracy (%)	35,60	34,71	35,87	31,86	34,41
Time of training (s)	1,436	2,986	4,255	0,009	55,317
Time of classification (ms)	42,52	42,30	41,23	0,04	74,00

Table 9.3: Table shows the model's average accuracy, training time, and classification time for each classifier with sorted feature vectors. The combination of parameters is time-series representation and multi-class classification.

former classifier but only on 36 samples from the blue group. Subjects 10 and 12 were in second place in overall performance. Both subjects have the MLP as the best classifier. Subject 10 achieved 62,86 % accuracy with 214 samples, and Subject 12 had 61,67 % accuracy with 188 samples.

In the gray group, the best subjects are 15, 19, and 23. Despite Subject 15 not having achieving the highest scores, his overall performance is at the same level as that of Subjects 19 and 23. Subject 19 achieved an accuracy of 68,33 % from the LSTM classifier with 62 samples. Subject 23 has 61,82 % as well from the LSTM classifier with 224 samples.

The situation is the same in the orange group. Subject 27 outperformed the other subjects in 4 out of 5 classifiers with his 220 samples. His best-achieved accuracy is 62,72 % from the LSTM classifier. Subject 26 is in second place with his 60,56 % accuracy from the MLP classifier with 182 samples.

At first glance, Subject 31 has the highest accuracy of 62,86 % from the LSTM with 70 samples. However, Subject 39 performs better across all classifiers with 62,22 % accuracy from the MLP with 182 samples.

It seems that the frequency spectrum negates the differences between groups.

Again, this test performed poorly. 3 out of 5 accuracies from Table 9.4 are under 50 %. Regarding average training time, the statistical approach is the fastest at 7 ms. On the opposite of the spectrum is the Transformer with 43,003 s average training time. The average training time for other classifiers is in seconds. There is the same situation for the metric of the average time of classification. The fastest is the statistical approach, the transformer is the slowest. CNN, LSTM, and MLP have similar performance and it is between these values.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy	49,29	51,87	52,21	49,01	49,73
Time of training (s)	0,994	2,792	3,588	0,007	43,003
Time of classification (ms)	39,63	38,28	41,47	0,02	63,41

Table 9.4: Table shows the model's average accuracy, training time, and classification time for each classifier. The combination of parameters is frequency representation and binary classification.

9.1.4 Frequency multi-class

The accuracy of classification with parameters frequency representation and multi-class classification are represented by Table B.1. Table 9.5 shows each classifier's average accuracy, training times, and classification times for all subjects with unsorted feature vectors. Table 9.6 shows the same metrics for all subjects with sorted feature vectors.

The blue group was dominated by subject 13 as far as his/her results were concerned results. In unsorted vectors, he did not get the best result of 66,67 % from the statistical approach of Subject 06 with 60 samples but got the second-best accuracy of 65,00 % from the same classifier. This subject got the best results on average in both unsorted and sorted vectors with his 101 samples. In sorted vectors, this subject got an accuracy of 71,25 % from the LSTM and the Transformer.

In the gray group, it is the same situation as in the blue group. Subject 16 was the most successful so far. These subject results are outstanding because he achieved the best accuracies across all classifiers in unsorted and sorted vectors. These results are also supported by his 140 samples, which is a higher number of samples in the whole dataset. From unsorted vectors, this subject gets an accuracy of 75,71 % from the MLP, and in sorted vectors, it gets 73,33 % accuracy from the statistical approach.

As in previous results, the orange group subject with the highest accuracy is Subject 27 in both unsorted and sorted vectors. From unsorted vectors, this Subject gets 57,50 % accuracy from the MLP with his 165 samples. The

Statistical approach was able to achieve an accuracy of 58,00 % in sorted vectors. These results are not as good as results from previous groups.

The green group is nothing special from other groups. The best subject was subject 31, with an accuracy of 60,00 % from the Transformer with his 70 samples on sorted vectors. In unsorted vectors was the best Subject 33 with 52,73 % accuracy from the CNN.

There are no big differences between groups. The main difference is between subjects.

On the average, this test performed poorly. Accuracies from Tables 9.6 and 9.5 show a maximum average accuracy of 44,33 % for the statistical approach classifier with sorted vectors. Relationships of other metrics are the same here.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	41,10	42,82	42,61	44,33	38,07
Time of training (s)	1,439	3,088	4,403	0,009	55,704
Time of classification (ms)	39,17	37,34	36,16	0,01	42,30

Table 9.5: Table shows the model's average accuracy, training time, and classification time for each classifier with unsorted feature vectors. The combination of parameters is frequency representation and multi-class classification.

Metrics	CNN sorted	LSTM sorted	MLP sorted	SC sorted	Transformer sorted
Average accuracy (%)	42,26	42,17	42,43	43,59	42,32
Time of training (s)	1,436	2,986	4,255	0,009	55,317
Time of classification (ms)	35,45	34,86	35,94	0,01	42,16

Table 9.6: Table shows the model's average accuracy, training time, and classification time for each classifier with sorted feature vectors. The combination of parameters is frequency representation and multi-class classification.

9.1.5 Discussion

In this test scenario, binary classification performed better than multi-class classification. This is a predictable outcome.

The more interesting result is from data representation. Firstly, time-series representation has better results than frequency representation in the case of binary classification. time-series representation achieved higher accuracy but on subjects with fewer samples. The maximum accuracy of 75,71 % from Subject 35 from the Transformer and the average accuracy of 54,80 % from MLP. On the other hand, frequency representation has lower maximum values in both maxima (68,73 %, Subject 19, LSTM) and on average (52,21 %, MLP).

In multi-class classification, the situation is reversed. The time series achieved the best outcome from Subject 03 with an accuracy of 65,71 % from the LSTM. The best average accuracy of 35,87 % comes from MLP. The frequency representation was better here. Subject 16, with an accuracy of 73,33 % from the statistical approach, was the best result, which is almost 8 % more. On average, it achieved 44,33 % accuracy from the Statistical approach, which is again almost 8 % more. These results show that different data representations are suitable for different classifications.

In comparison with previous works from Chapter 3, these tests, on average, have lower performance, but in the best cases, they are on the same level. The results from Kodera's work are in Table 9.7.

Representation	Binary	Multi-class
Time-series	76,00 %	58,57 %
Frequency	65,17 %	50,81 %

Table 9.7: Results from Koderas work on the same combination of parameters. [20]

9.2 Kodera_29 results

This section covers the results of the Kodera_29 dataset. The results for time-series representation are presented in Table 9.8 for binary classification. The LSTM classifier achieved the highest accuracy with 54.44 %. The Statistical classification had the fastest training time. The Transformer classifier was the slowest but achieved the second-highest accuracy.

Below is a summary of the results for multi-class classification and time-series representation. Tables 9.9 and 9.10 show the performance of unsorted and sorted vectors, respectively. The LSTM classifier achieved the highest accuracy of 36.88 %. On the other hand, the statistical classification had the fastest training time, while the transformer had the slowest training time and achieved the second-best accuracy.

Table 9.11 presents the results for frequency representation for binary classification. The MLP classifier achieved the highest accuracy at 59,64 %.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	53,50	**54,44**	53,75	49,26	53,69
Time of training (s)	17,86	40,72	197,40	0,24	1667,81
Time of classification (ms)	38,47	39,18	72,32	0,04	50,41

Table 9.8: The table shows the results of the scenario with the Kodera_29 dataset, time-series representation, and binary classification. Average accuracy, training time, and classification time for each classifier are shown.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	35,97	34,29	34,25	34,10	35,05
Time of training (s)	13,5	32,9	124,1	0,2	254,3
Time of classification (ms)	38,44	39,51	40,71	0,04	48,20

Table 9.9: Part 01. The table shows the results of the scenario with the Kodera_29 dataset, time-series representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown.

In terms of training time, the statistical classifier was the fastest. On the other hand, the transformer classifier took the longest time to train, but it secured the second-highest accuracy.

For multi-class classification and frequency representation, there are Tables 9.12 for unsorted vectors and 9.13 for sorted ones. The best classifier in the MLP with 41,21 %. The fastest training time has the statistical classification. The transformer was the slowest but got the second place in accuracy.

9.2.1 Discussion

In binary classification, the results when using frequency representation achieved a higher accuracy than the results when using time-series representation. The difference is only around five % of accuracy. This outcome is opposite to results in intra-subject results from the previous Section 9.1.

