Zurich University
of Applied Sciences

School of
Engineering
InIT Institute of Applied
Information Technology

Multi-Task Learning for Deep Neural Network Representations of Human Brain Activity

Bachelor's Thesis

Benjamin Bertalan
Gian A. Hess

InIT
School of Engineering
ZHAW University of Applied Sciences

Supervisors:
Prof. Dr. Thilo Stadelmann
Dr. Ricardo Chavarriaga

June 11, 2021

Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

DECLARATION OF ORIGINALITY
Bachelor's Thesis at the School of Engineering

By submitting this Bachelor's thesis, the undersigned student confirms that this thesis is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members are not considered as third party).

The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the Bachelor thesis have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

City, Date:	**Name Student:**
Winterthur, 11.06.2021	Benjamin Bertalan
Winterthur, 11.06.2021	Gian Andri Hess

Zusammenfassung

Das Gebiet der Gehirn-Computer-Schnittstellen befasst sich mit der Aufgabe, Geräte direkt durch die neuronale Aktivität im Gehirn steuern zu können. Die Dekodierung von Elektroenzephalographie-Signalen (EEG) wird durch mehrere Probleme erschwert, von denen einige eine langwierige Kalibrierung von BCIs vor einer Anwendung erfordern. Kürzlich wurden Convolutional Neural Networks (CNN) mit Erfolg in diesem Bereich eingesetzt. Im Feld der Computer Vision haben Multitask-Learning-Modelle bewiesen, dass sie auf kleinen Datensätzen sehr gute Performanz erzielen können. Wir implementieren einen Deep-Multi-Task-Learning-Ansatz mit Convolutional Neural Network, bei dem einzelne Probanden als verschiedene Aufgaben (Tasks) betrachtet werden. Wir trainieren auf einem Mehrklassen-MI-EEG-Datensatz und vergleichen die resultierende Performanz mit Single-Task-Modellen. Des Weiteren untersuchen wir die Möglichkeit, das gelernte Grundgerüst zum Training von Klassifikatoren für neue Probanden zu verwenden, da dies das Potenzial hätte, die Kalibrierungszeit in Zukunft zu reduzieren. Schließlich verwenden wir zwei verschiedene Visualisierungsmethoden, um Einblicke in die gelernten Modelle, zum Zwecke der Interpretierbarkeit, zu erhalten. Unsere Ergebnisse zeigen, dass MTL in der Lage ist, die Leistung von CNNs bei der Dekodierung von MI-EEG-Signalen zu verbessern, aber ob die Leistung tatsächlich besser wird, hängt von der zugrunde liegenden Architektur ab. Unsere Visualisierungen zeigen, dass ein MTL-Modell andere Eigenschaften lernt als ihre Single-Task-Pendants.

Abstract

The field of brain-computer interfacing addresses the task of enabling devices to be controlled directly by neural activity in the brain. The decoding of electroencephalography signals (EEG) is complicated by several issues, some of which necessitate tedious calibration of BCIs prior to usage. Recently, convolutional neural networks have been applied with success to this domain. In computer vision, multi-task learning models have shown to perform well on small data sets. We implement a deep multi-tasklearning approach with convolutional neural networks, where different subjects are considered as different tasks. We train on a multi class MI-EEG data set and compare the resulting performance with single-task models. Further, we examine the possibility of using the learned backbone to train classifiers for new subjects, as this would have the potential of reducing calibration time in the future. Finally, we use two different visualization methods to gain insights into the learned models for the purpose of interpretability. Our results show that MTL is able to improve the performance of CNNs on decoding MI-EEG signals, but whether performance does improve is dependent on the underlying architecture. Our visualisations show, that a MTL model selects different features as their single-task counterparts.

Acknowledgements

We would like to express our deep gratitude to Prof. Dr. Thilo Stadelmann and Dr. Ricardo Chavarriaga, our supervisors, for their patient guidance, enthusiastic encouragement, and useful critiques of this thesis.