The multi-class classification has better results from frequency representation where accuracies are reaching, in most cases, above 40 %. On the other hand, the time-series representation with the highest accuracy of 36,88 % is worse. There is the same pattern as it is described in Section 9.1.

To compare these results with previous works, see Table 9.14, which sums up results from the latest work from Kodera [20] alongside results from this section. All achieved results are at a lower level of accuracy. The closest to previous work is binary classification on frequency representation, where the difference is around six % of accuracy. The results of the time-series

Metrics	CNN sorted	LSTM sorted	MLP sorted	SC sorted	Transformer sorted
Average accuracy (%)	36,73	**36,88**	35,66	34,49	35,26
Time of training (s)	12,5	33,2	128,2	0,2	311,6
Time of classification (ms)	38,14	39,17	41,08	0,04	170,01

Table 9.10: Part 02. The table shows the results of the scenario with the Kodera_29 dataset, time-series representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	58,24	58,48	**59,64**	55,62	56,61
Time of training (s)	6,63	7,26	4,02	0,05	173,18
Time of classification (ms)	37,12	38,23	63,16	0,02	50,02

Table 9.11: The table shows the results of the scenario with the Kodera_29 dataset, frequency representation, and binary classification. Average accuracy, training time, and classification time for each classifier are shown.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	41,03	40,00	**41,21**	39,52	36,15
Time of training (s)	4,48	7,73	3,34	0,03	131,46
Time of classification (ms)	38,21	37,92	39,07	0,03	51,37

Table 9.12: Part 01. The table shows the results of the scenario with the Kodera_29 dataset, frequency representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown.

representation are both down by 22 %.

From these results, there was confusion about what happened. So, one classifier was chosen to see where the problem is. The chosen classifier was the MLP with the same architecture as in Kodera's work [20] for help pointing out the problem. The results are surprising because these results are lower as well. The first idea was that the sampling rate causes it. The original dataset has a sampling rate of 500 Hz, but to be able to add samples from Farabbi, this rate was decreased to half 250 Hz, as shown in Table 6.1 in Chapter 6. This decrease with high probability causes this lower accuracies.

Metrics	CNN sorted	LSTM sorted	MLP sorted	SC sorted	Transformer sorted
Average accuracy (%)	41,07	39,78	40,85	40,51	36,88
Time of training (s)	5,96	8,77	3,71	0,03	138,96
Time of classification (ms)	38,51	37,07	38,81	0,03	168,58

Table 9.13: Part 02. The table shows the results of the scenario with the Kodera_29 dataset, frequency representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown.

	Binary		Multi-class	
Work	This	Kodera	This	Kodera
Time-series	54,44 %	76,00 %	36,88 %	58,57 %
Frequency	59,64 %	65,17 %	41,21 %	50,81 %

Table 9.14: The best results of this Section compared to Kodera´s results.

9.3 Farabbi_12 results

This section covers the results of the Kodera_29 dataset on classifiers, which were described in Chapter 8. It's unnecessary to repeat that the Statistical approach was the fastest, and the Transformer was the slowest. This applies to all the following results.

In the setup of binary classification and time-series representation, the CNN gets the highest score of 63,35 % accuracy; see Table 9.15.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	**63,35**	60,26	60,29	61,48	60,77
Time of training (s)	6,6	12,3	53,0	0,1	115,9
Time of classification (ms)	37,64	66,91	41,65	0,04	48,46

Table 9.15: The table shows the results of the scenario with the Farabbi_12 dataset, time-series representation, and binary classification. Average accuracy, training time, and classification time for each classifier are shown.

The tables in Tables 9.16 and 9.17 show the results of multi-class classification for unsorted and sorted vectors, respectively. There is the best classifier, the Transformer, with 48,90 % accuracy from sorted vectors.

In the frequency representation and binary classification, the best classifier is the CNN with 51,81 % accuracy; see Table 9.18.

The results of multi-class classification for frequency representation are shown in Tables 9.19 and 9.20 for unsorted and sorter vectors. The best

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	47,68	46,13	45,48	41,74	47,03
Time of training (s)	7,1	14,8	56,5	0,1	145,8
Time of classification (ms)	39,01	38,86	42,84	0,04	48,85

Table 9.16: Part 01. The table shows the results of the scenario with the Farabbi_12 dataset, time-series representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown.

Metrics	CNN sorted	LSTM sorted	MLP sorted	SC sorted	Transformer sorted
Average accuracy (%)	47,34	41,21	44,55	41,10	**48,90**
Time of training (s)	7,4	16,9	55,0	0,1	117,6
Time of classification (ms)	37,90	38,56	43,24	0,04	1165,93

Table 9.17: Part 02. The table shows the results of the scenario with the Farabbi_12 dataset, time-series representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	**51,81**	51,23	51,10	49,10	51,10
Time of training (s)	2,85	4,07	1,77	0,02	57,06
Time of classification (ms)	37,22	37,98	56,76	0,02	49,78

Table 9.18: The table shows the results of the scenario with the Farabbi_12 dataset, frequency representation, and binary classification. Average accuracy, training time, and classification time for each classifier are shown.

classifier is the transformer, with achieved accuracy of 49,10 %.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	43,29	41,03	41,68	34,71	**49,10**
Time of training (s)	2,88	4,74	1,95	0,02	66,55
Time of classification (ms)	36,61	37,86	37,98	0,02	49,33

Table 9.19: Part 01. The table shows the results of the scenario with the Farabbi_12 dataset, frequency representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown.

Metrics	CNN sorted	LSTM sorted	MLP sorted	SC sorted	Transformer sorted
Average accuracy (%)	43,77	42,08	42,27	33,90	48,70
Time of training (s)	3,30	4,96	1,93	0,02	56,93
Time of classification (ms)	36,93	59,96	37,36	0,02	148,38

Table 9.20: Part 02. The table shows the results of the scenario with the Farabbi_12 dataset, frequency representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown.

9.3.1 Discussion

As anticipated, these results are different from Kodera_29 results in Section 9.2. The time-series representation is more suitable for binary classification in this dataset. This is the same as the Intra-subject Section 9.1 and opposite to Kodera_29 Section 9.2. In the frequency representation, there is no big difference in the results. The best results are from the transformer, with a difference only of 0,2 %.

In comparison to results from section 9.2, these results are higher in 3 out of 4 cases (see Table 9.21).

	Binary		Multi-class	
Dataset	Farabbi_12	Kodera_29	Farabbi_12	Kodera_29
Time-series	63,35 %	54,44 %	48,90 %	36,88 %
Frequency	51,81 %	59,64 %	49,10 %	41,21 %

Table 9.21: The best results of this section compared to results from Section 9.2.

9.4 Hrabik_41 results

This Section covers the results of the dataset composed from Kodera_29 and Farabbi_12 on classifiers described in Chapter 8.

The results for time-series representation are presented in Table 9.22 for binary classification. The CNN classifier achieved the highest accuracy with 55,68 %.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	**55,68**	54,61	55,32	53,16	53,20
Time of training (s)	27,4	66,0	354,3	0,4	648,5
Time of classification (ms)	38,03	38,67	74,20	0,05	48,20

Table 9.22: The table shows the results of the scenario with the All Subjects dataset, time-series representation, and binary classification. Average accuracy, training time, and classification time for each classifier are shown.

The tables for unsorted and sorted vectors display the results of the same representation and multi-class classification, as shown in Tables 9.23 and 9.24, respectively. There is the best classifier, the Transformer, with 37,41 % accuracy from sorted vectors.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	37,55	37,09	37,09	36,32	38,16
Time of training (s)	23,2	64,1	384,6	0,3	1464,0
Time of classification (ms)	38,82	39,63	46,61	0,05	51,81

Table 9.23: Part 01. The table shows the results of the scenario with the All Subjects dataset, time-series representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown.

Metrics	CNN sorted	LSTM sorted	MLP sorted	SC sorted	Transformer sorted
Average accuracy (%)	37,99	36,31	36,29	36,66	**38,41**
Time of training (s)	20,2	60,9	297,8	0,3	1297,0
Time of classification (ms)	38,82	39,20	41,97	0,06	51,38

Table 9.24: Part 02. The table shows the results of the scenario with the All Subjects dataset, time-series representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown.

Table 9.25 shows results for binary classification on frequency representation. The best classifier is the CNN, with an accuracy of 56,42 %.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	**56,42**	54,24	55,47	54,80	53,53
Time of training (s)	9,36	9,03	4,57	0,06	238,06
Time of classification (ms)	37,74	37,66	58,58	0,02	48,99

Table 9.25: The table shows the results of the scenario with the All Subjects dataset, frequency representation, and binary classification. Average accuracy, training time, and classification time for each classifier are shown.

For multi-class classification and the same representation, the results are in Tables 9.26 and 9.27 for unsorted and sorted vectors, respectively. The best classifier is the LSTM, with an accuracy of 41,38 % for unsorted vectors.

Metrics	CNN	LSTM	MLP	SC	Transformer
Average accuracy (%)	**41,38**	39,95	38,18	32,49	38,74
Time of training (s)	8,26	10,29	6,01	0,05	194,69
Time of classification (ms)	37,50	37,56	37,53	0,02	40,21

Table 9.26: Part 01. The table shows the results of the scenario with the All Subjects dataset, frequency representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown.

Metrics	CNN sorted	LSTM sorted	MLP sorted	SC sorted	Transformer sorted
Average accuracy (%)	34,51	34,53	32,43	29,25	37,66
Time of training (s)	8,29	9,27	3,57	0,05	212,79
Time of classification (ms)	37,02	37,45	37,08	0,02	39,93

Table 9.27: Part 02. The table shows the results of the scenario with the All Subjects dataset, frequency representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown.

9.4.1 Discussion

These results show some improvement from the base dataset Kodera_29. Table 9.28 shows the best results from each category. For comparison with Section 9.2, these results have improved in 3 out of 4 best cases. There is only a slight improvement. On the other hand, the sum of improvement is 2,94 %, and the degradation is 3,22 %. So, in the end, adding new data from different sources is more harmful than beneficial.

In the same manner, these results can be compared to the original results from Kodera (see Table 9.28). Because there is no improvement from dataset Kodera_29 as described above, there is no need to delve more into it.

The results achieved on the Farabbi_12 dataset are better than those on the Kodera_29 and even composed datasets (see Table 9.28).

Binary			
Dataset	Hrabik_41	Farabbi_12	Kodera_29
Time-series	55,68 %	63,35 %	54,44 %
Frequency	56,42 %	51,81 %	59,64 %
Multi-class			
Dataset	Hrabik_41	Farabbi_12	Kodera_29
Time-series	38,41 %	48,90 %	36,88 %
Frequency	41,38 %	49,10 %	41,21 %

Table 9.28: The best results of this section compared to results from Section 9.2 and Section 9.3.

9.5 Discussion

All achieved results provided in Sections 9.2, 9.3, and 9.4 are lower than results from Kodera's work. It is shown in Table 9.14 where there is a difference of 20 % of accuracy in some cases. Several possible causes of this outcome were found.

The first was the sampling rate. So, there was a created test on the Kodera_29 dataset with an original sampling rate of 500 Hz. The results of this scenario are shown in Table 9.29. These results are basically the same as in Table 9.14 on the 250 Hz sampling rate described in Section 9.2. So, the sampling rate is not the cause of this decline in accuracy.

	Binary	Multi-class
Time-series	53,71 %	38,19 %
Frequency	60,36 %	42,32 %

Table 9.29: The best results of the original sampling rate of 500 Hz on the Kodera_29 dataset.

The next cause could be the architecture of neural networks. To test this theory, the project of Kodera's work was taken. This test only focused on time-series representation with binary and multi-class classification. The results of this test are shown in Table 9.30. These results are surprising because they have not achieved the proposed results from work from Kodera [20]. The best results are only the difference of 6% from the results of this thesis from Table 9.7. This 6% can be caused by architecture. There is no simple explanation for the difference of 20% from the proposed results.

Classification	CNN	LSTM	MLP
Binary	60,94 %	55,28 %	49,59 %
Multi-class	40,04 %	40,70 %	33,84 %

Table 9.30: Classification results using the original sampling rate and architecture from [20].

The comparison with works of Mochura [18] and Saleh [19] is pointless here. Not only did they use different approaches, but they had even better results than Kodera [20].

10 Conclusion

In Chapter 2, there were introduced principles of EEG. After that, in Chapter 3, there were shown works that were done on this topic by the neuroinformatics group at the University of West Bohemia. The overview of available free datasets was described in Chapter 4. In Chapter 5, there were described classification techniques. Chapters 6 and 7 describe analysis of gathered data. Implementation of proposed tests was shown in Chapter 8. Accomplished results were presented in the Chapter 9.

The intra-subject results showed a high variance in the performance of subjects. This effect can be caused by several things. Firstly, there is a difference in the number of samples for each subject. This is causing some classifiers to have stable results on data from subjects with a higher number of samples. Secondly, these results indicate that there is a big difference between subjects. Some subjects have stable performance through all tests, and some have better results for different setups. This shows that each subject is unique and needs to be treated differently. Lastly, there can be a subject that simply cannot produce proper EEG signals for MI. For instance, it is Subject 11, which has a high number of samples but very low performance in all tests.

Results from inter-subject testing even support these findings. Aggregating data of subjects to a big dataset does not generally achieve better results. The idea that a similar dataset from another source could bring more samples and thus increase the accuracy was wrong. The experiment was similar enough, but something caused a more extensive accuracy loss than the increase of accuracy gained by more samples. The first possible cause was the sampling rate, which was cut in half for the original dataset Kodera_29 to have data with the exact sizes of samples. Further testing found that this was not the leading cause. Data were resampled with the original rate, and the results were not that much different. In the end, the various architectures could lead to bigger differences in classification results than changes in the sampling rate.

In light of these findings, there are a few possible ways of progress. The first is to go back to representations via ERD/ERS from work from Mochura [18]. The second way is to enhance the measurement to get more samples for future subjects. The last possible way could be to choose one neural network, from the results, it could be MLP or CNN, and try to find the best architecture via testing different settings of activation functions, normalization, number of neurons, layers, or even combination of architectural styles of NNs.

The source codes for this thesis are available on a GitHub repository: https://github.com/Hrabikv/Diploma_thesis.

List of Figures

2.1 Electrode locations of International 10-20 system for EEG (electroencephalography) recording. Image source: [3] Highlighted electrodes C3 and C4 are on parts for controlling movements of the right and left hand. 10

5.1 Example of neuron structure. The inputs with weights are on the left side. In the middle is the sum. The activation function with the result is on the right side. Source: [40] . . 28
5.2 Example of ANN structure. The input layer is on the left side. In the middle are hidden layers. The output layer is on the right side. Source: [38] 29
5.3 Example of CNN structure. The input layer is on the left side. In the middle, there are Convolutional layers and Pooling layers. The output layer is on the right side. Source: [43] . . 30
5.4 Structure of LSTM. The recurrent connection of the cells is shown. Source: [45] 31
5.5 The structure of the cell of the LSTM. The same cell is shown in the three different stages. Source: [44] 32
5.6 Structure of the Transformer. The encoder is on the left side, and the decoder is on the right side. Source: [47] 33

6.1 This is an example of the structure of the Kodera-29 dataset. Within the root folder are directories of each measurement labelled by the date of measurement. Inside these directories, there are sets of three data files corresponding to each subject involved in the measurement. 36
6.2 Example of the structure of data for one subject. 37

8.1 The structure of the project. 41
8.2 Example of computed pivots. Each point represents an EEG signal fingerprint constructed by dimensionality reduction Common Spatial Patterns (CSP) algorithm [54]. The blue color points represent left-hand samples, the red color points represent the rest, and the green color points represent right-hand samples. The purple color points are the pivots of each class. 42
8.3 Structure of MLP. 45
8.4 Structure of CNN. 45

8.5 Structure of LSTM. 46
8.6 Structure of Transformer. 46

A.1 Directory structure of the project. 79
A.2 Configuration file of the project. 81

List of Tables

3.1 Overview of information from individual works. It contains the name of the author, the number of people in the dataset, the method of classification used, and the best-achieved result. 17