Contents

Zusammenfassung ii

Abstract iii

Acknowledgements iv

1 Introduction 1

1.1 Motivation . . . 1

1.2 Objectives . . . 2

2 Theoretical background 3

2.1 Brain-computer interfaces (BCI) . . . 3

2.2 Electroencephalography (EEG) . . . 4

2.3 Motor imagery (MI) . . . 5

2.4 Artificial neural networks (ANNs) . . . 6

2.4.1 Convolutional neural networks (ConvNets/CNNs) . . . 7

2.5 Transfer learning . . . 8

2.6 Multi-task learning . . . 9

3 Related work 11

3.1 Deep convolutional neural networks for motor imagery classification 11

3.1.1 ShallowFBCSPNet . . . 12

3.1.2 Deep4Net . . . 12

3.1.3 EEGNet . . . 12

3.2 Transferability of features in deep neural networks . . . 13

3.3 Transfer learning for EEG decoding . . . 13

3.4 Multi-task learning for EEG decoding . . . 14

4 Methods **15**

4.1 Multi-task network architecture 15

4.1.1 Forwarding to all heads 16

4.1.2 Splitting the models 16

4.2 Pretraining 16

4.3 Task weighting 17

4.4 Transfer learning 18

4.5 Visualizations 19

4.5.1 Input-perturbation network-prediction correlation maps . 19

4.5.2 Most-activating input windows 19

4.6 Data set 19

4.7 Preprocessing 20

4.8 Hyperparameter optimization 20

4.9 Baselines 20

4.10 Experiments 21

4.11 Performance evaluation & statistics 22

5 Results **23**

5.1 Validation of multi-task learning implementation 23

5.2 Result 1: Deep4Net 24

5.2.1 Layer split sensitivity 24

5.2.2 Comparison of multi-task architecture to single-task architecture performance 24

5.2.3 Forwarding all subjects to all heads 24

5.2.4 Transfer learning performance 24

5.2.5 Pretrained models 25

5.3 Result 2: EEGNet 27

5.3.1 Layer split sensitivity 27

5.3.2 Comparison of multi-task architecture to single-task architecture performance 27

5.3.3 Forwarding all subjects to all heads 27

5.3.4 Transfer learning performance 27

5.3.5 Pretrained models 28

5.4 Result 3: ShallowFBCSPNet 30
5.4.1 Layer split sensitivity 30
5.4.2 Comparison of multi-task architecture to single-task architecture performance 30
5.4.3 Forwarding all subjects to all heads 30
5.4.4 Transfer learning performance 31
5.4.5 Pretrained models 31
5.5 Result 4: Task weighting 33
5.6 Result 5: Confusion matrices 34
5.7 Result 6: Input-perturbation network-prediction correlation maps 37
5.8 Result 7: Most-activating input windows 40

6 Discussion and outlook 42
6.1 Conclusions 42
6.2 Limitations 44
6.3 Future work 44

Bibliography 45

List of Figures 51

List of Tables 53

A Project Description A-1

B Contact information and code access B-1

C Figures C-1
C.1 Architectures C-1

Chapter 1

Introduction

1.1 Motivation

Brain-computer interfaces allow a computer to be controlled without activating the nervous system (i.e. without any involvement of the motor periphery or speech) through direct recording and decoding of brain activity. They can be used in several areas of application, one of which is to enable individuals who are prevented by disability from controlling a device through conventional methods. Examples of this are controlling a wheelchair or prosthetic limbs [1, 2].
Electroencephalography (EEG) is often used as a method of recording neuronal activity, as it is non-invasive and thus eliminates the risk of surgery for the user. However, decoding EEG signals is challenging for several reasons, including low signal-to-noise ratio, high inter-subject and inter-session variability and low spatial resolution. Another issue is the high cost of collecting labelled data in this domain. The resulting data sets are significantly smaller than data sets used in other areas where neural networks are applied, for example computer vision or natural language processing. This creates a bottleneck, as neural networks require large amounts of data to reach an acceptable performance level. An additional issue is the need for expert knowledge to extract meaningful features [3]. To address these challenges, various methods from the field of deep learning are used, which have proven to be extremely successful in other areas such as computer vision [4]. The application of neural networks in recent years has enabled the decoding of these signals in an end-to-end manner, removing the requirement to craft features manually [5]. It has been shown that convolutional neural networks can achieve a similar performance level in decoding EEG signals compared to earlier approaches [6]. However, the problems associated with high inter-subject and inter-session variability persist, necessitating tedious calibration for a given subject or a new session prior to usage to render brain-computer interfaces usable. A promising method to shorten calibration time is transfer learning. In transfer learning, previously learned knowledge is used to learn a target task. Although research on transfer learning for BCI is relatively new, several transfer learning methods have already been applied to BCIs [7, 8].
Another method that promises better generalization by using domain-specific

knowledge is multi-task learning (MTL). MTL has already been applied with success in areas such as computer vision. Typically, MTL involves training different related tasks from a certain domain together. An example from computer vision would be to simultaneously train segmentation, depth estimation, and object recognition on an image [9]. Due to the high inter-subject variability when decoding EEG signals, we believe it is reasonable to consider different subjects as different tasks. MTL has also shown increased performance in scenarios where the data sets are small.[10] We think that the scanty data sets in BCI could benefit from this potential. We will therefore try to apply MTL in this manner, with the expectation that the sharing of knowledge between different subjects will lead to improved generalization.
Furthermore, because the interpretation of deep neural networks is intrinsically difficult, we want to use visualization methods to achieve better interpretability of our models.

1.2 Objectives

The aim of this thesis is to explore multi-task learning in a deep learning approach suitable for the characteristics of brain-machine interfacing, in order to overcome current issues in decoding of EEG data. We start by briefly analyzing the current state of EEG decoding, transfer learning, and how it can be extended to a MTL architecture. We discuss publications on models used in EEG decoding, transferability of features in deep neural networks, and how MTL can be used in EEG decoding. We want to evaluate the performance of three deep learning models Deep4Net, EEGNet and ShallowFBCSPNet, when a multi-task learning architecture, where each subject in the data set is considered a task. Additionally, we want to evaluate ways to further improve the performance of the MTL models.

While state of the art EEG decoding research reaches new accuracy highscores in EEG classification, we are not focusing on outperforming these with our work, but instead propose a new architecture and analyze its potential for EEG decoding. Our aim is to create an approach that is helpful for future research in this field.

Chapter 2

Theoretical background

This chapter serves to briefly present the theoretical foundations on which this thesis is based.

2.1 Brain-computer interfaces (BCI)

Brain-computer interfacing is an interdisciplinary field that attempts to use achievements from neuroscience, signal processing, artificial intelligence, and information theory to control devices by directly using brain activity. The origin of BCIs can be traced to works by Delgado and Fetz in the 1960s [11, 12]. The more recent increase in interest in BCI can be attributed to both more available computing capacity and better signal processing and machine learning algorithms.
BCIs make use of electrical potentials produced by neurons. Recording of these potentials can be done either by invasive or non-invasive methods [13].

Brain-computer interfaces rely on signal processing and machine learning algorithms. Processing of the signal is typically necessary because not all signals are caused by or correlate with the relevant mental processes, due to the presence of muscle or motion artifacts and electromagnetic noise in the recorded signal. The components which are yielded from the preprocessing steps are called *features*. This term is commonly used in machine learning literature to designate the inputs to the respective algorithms.