4.1 Assigning the aliases of obtained data sets. 19
4.2 Comparison of available datasets. 25

6.1 Table of parameters of datasets alongside their combination. 34

9.1 Table shows the model's average accuracy from all subjects, training time, and classification time for each classifier. The combination of parameters is time-series representation and binary classification. 49
9.2 Table shows the model's average accuracy, training time, and classification time for each classifier with unsorted feature vectors. The combination of parameters is time-series representation and multi-class classification. 51
9.3 Table shows the model's average accuracy, training time, and classification time for each classifier with sorted feature vectors. The combination of parameters is time-series representation and multi-class classification. 51
9.4 Table shows the model's average accuracy, training time, and classification time for each classifier. The combination of parameters is frequency representation and binary classification. 52
9.5 Table shows the model's average accuracy, training time, and classification time for each classifier with unsorted feature vectors. The combination of parameters is frequency representation and multi-class classification. 53
9.6 Table shows the model's average accuracy, training time, and classification time for each classifier with sorted feature vectors. The combination of parameters is frequency representation and multi-class classification. 53
9.7 Results from Koderas work on the same combination of parameters. [20] 54

9.8 The table shows the results of the scenario with the Kodera_29 dataset, time-series representation, and binary classification. Average accuracy, training time, and classification time for each classifier are shown. 55
9.9 Part 01. The table shows the results of the scenario with the Kodera_29 dataset, time-series representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown. 55
9.10 Part 02. The table shows the results of the scenario with the Kodera_29 dataset, time-series representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown. 56
9.11 The table shows the results of the scenario with the Kodera_29 dataset, frequency representation, and binary classification. Average accuracy, training time, and classification time for each classifier are shown. 56
9.12 Part 01. The table shows the results of the scenario with the Kodera_29 dataset, frequency representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown. 56
9.13 Part 02. The table shows the results of the scenario with the Kodera_29 dataset, frequency representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown. 57
9.14 The best results of this Section compared to Kodera´s results. 57
9.15 The table shows the results of the scenario with the Farabbi_12 dataset, time-series representation, and binary classification. Average accuracy, training time, and classification time for each classifier are shown. 57
9.16 Part 01. The table shows the results of the scenario with the Farabbi_12 dataset, time-series representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown. 58
9.17 Part 02. The table shows the results of the scenario with the Farabbi_12 dataset, time-series representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown. 58

9.18 The table shows the results of the scenario with the Farabbi_12 dataset, frequency representation, and binary classification. Average accuracy, training time, and classification time for each classifier are shown. 58
9.19 Part 01. The table shows the results of the scenario with the Farabbi_12 dataset, frequency representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown. 58
9.20 Part 02. The table shows the results of the scenario with the Farabbi_12 dataset, frequency representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown. 59
9.21 The best results of this section compared to results from Section 9.2. 59
9.22 The table shows the results of the scenario with the All Subjects dataset, time-series representation, and binary classification. Average accuracy, training time, and classification time for each classifier are shown. 60
9.23 Part 01. The table shows the results of the scenario with the All Subjects dataset, time-series representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown. 60
9.24 Part 02. The table shows the results of the scenario with the All Subjects dataset, time-series representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown. 60
9.25 The table shows the results of the scenario with the All Subjects dataset, frequency representation, and binary classification. Average accuracy, training time, and classification time for each classifier are shown. 61
9.26 Part 01. The table shows the results of the scenario with the All Subjects dataset, frequency representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown. 61
9.27 Part 02. The table shows the results of the scenario with the All Subjects dataset, frequency representation, and multi-class classification. Average accuracy, training time, and classification time for each classifier are shown. 61
9.28 The best results of this section compared to results from Section 9.2 and Section 9.3. 62

9.29 The best results of the original sampling rate of 500 Hz on the Kodera_29 dataset. 63
9.30 Classification results using the original sampling rate and architecture from [20]. 63

B.1 Table of average classification accuracies of all classifiers for intra-subject test with parameters Time-series representation and binary classification. 83
B.2 Part 01 of the table of average classification accuracies with parameters Time-series representation, multi-class classification of all classifiers for the intra-subject test. 84
B.3 Part 02 of the table of average classification accuracies with parameters Time-series representation, multi-class classification of all classifiers for the intra-subject test. 85
B.4 Table of average classification accuracies with parameters Frequency representation, binary classification of all classifiers for intra-subject test. 86
B.5 Part 01 of the table of average classification accuracies with parameters Frequency representation, multi-class classification of all classifiers for intra-subject test. 87
B.6 Part 02 of the table of average classification accuracies with parameters Frequency representation, binary classification of all classifiers for intra-subject test. 88

Bibliography

[1] S. C. S. M. Joseph I. Sirven MD. (2023) Electroencephalography (eeg). [Online]. Available: https://www.epilepsy.com/diagnosis/eeg

[2] H. J. C. George H. Klem (USA), Hans Otto LuÈders (USA) and C. E. (Germany). (1999) The ten±twenty electrode system of the international federation. [Online]. Available: https://media.journals.elsevier.com/content/files/clinph-chapter11-14082757.pdf

[3] Wikipedia. (2023) 10–20 system (eeg). Image. [Online]. Available: https://en.wikipedia.org/wiki/10%E2%80%9320_system_(EEG)

[4] Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. H. Falk, and J. Faubert, "Deep learning-based electroencephalography analysis: a systematic review," *Journal of Neural Engineering*, vol. 16, no. 5, p. 051001, aug 2019. [Online]. Available: https://dx.doi.org/10.1088/1741-2552/ab260c

[5] N. Padfield, J. Zabalza, H. Zhao, V. Masero, and J. Ren, "Eeg-based brain-computer interfaces using motor-imagery: Techniques and challenges," *Sensors*, vol. 19, no. 6, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/6/1423

[6] A. A. Nayak CS, "Eeg normal waveforms." *StatPearls [Internet]*, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK539805/

[7] M. Nunez, P. Nunez, and R. Srinivasan, *Electroencephalography (EEG): neurophysics, experimental methods, and signal processing*, 01 2016, pp. 175–197.

[8] G. E. Chatrian, M. C. Petersen, and J. A. Lazarte, "The blocking of the rolandic wicket rhythm and some central changes related to movement," *Electroencephalography and Clinical Neurophysiology*, vol. 11, no. 3, pp. 497–510, 1959. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0013469459900483

[9] S. Sur and V. K. Sinha, "Event-related potential: An overview," Jan 2009. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3016705/

[10] K. MNakayashiki, M. Saeki, Y. Takata, Y. Hayashi, and T. Kondo, "Modulation of event-related desynchronization during kinematic and kinetic hand movements," *Journal of NeuroEngineering and Rehabilitation*, vol. 11, 2014. [Online]. Available: https://doi.org/10.1186/1743-0003-11-90

[11] G. Pfurtscheller and C. Neuper, “Motor imagery and direct brain-computer communication,” *Proceedings of the IEEE*, vol. 89, no. 7, pp. 1123–1134, 2001.

[12] M. Jeannerod, “Mental imagery in the motor context,” *Neuropsychologia*, vol. 33, no. 11, pp. 1419–1432, 1995, the Neuropsychology of Mental Imagery. [Online]. Available: https://www.sciencedirect.com/science/article/pii/002839329500073C

[13] M. A. Cervera, S. R. Soekadar, J. Ushiba, J. d. R. Millán, M. Liu, N. Birbaumer, and G. Garipelli, “Brain-computer interfaces for post-stroke motor rehabilitation: a meta-analysis,” *Annals of Clinical and Translational Neurology*, vol. 5, no. 5, pp. 651–663, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/acn3.544

[14] WHO. (2023) Rehabilitation. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/rehabilitation

[15] I. Lazarou, S. Nikolopoulos, P. C. Petrantonakis, I. Kompatsiaris, and M. Tsolaki, “Eeg-based brain–computer interfaces for communication and rehabilitation of people with motor impairment: A novel approach of the 21st century,” *Frontiers in Human Neuroscience*, vol. 12, 2018. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnhum.2018.00014

[16] T. Mulder, “Motor imagery and action observation: cognitive tools for rehabilitation,” *Journal of Neural Transmission*, vol. 114, 2007. [Online]. Available: https://doi.org/10.1007/s00702-007-0763-z

[17] B. Krüger, M. Hettwer, A. Zabicki, B. de Haas, J. Munzert, and K. Zentgraf, “Practice modality of motor sequences impacts the neural signature of motor imagery,” *Scientific Reports*, vol. 10, 2020. [Online]. Available: https://doi.org/10.1038/s41598-020-76214-y

[18] P. Mochura, “Movement detection from eeg signals during exercise on a rehabilitation robot [online],” 2021, thesis, University of West Bohemia. [Online]. Available: https://otik.zcu.cz/handle/11025/45196

[19] J. Y. Saleh, “Design of movement detector of measured eeg data [online],” 2022, thesis, University of West Bohemia. [Online]. Available: https://dspace5.zcu.cz/handle/11025/49544

[20] J. Kodera, R. Mouček, P. Mautner, and J. Průcha, *Augmentation of Motor Imagery Data for Brain-Controlled Robot-Assisted Rehabilitation.*, 12 2024, pp. 812–819.

[21] A. Subasi and M. I. Gursoy, “Eeg signal classification using pca, ica, lda and support vector machines,” *Expert Systems with Applications*, vol. 37, no. 12, pp. 8659–8666, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417410005695

[22] J. Kodera, R. Mouček, P. Moutner, P. Mochura, J. Y. Saleh, P. Brůha, J. Šolcová, L. Vařeka, and P. Šnejdar, “Eeg motor imagery,” May 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7893847

[23] J. J, C. J, S. K, K. B, L. B, L. D, L. D, and L. S, “Supporting data for "multimodal signal dataset for 11 intuitive movement tasks from single upper extremity during multiple recording sessions",” 2020. [Online]. Available: http://gigadb.org/dataset/view/id/100788/Sample_page/3

[24] G. Averta, F. Barontini, V. Catrambone, S. Haddadin, G. Handjaras, J. P. O. Held, T. Hu, E. Jakubowitz, C. Kanzler, J. Kühn, O. Lambercy, A. Leo, A. Obermeier, E. Ricciardi, A. Schwarz, G. Valenza, A. Bicchi, and M. Bianchi, “U-limb,” 2020. [Online]. Available: https://doi.org/10.7910/DVN/FU3QZ9

[25] H. Liu and X. Lv, “Eeg datasets of stroke patients,” 9 2023. [Online]. Available: https://figshare.com/articles/dataset/EEG_datasets_of_stroke_patients/21679035

[26] A. Farabbi, F. Ghiringhelli, L. Mainardi, J. M. Sanches, P. Moreno, J. Santos-Victor, P. Figueiredo, and A. Vourvopoulos, “Motor-imagery eeg dataset during robot-arm control,” feb 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5882500

[27] M. Jun, B. Yang, and W. Qiu, “Shu dataset,” 8 2022. [Online]. Available: https://figshare.com/articles/software/shu_dataset/19228725

[28] D. Pauline, R. Aline, R. Sébastien, P. Léa, and L. Fabien, “A large EEG database with users’ profile information for motor imagery Brain-Computer Interface research,” Jun. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8089820

[29] V. Peterson, C. M. Galvan, H. S. Hernadez, and R. Spies, “"motor imagery vs rest - low-cost eeg system",” 2021.

[30] J. R. Stieger, S. A. Engel, and B. He, “Continuous sensorimotor rhythm based brain computer interface learning in a large population,” *Scientific Data*, vol. 8, no. 1, p. 98, Apr 2021. [Online]. Available: https://doi.org/10.1038/s41597-021-00883-1

[31] J.-H. Jeong, J.-H. Cho, K.-H. Shim, B.-H. Kwon, B.-H. Lee, D.-Y. Lee, D.-H. Lee, and S.-W. Lee, "Multimodal signal dataset for 11 intuitive movement tasks from single upper extremity during multiple recording sessions," *Gigascience*, vol. 9, no. 10, Oct. 2020.

[32] J. Ma, B. Yang, W. Qiu, Y. Li, S. Gao, and X. Xia, "A large eeg dataset for studying cross-session variability in motor imagery brain-computer interface," *Scientific Data*, vol. 9, no. 1, p. 531, Sep 2022. [Online]. Available: https://doi.org/10.1038/s41597-022-01647-1

[33] P. Dreyer, A. Roc, L. Pillette, S. Rimbert, and F. Lotte, "A large eeg database with users' profile information for motor imagery brain-computer interface research," *Scientific Data*, vol. 10, no. 1, p. 580, Sep 2023. [Online]. Available: https://doi.org/10.1038/s41597-023-02445-z

[34] V. Peterson, C. Galván, H. Hernández, M. P. Saavedra, and R. Spies, "A motor imagery vs. rest dataset with low-cost consumer grade eeg hardware," *Data in Brief*, vol. 42, p. 108225, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340922004280

[35] J. Stieger, "Human EEG Dataset for Brain-Computer Interface and Meditation," 2 2021. [Online]. Available: https://figshare.com/articles/dataset/Human_EEG_Dataset_for_Brain-Computer_Interface_and_Meditation/13123148

[36] R. Gandhi. (2018) Support vector machine — introduction to machine learning algorithms. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction_to-machine-learning-algorithms-934a444fca47

[37] G. Rohith. (2018) Naive bayes classifier. [Online]. Available: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

[38] M. Musiol. Speeding up deep learning computational aspects of machine learning. [Online]. Available: https://www.researchgate.net/figure/A-general-model-of-a-deep-neural-network-It-consists-of-an\-input-layer-some-here-two_fig1_308414212

[39] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 4, pp. 809–821, 2015.

[40] M. Çakir and G. Çınarer, *MECHANICAL OBJECT PARTS DETECTION USING DEEP LEARNING BASED YOLO MODELS*, 12 2022, pp. 205–226.

[41] M. Mandal, “Introduction to Convolutional Neural Networks (CNN) — analyticsvidhya.com,” https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/, [Accessed 08-03-2024].

[42] IMB, “What are Convolutional Neural Networks? | IBM — ibm.com,” https://www.ibm.com/topics/convolutional-neural-networks, [Accessed 08-03-2024].

[43] S. Saha, “A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way — towardsdatascience.com,” https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53, [Accessed 08-03-2024].

[44] C. Olah, “Understanding LSTM Networks – colah&apos;s blog — colah.github.io,” https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015, [Accessed 10-03-2024].

[45] G. Hull, “Building a Neural Network Zoo From Scratch: The Long Short-Term Memory Network — CallMeTwitch,” https://medium.com/@CallMeTwitch/building-a-neural-network-zoo-from-scratch-the-long-short-term-memory-network-1cec5cf31b7, 2022, [Accessed 10-03-2024].

[46] S. Saxena, “What is LSTM? Introduction to Long Short-Term Memory — analyticsvidhya.com,” https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/, 2024, [Accessed 10-03-2024].

[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2023.

[48] “Transformer Neural Networks: A Step-by-Step Breakdown — builtin.com,” https://builtin.com/artificial-intelligence/transformer-neural-network, 2022, [Accessed 13-03-2024].

[49] G. Giacaglia, “Transformers — towardsdatascience.com,” https://towardsdatascience.com/transformers-141e32e69591, 2019, [Accessed 13-03-2024].

[50] Turing, “The Ultimate Guide to Transformer Deep Learning — turing.com,” https://www.turing.com/kb/brief-introduction-to-transformers-and-their-power#what-is-the-transformer-model?, [Accessed 13-03-2024].

[51] IBM. (2023) What is the k-nearest neighbors algorithm? [Online]. Available: https://www.ibm.com/topics/knn

[52] B. David, *Bayesian Reasoning and Machine Learning.* Cambridge University Press, 2012.

[53] A. Schlögl, "Gdf - a general dataformat for biosignals," 2013.

[54] U. Lal, "Neuroscience Meets Data Science: Exploring Common Spatial Pattern (CSP) and Its Applications in… — medium.com," https://medium.com/geekculture/common-spatial-pattern-and-its-applications-in-the-healthcare-industry-faa4311dab79, [Accessed 06-04-2024].

A User Guide

This chapter is a guide describing setting up the environment and testing different setups for classification. The project is written in Python version 3.10. The project was tested on a laptop with Windows 10.

First, you need all the files from the GitHub repository: https://github.com/Hrabikv/Diploma_thesis. You can download it via GitHub UI, or you can clone it.

A.1 Structure of the repository

The project starts with directory *src*, where all source code files are. You needed data from [22] and [26]. These datasets need to be in folder *data* on the same level as *src* and need to be named **kodera_29** and **farabbi_12**.