In general, decoding of (preprocessed) signals can be viewed as applying a mathematical function,

$$\boldsymbol{y} = f(\boldsymbol{X}^r, \theta) \tag{2.1}$$

with $\boldsymbol{X}^r \in \mathbb{R}^{N \times T}$ being the measured brain activity (with N channels and T samples), and θ the given parameters. The output $\boldsymbol{y}$, the inferred command, is then used to control a device. The steps that together constitute a closed loop are illustrated in Figure 2.1. The general principles are described more thoroughly

in Iturrate et al. [14].

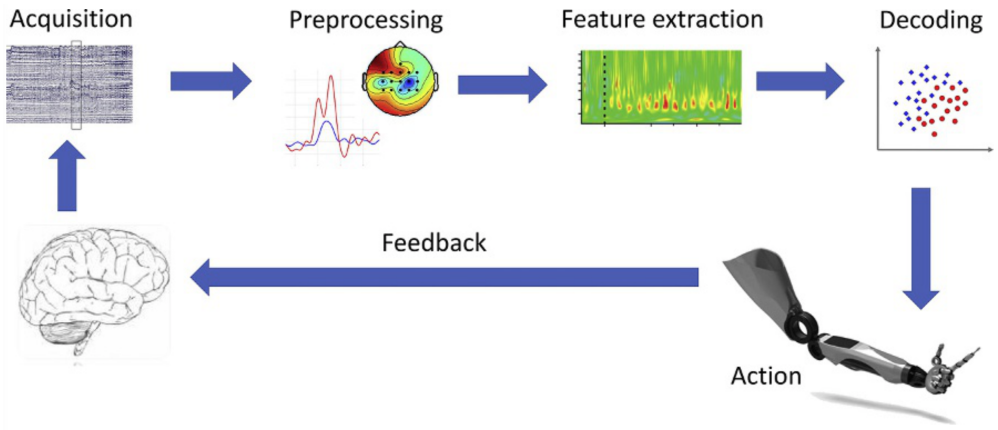

Figure 2.1: **Processing steps of a brain-computer interface**: The acquired signal is preprocessed and relevant features are selected. A decoder receives the features as an input and infers an action. The user receives feedback either explicitly or via their own senses [14].

2.2 Electroencephalography (EEG)

Electroencephalography (EEG) is a popular non-invasive technique for recording neural activity that involves placing sensors on the scalp. The sensors measure the summed postsynaptic potentials of many thousands of neurons [15]. Currents tangential to the scalp are not detected. EEG recordings have a poor spatial resolution, mainly due to the layers of tissue that lie between the source of the signals and the sensors. Furthermore, since voltage fields fall with the square of the distance, signals originating deep in the brain are not detected. The temporal resolution, on the other hand, is good (in the millisecond range).
The amplitude of the recorded signals is in the range of a few tens of microvolts. As a result, the signals can be easily contaminated by electrical lines, but also by movements (blinking, talking, movement of the eyebrows, etc.). Subjects are therefore usually instructed to avoid any movement, and powerful artifact removal algorithms are used to exclude or filter out parts of the EEG signal that are corrupted by muscle artifacts. Additional sources of noise include changing electrode impedances and varying psychological states of the user due to boredom, distraction, stress, or frustration [13].

For consistent placement and naming of sensors across laboratories, there is an international standard, the 10-20 system. Reference electrodes are specified, and

from these points the perimeters are divided into 10 and 20 percent intervals, as shown in Figure 2.2.

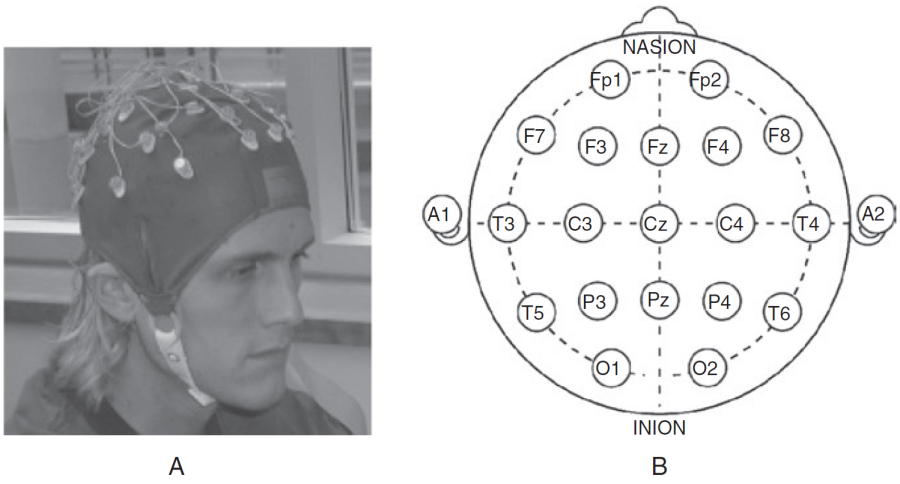

Figure 2.2: **(A) Subject wearing a 32-electrode EEG cap. (B) International 10–20 system for standardized EEG electrode locations on the head** [13].

The waves captured by EEG recordings are typically categorized into different frequency ranges that are associated with different mental states. For example, alpha waves (or the *alpha rhythm*) occur in awake subjects in the absence of movement and beta waves are recorded when a person is alert and concentrating. Theta waves are associated with drowsiness or idleness in children and adults, while delta waves can be observed in infants in general and in adults at certain stages during sleep. One particular type of alpha wave popular in applications of BCIs is the mu rhythm. It is found over the sensorimotor areas and is influenced, namely attenuated or suppressed, by movement or by imagining movement [13]. The EEG rhythms and their frequency ranges can be seen in Figure 2.3.

2.3 Motor imagery (MI)

Motor imagery is a frequently used paradigm in BCI. Motor imagery is described as mentally rehearsing a motor action without performing movement [16]. Conscious motor imagination and unconscious preparation for a movement share common mechanisms and are functionally equivalent. Therefore, very similar brain signals occur during motor imagery as during actually performed movement [17]. The movement of different body parts is associated with different brain areas. For example, when imagining movements in the hands, signals occur in the regions over which electrodes C3 and C4 are placed [18].

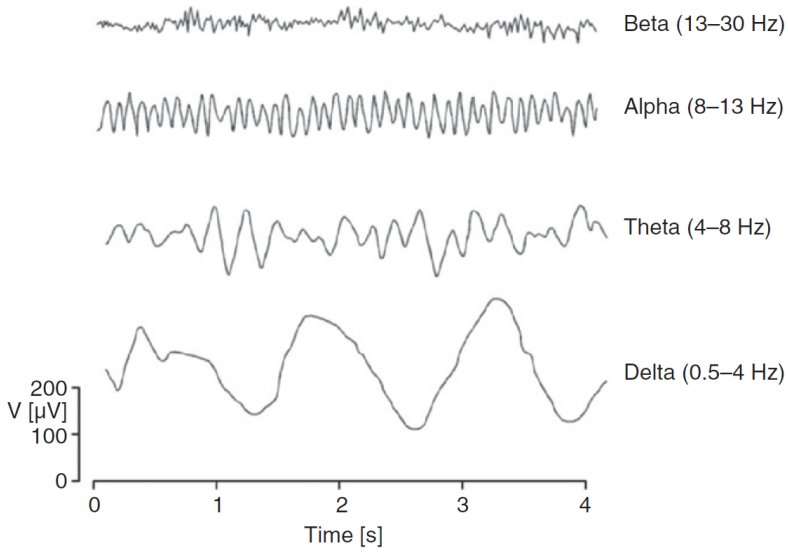

Figure 2.3: **Examples of EEG rhythms and their frequency range** [13].

2.4 Artificial neural networks (ANNs)

Artificial neural networks are loosely inspired by the biological brain. They consist of artificial neurons that mimic biological neurons (albeit much less complex). Artificial neurons are a generalization of the idea of a perceptron, and artificial neural networks are therefore also called *multilayer perceptrons*. The perceptron was first introduced by Frank Rosenblatt in 1958 [19]. An illustration of a biological neuron and an artificial neuron is shown in Figure 2.4. An artificial neuron receives an input $\boldsymbol{x}$, which is multiplied by the weights $\boldsymbol{w}$. The products $x_i w_i$ are summed and a bias b is added before they are passed to a non-linear activation function f, whose output is also the output of the neuron. The output of a neuron can therefore be written as $f(\sum_{i=0}^{n} x_i w_i + b)$. As an activation function, the *rectified linear unit* activation function (ReLU) is recommended for most feed-forward neural networks. This is because, although it is non-linear, its closeness to linearity facilitates the optimization of the model by gradient-based methods [20]. ReLU is defined by the activation function $f(x) = max(0, x)$ depicted in Figure 2.5.

By arranging the neurons in layers, and creating connections between neurons of two adjacent layers (but none in the layer itself), a neural network is obtained (Figure 2.6). The outputs of the neurons of the last layer together form the output $\hat{y}$ of the network. To train the network, input samples are fed through the network, and the distance of the output $\hat{y}$ to the true value y (label of the input) is calculated (by default the Euclidean distance or *L2 norm*, but many more exist).

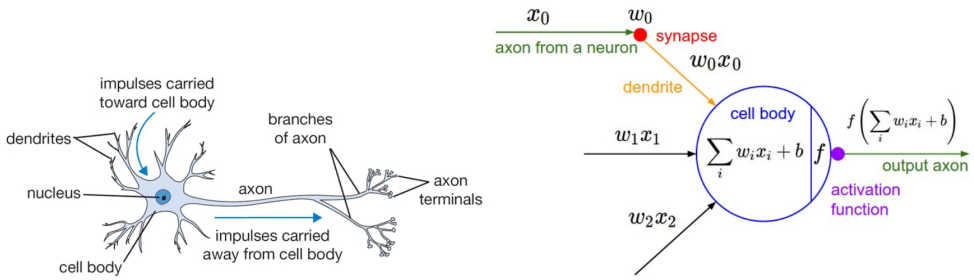

Figure 2.4: **A representation of a biological neuron (left) and its mathematical model (right)** [21].

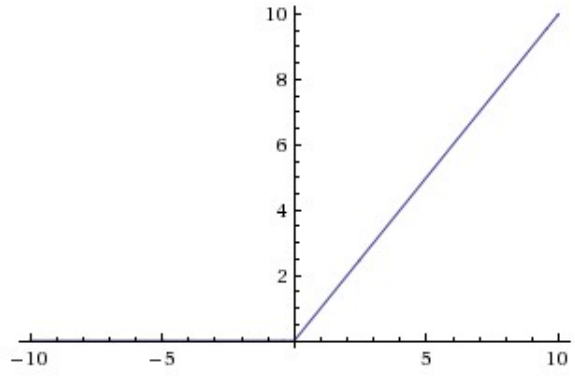

Figure 2.5: **Rectified Linear Unit (ReLU) activation function**, which is zero when $x < 0$ and then linear with slope 1 when $x > 0$ [21].