This project creates two new directories. The first is *PREPROCESSED DATA FOLDER*, where preprocessed data for repeated usage are saved. The second is the folder *results*. In this directory, there are folders for each scenario and one folder with raw results. Aggregated results are in each scenario folder and are saved as

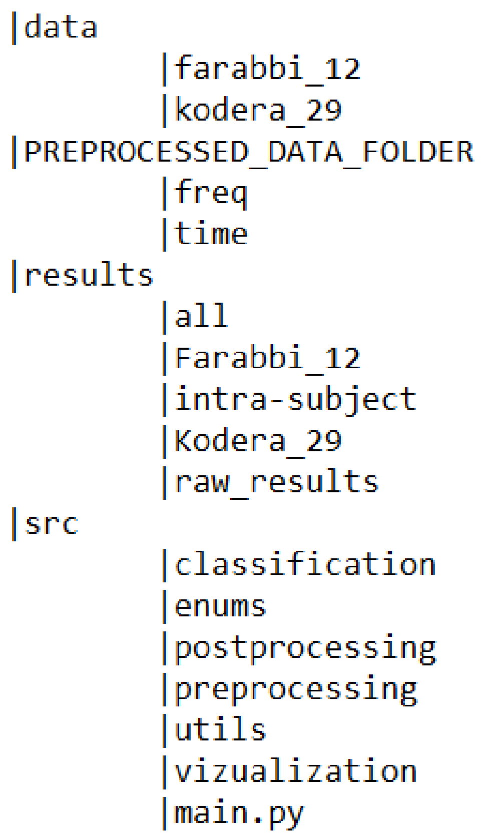
$$< name > \_ < representation >_< classification >_< metric > .xlsx$$

- name - Name of the test (Intra-subject, Kodera_29, Farabbi_12, Hrabik_41).
- representation - Used representation (time-series, frequency).
- classification - Type of classification (binary, multi-class).
- metric - Type of metric (average accuracy, best accuracy, time of training, time of classification). For each metric, there is one file.

Raw results are saved as

$$< classifier > .csv,$$

where the classifier is the name used classifier. Figure A.1 shows the structure of the project with all directories.



```
|data
        |farabbi_12
        |kodera_29
|PREPROCESSED_DATA_FOLDER
        |freq
        |time
|results
        |all
        |Farabbi_12
        |intra-subject
        |Kodera_29
        |raw_results
|src
        |classification
        |enums
        |postprocessing
        |preprocessing
        |utils
        |vizualization
        |main.py
```

Figure A.1: Directory structure of the project.

A.2 Installation

Installation In the main directory, you will find a file called **requirements.txt**. This file contains a list of all the external libraries that were used in the project. To get those, you need to install Python 3.10. It is recommended that a virtual environment is created to make sure that there are no collisions in the version of libraries. Run this command to create one:

```
python -m venv venv
```

This created a virtual environment named **venv**. To activate this environment, run this command:

```
venv\Scripts\activate
```

For installation of all libraries to the environment, run this command:

```
pip install -r requirements.txt
```

A.3 Work with the Project

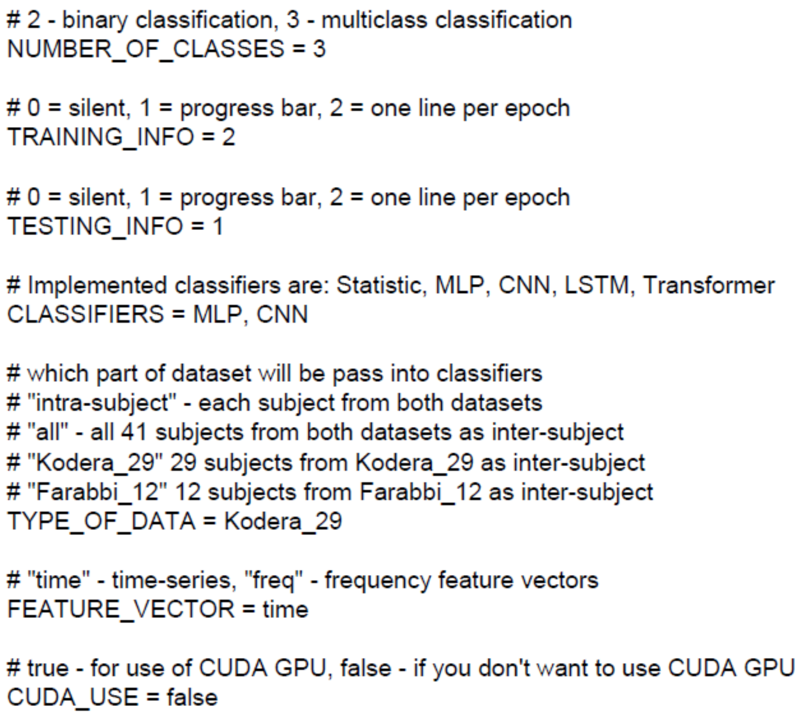
The entry point of the program is **main.py** in the src folder. Make sure you have a data folder in your project. To start a program, run this command:

```
python src/main.py
```

in the same virtual environment as the libraries you installed above.

A.4 Parameters of the Project

The project has seven parameters that you can change without having to open source code files. All parameters are in **config.txt** file. This file is shown in Figure A.2. All these parameters are mandatory. The first is NUMBER_OF_CLASSES, which indicates what type of classification is wanted. The number "2" is for binary classification, and the number "3" is for multi-class classification. The second TRAINING_INFO and third TESTING_INFO parameters are about information printed into a console during the run. The fourth parameter, called CLASSIFIERS, is an array of classification techniques that will be used during classification. This parameter can have multiple values. The fifth parameter, TYPE_OF_DATA, is the choice of the dataset that will be used. The FEATURE_VECTOR parameter is about the representation of the samples of the chosen dataset. The last parameter, CUDA_USE, is the switch of the CUDA Toolkit environment. All possible values of all parameters are shown in Figure A.2.



```
# 2 - binary classification, 3 - multiclass classification
NUMBER_OF_CLASSES = 3

# 0 = silent, 1 = progress bar, 2 = one line per epoch
TRAINING_INFO = 2

# 0 = silent, 1 = progress bar, 2 = one line per epoch
TESTING_INFO = 1

# Implemented classifiers are: Statistic, MLP, CNN, LSTM, Transformer
CLASSIFIERS = MLP, CNN

# which part of dataset will be pass into classifiers
# "intra-subject" - each subject from both datasets
# "all" - all 41 subjects from both datasets as inter-subject
# "Kodera_29" 29 subjects from Kodera_29 as inter-subject
# "Farabbi_12" 12 subjects from Farabbi_12 as inter-subject
TYPE_OF_DATA = Kodera_29

# "time" - time-series, "freq" - frequency feature vectors
FEATURE_VECTOR = time

# true - for use of CUDA GPU, false - if you don't want to use CUDA GPU
CUDA_USE = false
```

Figure A.2: Configuration file of the project.

B Tables of results

ID	CNN	LSTM	MLP	SC	Transformer	# samples
subject_01	49,44	46,11	51,67	51,67	44,44	184
subject_02	37,88	40,91	37,88	40,91	36,36	68
subject_03	50,00	55,45	43,64	47,27	46,36	110
subject_04	50,00	46,25	52,50	42,50	51,25	80
subject_05	51,95	42,86	44,16	42,86	53,25	78
subject_06	48,75	50,00	51,25	51,25	56,25	80
subject_07	36,25	37,50	51,25	42,50	41,25	80
subject_08	46,97	50,00	48,48	43,94	45,45	68
subject_09	55,56	55,56	**75,00**	**58,33**	**61,11**	36
subject_10	**60,00**	59,52	61,43	56,19	56,19	214
subject_11	48,95	45,79	43,16	47,37	48,42	192
subject_12	**60,00**	59,44	61,11	56,67	60,00	188
subject_13	48,46	**60,00**	56,92	43,85	46,15	134
subject_14	57,58	57,58	57,58	51,52	51,52	35
subject_15	**60,67**	**56,00**	58,67	49,33	57,33	154
subject_16	52,78	55,00	53,89	45,00	52,22	186
subject_17	38,33	41,67	50,00	39,17	39,17	126
subject_18	50,71	50,00	51,43	46,43	46,43	148
subject_19	53,33	43,33	58,33	50,00	51,67	62
subject_20	56,00	52,00	50,00	50,00	56,00	50
subject_21	47,50	55,00	48,33	50,83	54,17	126
subject_22	57,50	55,00	**72,50**	50,00	47,50	40
subject_23	58,18	55,45	64,55	**53,64**	**60,00**	224
subject_24	0,00	0,00	0,00	0,00	0,00	18
subject_25	48,33	51,11	54,44	51,67	47,78	184
subject_26	55,00	57,22	58,33	**56,67**	52,22	182
subject_27	**57,27**	**65,91**	**63,18**	54,55	**57,73**	220
subject_28	50,48	51,43	52,38	48,57	49,05	216
subject_29	52,67	53,33	50,00	50,67	48,67	156
subject_30	64,58	62,50	52,08	54,17	52,08	48
subject_31	50,00	45,71	50,00	51,43	47,14	70
subject_32	58,33	47,92	50,00	58,33	60,42	48
subject_33	61,82	52,73	56,36	52,73	45,45	58
subject_34	51,67	65,00	**71,67**	61,67	60,00	64
subject_35	60,00	48,57	70,00	68,57	**75,71**	70
subject_36	61,43	59,05	59,05	64,76	58,10	216
subject_37	**70,00**	**67,50**	66,00	67,00	69,50	204
subject_38	49,50	58,00	51,50	53,50	52,00	208
subject_39	66,11	62,78	67,22	63,33	62,78	182
subject_40	**70,00**	67,00	70,00	**69,50**	66,50	200
subject_41	59,47	54,74	61,05	54,74	59,47	190