This distance is called *loss*, and is used to optimize the network. For this purpose, the gradients are calculated (in the case of gradient-based optimization) and back-propagated through the network and the weights are adjusted accordingly. This process can be considered as approximating a multivariable function that maps the input to the output [22].

2.4.1 Convolutional neural networks (ConvNets/CNNs)

Convolutional neural networks are inspired by the visual cortex of animals [24]. CNNs are very similar to common neural networks as they consist of neurons that have learnable weights and biases. However, as the name indicates, they additionally use *convolutions* as an operation. Discrete convolution can be viewed as multiplication by a matrix. In the context of CNNs, a kernel (a matrix of weights) is slid over the input and a weighted sum is computed at each position. The use of convolution is motivated by the fact that, for example, in images or

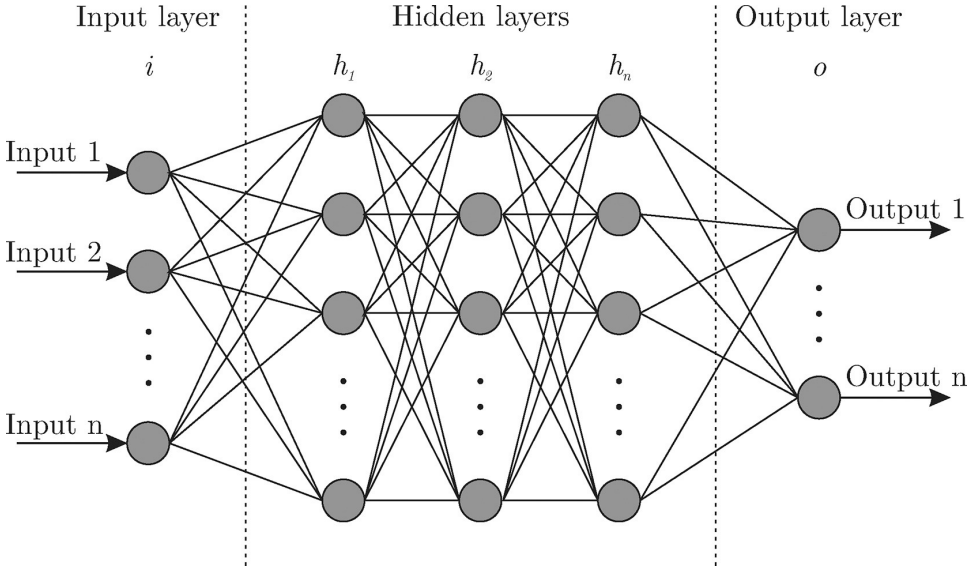

Figure 2.6: **Artificial Neural Network** [23]

time series, inputs that are far apart (spatially or temporally) are presumably not dependent on each other. By using a kernel that is smaller than the input, an output unit no longer interacts with every input unit, which is termed *sparse connectivity*. In a deep convolutional network, units in the deeper layers may interact indirectly with a larger portion of the input. The number of neurons with which a unit interacts indirectly defines the *receptive field* of the unit. A typical layer of a CNN consists of three stages. First the convolutions are applied, then the non-linearity, and after that the output is often modified by a so called *pooling* function. The pooling function takes the summary statistics of nearby outputs at a given location and replaces the output of the network with it. Without pooling, small translations in the input could result in a large variance in the output [20].

2.5 Transfer learning

In cases where large data sets are not available due to expensive acquisition or expensive labeling, transfer learning can be useful in overcoming the problem of insufficient amounts of training data. Transfer learning relaxes the hypothesis that the training data is independent and identically distributed (i.i.d.) with the test data. Transfer learning aims to facilitate the training of a target task $\mathcal{T}_t$ from a target domain $\mathcal{D}_t$ by using a source task $\mathcal{T}_s$ from a source domain $\mathcal{D}_s$ where $\mathcal{D}_t \neq \mathcal{D}_s$ or $\mathcal{T}_t \neq \mathcal{T}_s$. Usually there is more data available from the source domain, or training the source tasks is easier [25]. An illustration comparing the traditional learning process to transfer learning is shown in Figure 2.7. Pan and Yang note in their survey on transfer learning that the transfer learning research

community is concerned with three main questions: (1) what to transfer, (2) how to transfer, and (3) when to transfer. The first two questions are concerned with what knowledge between the domains or tasks is shared and can be transferred, and the development of algorithms that are able to transfer this knowledge. The third question arises because *negative transfer* is possible, i.e. the transferring of knowledge inhibits the performance of the target task. They note that the important issue of how to avoid negative transfer is attracting more and more attention [26].

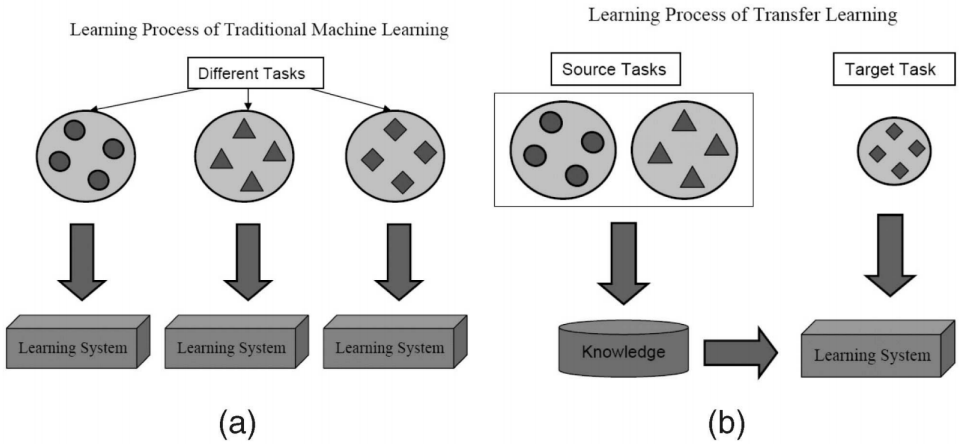

Figure 2.7: **Different learning processes between (a) traditional machine learning and (b) transfer learning** [26].

2.6 Multi-task learning

Multi-task learning (MTL) refers to a strategy in which several related tasks are trained simultaneously while using a shared representation. The primary goal is to increase generalization performance, i.e. performance on samples which were not used to train the model [27]. By training multiple tasks in parallel, a model can be enabled to leverage domain-specific knowledge from the training signals of other tasks. In this context, other tasks serve as inductive bias, that is, a set of assumptions of a learning algorithm that leads to a preference of certain hypotheses over other hypotheses [28, 29]. An illustration of a multi-task neural network is shown in Figure 2.8. Goodfellow et al. describe MTL as a type of regularization, where sharing the lower layers can be seen as imposing a soft constraint on the parameters of the model [20].

MTL is similar to transfer learning, in the sense that both methods aim to transfer knowledge. The difference is the task on which more focus is laid. With transfer learning, one tries to learn a target task better with the help of a source

task. In MTL, one aims to improve all tasks by sharing knowledge among all tasks. Transfer learning therefore puts more focus on the target task, while in MTL all tasks are considered equally important [30].

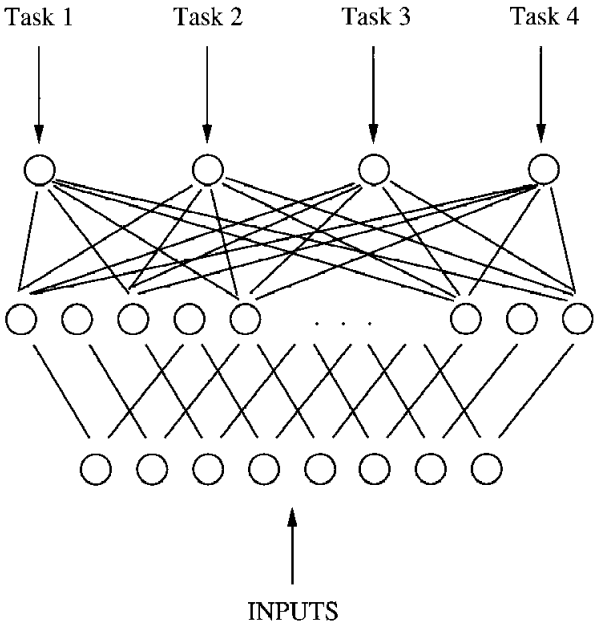

Figure 2.8: **Illustration of a multi-task neural network** [28]. Unlike a single-task model, a multi-task model has outputs that are associated with multiple tasks. In this illustration, only one output per task is shown, but there can be (and usually will be in practice) multiple outputs.

Chapter 3

Related work

In this chapter, we briefly summarize literature that is related to our work or on which we have based parts of our work.

3.1 Deep convolutional neural networks for motor imagery classification

Schirrmeister et al. showed that deep convolutional neural networks, together with recent advances from the field of machine learning, such as batch normalization, exponential linear units and dropout combined with a cropped training strategy can achieve at least as good a performance as the widely used filter bank common spatial patterns (FBCSP) algorithm. To accomplish this, they had to address some difficulties in applying CNNs to EEG. CNNs were initially developed for image recognition. Yet, EEG signals have characteristics that differ from those of images. In addition to the aforementioned problems associated with EEG signals, unlike images, they are not static, but dynamic time series. Learning features from EEG data with CNNs is therefore more difficult than learning features from images. For this reason, Schirrmeister et al. adapted CNNs from computer vision for EEG input. They analyzed the influence of design choices, for example the overall architecture of the networks, as well as training strategies. Regarding design choices, the first important decision for them was how to represent the input. They argue that EEG signals do not show any obvious hierarchical compositionality in space, but that it is known that EEG signals are organized over multiple time scales, i.e., appear as nested oscillations. For this reason, they decided to represent the input as a 2D array with the number of time steps as the width and the number of electrodes as the height. Further concerning the design decisions, they evaluated architectures with varying number of layers. The two models that could achieve similar or better performance than FBCSP are the ShallowFBCSPNet (2 layers) and the Deep4Net (5 layers). We describe the models further below. Regarding the training strategy, cropped training, i.e. using sliding input windows in order to increase the amount of training samples, proved to be beneficial, especially for the deep model. They attribute this to the

fact that a large number of examples are needed to learn the relevant features, and that the model does not overfit to phase information due to the shifted windows.

They do, however, note that besides the advantages CNNs provide, such as end-to-end learning, i.e. learning from raw data without any a priori feature selection and the scalability to large datasets, there are also disadvantages using CNNs compared to other machine learning models. The disadvantages include the requirement of a large amount of training data, the time needed to train CNNs, and that they may output false predictions with high confidence. Additionally, CNNs are difficult to interpret, i.e. it is difficult to determine which characteristics of the data are used by them to perform a prediction. To address this, they proposed a visualization method, which we describe in more detail in chapter *Methods* [6].

3.1.1 ShallowFBCSPNet

The ShallowFBCSPNet is inspired by the FBCSP pipeline. The first two layers perform temporal convolution and spatial filtering. These are followed by a squaring non-linearity, mean pooling and a logarithmic activation function. The steps are analogous to the trial log-variance computation in FBCSP [6]. An illustration of the architecture is shown in appendix C.