Table B.1: Table of average classification accuracies of all classifiers for intra-subject test with parameters Time-series representation and binary classification.

ID	CNN	LSTM	MLP	SC	Transformer	# samlpes
subject_01	30,00	22,31	26,15	27,69	26,92	139
subject_02	20,00	18,00	26,00	10,00	16,00	51
subject_03	**48,75**	**56,25**	**53,75**	32,50	**50,00**	82
subject_04	31,67	33,33	30,00	26,67	36,67	60
subject_05	16,36	32,73	20,00	18,18	27,27	59
subject_06	30,00	33,33	31,67	28,33	31,67	60
subject_07	23,33	25,00	21,67	28,33	28,33	60
subject_08	34,00	30,00	42,00	38,00	34,00	51
subject_09	0,00	0,00	0,00	0,00	0,00	28
subject_10	31,25	31,88	33,13	**38,13**	26,88	161
subject_11	35,00	39,29	35,71	36,43	32,86	144
subject_12	30,00	26,43	35,71	31,43	28,57	140
subject_13	30,00	35,00	35,00	32,00	31,00	101
subject_14	0,00	0,00	0,00	0,00	0,00	27
subject_15	41,82	**41,82**	33,64	35,45	30,91	115
subject_16	29,29	34,29	30,00	33,57	31,43	140
subject_17	32,22	31,11	30,00	34,44	32,22	95
subject_18	39,09	31,82	36,36	33,64	28,18	111
subject_19	25,00	31,82	31,82	29,55	29,55	46
subject_20	30,56	30,56	38,89	36,11	33,33	38
subject_21	34,44	37,78	32,22	30,00	32,22	95
subject_22	**46,67**	33,33	**46,67**	**53,33**	**46,67**	31
subject_23	38,75	32,50	39,38	43,13	33,13	168
subject_24	0,00	0,00	0,00	0,00	0,00	14
subject_25	34,62	31,54	26,92	36,92	27,69	138
subject_26	**48,46**	36,92	35,38	**45,38**	**46,92**	137
subject_27	41,88	**42,50**	**45,00**	41,25	41,25	165
subject_28	30,63	28,75	24,38	28,13	30,00	163
subject_29	33,64	22,73	26,36	34,55	31,82	118
subject_30	**56,25**	27,08	41,67	27,08	39,58	48
subject_31	38,57	30,00	40,00	32,86	38,57	70
subject_32	35,42	37,50	45,83	35,42	39,58	48
subject_33	32,73	30,91	36,36	29,09	40,00	58
subject_34	50,00	33,33	41,67	46,67	41,67	64
subject_35	45,71	32,86	44,29	40,00	51,43	70
subject_36	47,62	44,76	49,05	42,38	50,95	216
subject_37	56,00	51,50	50,50	43,00	**58,00**	204
subject_38	45,50	46,00	53,50	39,50	44,00	208
subject_39	46,67	45,56	52,22	35,56	46,11	182
subject_40	54,00	**53,50**	**58,00**	**48,50**	56,00	200
subject_41	42,11	38,42	41,58	33,68	44,21	190

Table B.2: Part 01 of the table of average classification accuracies with parameters Time-series representation, multi-class classification of all classifiers for the intra-subject test.

ID	CNN sorted	LSTM sorted	MLP sorted	SC sorted	Transformer sorted	# samlpes
subject_01	28,33	33,33	31,67	26,67	31,67	139
subject_02	26,67	23,33	33,33	20,00	20,00	51
subject_03	**51,43**	**65,71**	**60,00**	35,71	**62,86**	82
subject_04	30,00	30,00	26,67	31,67	36,67	60
subject_05	36,00	28,00	26,00	26,00	10,00	59
subject_06	23,33	30,00	38,33	28,33	33,33	60
subject_07	18,33	31,67	30,00	25,00	25,00	60
subject_08	50,00	50,00	40,00	**42,50**	37,50	51
subject_09	0,00	0,00	0,00	0,00	0,00	28
subject_10	42,00	40,67	36,67	38,00	37,33	161
subject_11	35,00	30,83	33,33	35,83	34,17	144
subject_12	35,83	40,83	42,50	40,00	27,50	140
subject_13	28,75	31,25	31,25	27,50	35,00	101
subject_14	0,00	0,00	0,00	0,00	0,00	27
subject_15	**45,00**	43,00	41,00	43,00	**42,00**	115
subject_16	35,00	35,83	31,67	30,00	35,83	140
subject_17	38,89	31,11	32,22	36,67	32,22	95
subject_18	40,00	37,00	37,00	33,00	34,00	111
subject_19	36,36	30,30	30,30	36,36	36,36	46
subject_20	30,30	**45,45**	**45,45**	**45,45**	30,30	38
subject_21	35,56	41,11	26,67	30,00	30,00	95
subject_22	0,00	0,00	0,00	0,00	0,00	31
subject_23	42,00	40,67	42,00	39,33	39,33	168
subject_24	0,00	0,00	0,00	0,00	0,00	14
subject_25	36,15	37,69	39,23	37,69	36,15	138
subject_26	**42,50**	42,50	44,17	**41,67**	37,50	137
subject_27	38,00	**44,67**	**45,33**	40,00	38,67	165
subject_28	35,33	36,00	32,00	33,33	35,33	163
subject_29	33,64	37,27	30,00	25,45	**40,00**	118
subject_30	**61,11**	27,78	**52,78**	36,11	50,00	48
subject_31	38,00	32,00	40,00	30,00	52,00	70
subject_32	41,67	29,17	36,11	40,28	48,61	48
subject_33	38,64	43,18	38,64	31,82	40,91	58
subject_34	42,00	32,00	50,00	42,00	52,00	64
subject_35	54,00	22,00	52,00	38,00	52,00	70
subject_36	47,50	49,00	52,00	41,50	45,00	216
subject_37	56,00	51,50	51,50	42,00	**57,00**	204
subject_38	41,00	43,50	49,50	40,50	48,00	208
subject_39	49,41	51,18	46,47	35,29	49,41	182
subject_40	50,00	**55,26**	52,63	**49,47**	56,32	200
subject_41	45,88	48,24	42,35	30,00	41,76	190

Table B.3: Part 02 of the table of average classification accuracies with parameters Time-series representation, multi-class classification of all classifiers for the intra-subject test.