3.1.2 Deep4Net

The Deep4Net is inspired by successful models in computer vision. It consists of four convolution-max-pooling blocks, with a special first block designed to handle EEG input and a softmax classification layer. The architecture can be seen in appendix C. The reasons given by Schirrmeister et al. for using a generic architecture are as follows: They wanted to see if an architecture designed with minimal expert knowledge could achieve competitive performance and to support the idea that CNNs can be used as a general-purpose tool to decode brain signals. Additionally, such an architecture can directly benefit from future methodological advances in deep learning [6].

3.1.3 EEGNet

A further model for EEG decoding is the EEGNet proposed by Lawhern et al. In contrast to Deep4Net, EEGNet does not have a generic architecture, but is specifically designed for decoding EEG signals. Unlike the Deep4Net, the EEGNet uses depth-wise and separable convolutions after the temporal convolution. The authors argue that in addition to reducing the number of learnable parame-

ters, this enables spatial filters to be learned directly for each temporal filter (for depthwise convolution) and achieves decoupling of relationships within and across feature maps (for separable convolution). The EEGNet has up to two orders of magnitude fewer learnable parameters compared to the ShallowFBCSPNet and Deep4Net. It should be noted that to facilitate implementation, they used 2D convolutions, but in reality 1D convolutions are performed [31]. An illustration of the architecture can also be found in appendix C.

3.2 Transferability of features in deep neural networks

Yosinki et al. investigated the transferability of features in deep neural networks. They make the observation that many deep neural networks trained on images learn similar features in the first layer. These features do not seem to be specific to a dataset or task. They label such features which are applicable to many data sets and tasks and occur regardless of the exact cost function as *general*. Features in the last layers of a network successfully trained towards a supervised classification objective are referred to as *specific*. Moreover, they ask themselves to what extent it is quantifiable to which degree a layer is general or specific, where the transition from general to specific takes place, and which criteria are decisive for the transferability of features [32].

3.3 Transfer learning for EEG decoding

Variations between subjects or sessions are typical in EEG signals. Transfer learning is thus promising for decoding EEG signals because transfer learning does not assume that the train set and the test set are in the same feature space and are subject to the same probability distribution. A review conducted by Zhang et al. that examined 80 studies from 2010 to 2020 on the application of transfer learning to EEG decoding concluded that transfer learning allows adequate initialization of BCIs for new subjects and thus reduces calibration time. In addition, higher accuracies were achieved across sessions and subjects [8]. Uran et al. investigated the application of transfer learning to two MI-EEG data sets. They compared different training strategies, of which *frozen learning* performed best, closely followed by *distributed learning* (pooled learning). Frozen learning involves training a model on all but one subject, then continuing to train the model with the remaining subject while the lower layers of the model are frozen, meaning the parameters of these layers are not adjusted during the training process. In distributed learning, training is performed on the first session of all subjects, and the performance evaluated on the second session of each subject [33].

3.4 Multi-task learning for EEG decoding

Multi-task learning has thus far already been applied to problems in computer vision, natural language processing, and bioinformatics, among other spheres. In their survey on multi-task learning, Zhang and Yang list only one study on MTL for brain-computer interfaces [30]. In this work, Alamgir et al. proposed a multi-task framework for decoding MI-EEG data to learn feature characteristics that are consistent across subjects. They used a linear classifier for this purpose and considered each subject as a task. According to them, their system could achieve satisfactory classification performance for new subjects without calibration. They note that when their classifier was calibrated with subject-specific data, a significant increase in accuracy was achieved compared to a classifier trained on subject-specific data only [34].

Other approaches to MI-EEG decoding that involve multi-task learning have been proposed, however, not regarding each subject as a task, but training additional tasks, e.g. training an autoencoder or metric learning parallel to training on subjects [35, 36].

Chapter 4

Methods

4.1 Multi-task network architecture

Based on the successful decoding of EEG signals with deep convolutional neural networks and application of transfer learning with the latter, and motivated by the results of multi-task learning with a linear classifier, we propose a deep multi-task architecture for EEG decoding. Due to the high inter-subject variability, we want to consider each subject as a task. To the best of our knowledge, this approach has not yet been applied with deep neural networks. We believe that the end-to-end learning capability of CNNs, together with knowledge sharing between tasks with MTL will facilitate selecting features that are more consistent across subjects. When applying MTL to a problem, one has to consider whether the tasks are related in terms of low-level features or high-level concepts. Alamgir et al. note that despite the known inter-subject variability, the principle characteristics of features remain invariant across subjects. In motor imagery, event-related desynchronization (i.e. a decrease in power) of μ- and β-rhythms over the contralateral sensorimotor areas is typical [34]. Therefore, we expect features in the lower layers, where usually temporal and spatial convolutions are performed, to be consistent across subjects. Selected features would then correspond to channels and frequencies associated with the respective mental processes. Because of the hierarchical nature of CNNs, features become increasingly complex from one layer to the next. By combining the low-level features, they would be able to model complex frequency modulations. We believe that these can differ between subjects. The transition from general features, i.e. features that are consistent across subjects, to specific features has to occur somewhere in the network [32]. We consider it reasonable to split the network into one path per subject (we refer to these subject-specific paths as *heads*) before this transition. The reason we consider subject-specific heads to be potentially useful is that conflicting gradients may interfere with the optimization of the network. However, this approach reduces the number of samples that pass through a head. Since it is not known where the aforementioned transition from general to specific features takes place, we have chosen an additional hyperparameter *layer split* and made it part of our hyperparameter estimation. Mormont et al. have already applied such an archi-

tecture, similar to the one we envisioned for our problem, to images in the field of digital pathology [37]. An illustration of this architecture is shown in Figure 4.1. Because the models of Schirrmeister et al. were tailored to decode MI-EEG data and achieved good results, we want to build on these to see if performance can be further improved by MTL.