ID	CNN	LSTM	MLP	SC	Transformer	# samples
subject_01	48,33	49,44	52,78	51,67	51,11	184
subject_02	39,39	42,42	34,85	40,91	46,97	68
subject_03	43,64	51,82	47,27	47,27	44,55	110
subject_04	45,00	51,25	46,25	42,50	47,50	80
subject_05	53,25	45,45	50,65	42,86	46,75	78
subject_06	45,00	51,25	50,00	51,25	56,25	80
subject_07	37,50	47,50	40,00	42,50	45,00	80
subject_08	39,39	48,48	54,55	43,94	50,00	68
subject_09	52,78	55,56	61,11	**58,33**	**66,67**	36
subject_10	54,29	59,05	**62,86**	56,19	57,62	214
subject_11	50,00	53,16	47,89	47,37	47,89	192
subject_12	**57,78**	57,22	61,67	56,67	59,44	188
subject_13	56,15	**60,77**	61,54	43,85	43,08	134
subject_14	36,36	51,52	57,58	51,52	42,42	35
subject_15	57,33	58,00	61,33	49,33	53,33	154
subject_16	47,78	55,00	52,22	45,00	55,00	186
subject_17	38,33	47,50	46,67	39,17	42,50	126
subject_18	47,86	48,57	52,14	46,43	52,86	148
subject_19	53,33	**68,33**	**61,67**	50,00	48,33	62
subject_20	**58,00**	52,00	58,00	50,00	52,00	50
subject_21	50,83	56,67	55,00	50,83	49,17	126
subject_22	50,00	60,00	57,50	50,00	40,00	40
subject_23	50,00	61,82	57,73	**53,64**	**60,91**	224
subject_24	0,00	0,00	0,00	0,00	0,00	18
subject_25	48,33	52,22	51,11	51,67	47,22	184
subject_26	56,67	58,33	60,56	**56,67**	50,00	182
subject_27	**59,09**	**62,73**	**61,82**	54,55	**57,27**	220
subject_28	47,62	52,86	56,19	48,57	46,19	216
subject_29	52,67	47,33	55,33	50,67	56,67	156
subject_30	52,08	52,08	47,92	**62,50**	54,17	48
subject_31	60,00	**62,86**	61,43	48,57	**57,14**	70
subject_32	43,75	31,25	52,08	41,67	50,00	48
subject_33	54,55	56,36	52,73	54,55	49,09	58
subject_34	58,33	53,33	58,33	61,67	56,67	64
subject_35	57,14	54,29	41,43	48,57	44,29	70
subject_36	50,48	45,71	42,38	50,95	52,38	216
subject_37	48,00	56,50	54,50	50,00	47,00	204
subject_38	57,50	51,00	51,00	53,50	55,00	208
subject_39	**61,67**	57,78	**62,22**	61,11	54,44	182
subject_40	48,00	47,00	51,00	53,50	51,50	200
subject_41	52,63	52,11	49,47	49,47	50,53	190

Table B.4: Table of average classification accuracies with parameters Frequency representation, binary classification of all classifiers for intra-subject test.

ID	CNN	LSTM	MLP	SC	Transformer	# samlpes
subject_01	38,46	41,54	37,69	39,23	34,62	139
subject_02	36,00	44,00	46,00	34,00	32,00	51
subject_03	**62,50**	63,75	50,00	57,50	**52,50**	82
subject_04	41,67	53,33	55,00	51,67	31,67	60
subject_05	52,73	56,36	52,73	58,18	36,36	59
subject_06	60,00	61,67	60,00	**66,67**	26,67	60
subject_07	48,33	48,33	45,00	51,67	48,33	60
subject_08	24,00	44,00	36,00	48,00	48,00	51
subject_09	0,00	0,00	0,00	0,00	0,00	28
subject_10	30,63	43,75	49,38	43,75	30,00	161
subject_11	31,43	30,71	23,57	26,43	29,29	144
subject_12	52,86	47,14	42,86	45,71	33,57	140
subject_13	62,00	**64,00**	**63,00**	65,00	51,00	101
subject_14	0,00	0,00	0,00	0,00	0,00	27
subject_15	40,91	52,73	47,27	52,73	38,18	115
subject_16	**66,43**	**70,71**	**75,71**	**74,29**	**54,29**	140
subject_17	37,78	47,78	44,44	31,11	28,89	95
subject_18	40,91	41,82	46,36	53,64	29,09	111
subject_19	45,45	63,64	52,27	56,82	43,18	46
subject_20	52,78	52,78	44,44	55,56	50,00	38
subject_21	37,78	43,33	41,11	51,11	33,33	95
subject_22	43,33	53,33	53,33	63,33	36,67	31
subject_23	58,13	56,25	62,50	68,13	45,00	168
subject_24	0,00	0,00	0,00	0,00	0,00	14
subject_25	33,85	40,77	47,69	48,46	**51,54**	138
subject_26	40,00	43,85	48,46	46,92	36,15	137
subject_27	**47,50**	**50,00**	**57,50**	**56,25**	38,75	165
subject_28	46,88	38,13	43,13	46,25	33,13	163
subject_29	37,27	45,45	46,36	40,00	37,27	118
subject_30	43,75	35,42	47,92	43,75	43,75	48
subject_31	37,14	40,00	32,86	45,71	40,00	70
subject_32	29,17	16,67	16,67	25,00	41,67	48
subject_33	**52,73**	**45,45**	30,91	45,45	45,45	58
subject_34	46,67	35,00	**51,67**	**48,33**	50,00	64
subject_35	51,43	42,86	42,86	45,71	42,86	70
subject_36	37,62	35,24	35,71	38,57	49,52	216
subject_37	42,50	45,50	45,00	31,50	47,00	204
subject_38	44,50	39,00	46,00	44,00	**50,50**	208
subject_39	47,78	45,00	45,00	47,22	49,44	182
subject_40	38,50	36,50	37,50	35,00	46,50	200
subject_41	43,68	40,00	43,16	34,74	44,74	190

Table B.5: Part 01 of the table of average classification accuracies with parameters Frequency representation, multi-class classification of all classifiers for intra-subject test.

ID	CNN sorted	LSTM sorted	MLP sorted	SC sorted	Transformer sorted	# samlpes
subject_01	45,00	47,50	43,33	43,33	43,33	139
subject_02	33,33	50,00	43,33	33,33	36,67	51
subject_03	60,00	52,86	50,00	60,00	58,57	82
subject_04	53,33	55,00	51,67	48,33	45,00	60
subject_05	44,00	42,00	52,00	54,00	56,00	59
subject_06	53,33	65,00	56,67	66,67	40,00	60
subject_07	35,00	55,00	53,33	51,67	40,00	60
subject_08	42,50	27,50	42,50	52,50	45,00	51
subject_09	0,00	0,00	0,00	0,00	0,00	28
subject_10	50,00	44,67	45,33	50,00	46,67	161
subject_11	31,67	31,67	34,17	25,83	30,00	144
subject_12	41,67	50,83	45,00	48,33	43,33	140
subject_13	**65,00**	**71,25**	**67,50**	**67,50**	**71,25**	101
subject_14	0,00	0,00	0,00	0,00	0,00	27
subject_15	51,00	53,00	55,00	51,00	56,00	115
subject_16	**66,67**	**70,00**	**67,50**	**73,33**	**63,33**	140
subject_17	41,11	45,56	45,56	38,89	36,67	95
subject_18	56,00	54,00	51,00	57,00	38,00	111
subject_19	57,58	57,58	60,61	66,67	57,58	46
subject_20	51,52	45,45	45,45	66,67	48,48	38
subject_21	41,11	35,56	43,33	53,33	36,67	95
subject_22	0,00	0,00	0,00	0,00	0,00	31
subject_23	60,67	61,33	62,00	69,33	48,67	168
subject_24	0,00	0,00	0,00	0,00	0,00	14
subject_25	33,85	44,62	50,77	42,31	40,77	138
subject_26	44,17	54,17	48,33	50,00	43,33	137
subject_27	**58,67**	**53,33**	**54,00**	**58,00**	**46,00**	165
subject_28	40,00	36,00	48,67	46,67	42,00	163
subject_29	45,45	43,64	51,82	40,91	40,91	118
subject_30	52,78	41,67	33,33	50,00	58,33	48
subject_31	46,00	**50,00**	30,00	42,00	**60,00**	70
subject_32	32,64	25,00	27,78	17,36	34,72	48
subject_33	36,36	43,18	34,09	43,18	45,45	58
subject_34	48,00	42,00	50,00	**52,00**	54,00	64
subject_35	**56,00**	42,00	**52,00**	46,00	54,00	70
subject_36	35,00	31,50	34,00	34,00	43,50	216
subject_37	44,50	47,00	39,00	30,00	42,00	204
subject_38	47,00	40,50	47,00	42,00	44,50	208
subject_39	50,00	45,29	42,94	48,24	50,59	182
subject_40	35,79	35,79	38,42	36,32	46,84	200
subject_41	45,88	37,65	42,35	30,59	47,06	190

Table B.6: Part 02 of the table of average classification accuracies with parameters Frequency representation, binary classification of all classifiers for intra-subject test.