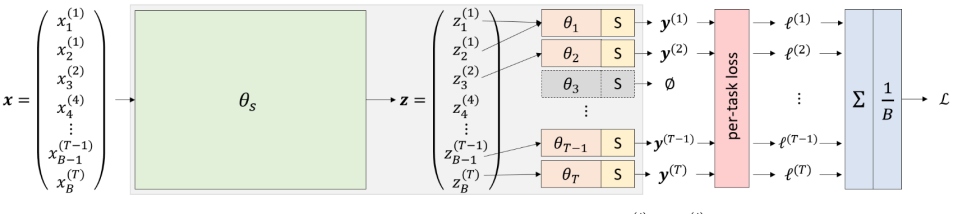

Figure 4.1: **Multi-task architecture proposed by Mormont et al.**: From the input $\boldsymbol{x}$ a shared representation $\boldsymbol{\theta_s}$ is computed. Samples from this representation are forwarded to the respective heads. The loss is calculated per task and aggregated [37].

4.1.1 Forwarding to all heads

Due to the possible disadvantage of a reduced number of samples passing through the upper layers, we will also explore an alternative approach. This approach involves forwarding the samples of all subjects to all heads. A new class *irrelevant* is introduced, as which the sample should be classified, if the sample does not belong to the subject associated with the head.

4.1.2 Splitting the models

To use the MTL architecture with the three models Deep4Net, EEGNet and ShallowFBCSPNet, we have to define split points for each model, where the backbone ends and the heads start. The Deep4Net is built with 4 convolutional pool blocks and a classification layer. We decided to set the splits between each block and the classification layer. For the EEGNet and ShallowFBCSPNet, the splits are between convolution layers and the classification layer. Where to set the split point for the experiments will be determined by a hyperparameter optimization with the *layer split* as a parameter. An overview of the split points can be seen in figure 4.2.

4.2 Pretraining

Mormont et al. have shown that pretraining MTL models has a positive effect on their performance [37]. We also see pretraining as a way to increase the amount

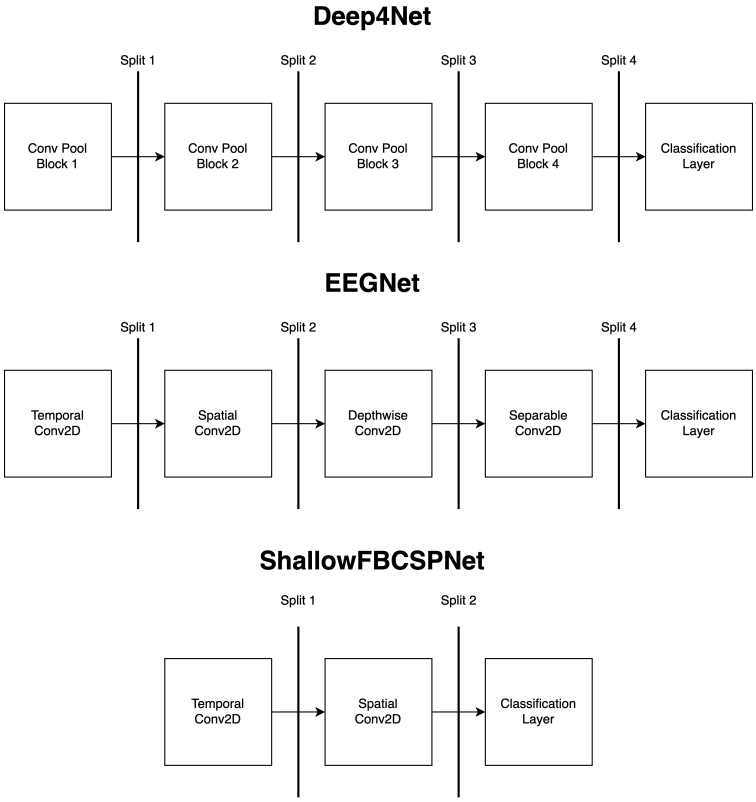

Figure 4.2: **MTL Layer Splits**

of samples passing through a head. We are examining different variants. Firstly, we want to see whether it is helpful if a head is pre-trained with data from all subjects. Secondly, we will look at how performance is affected when one head is trained only with data from the respective subject. Both variants are performed once with and once without a pretrained backbone (i.e. the shared layers).

4.3 Task weighting

Although many deep MTL models have been proposed due to the success of deep learning, the dynamics of MTL are still not well understood, neither at the theoretical level nor at the experimental level. In particular, the usefulness of different task pairs is not known a priori. Gong et al. found in their comparison of loss weighting strategies for multi-task learning in deep neural networks that the appropriate combination of the losses of different tasks is crucial for the success of the model. They also note that MTL usually shows no benefit when using combinations of tasks defined by the user [38]. For this reason, we implemented a widely adopted dynamic task weighting strategy based on task uncertainty

proposed by Kendall et al. [39]. We used a deviation of this loss-weighting strategy by Liebel and Körner, which has the benefit of no longer yielding negative loss values [40]. Using this strategy, the loss of our model is calculated as follows:

$$\mathcal{L}_{\mathcal{T}}(\boldsymbol{x}_{\mathcal{T}}, \boldsymbol{y}_{\mathcal{T}}, \hat{\boldsymbol{y}}_{\mathcal{T}}; \boldsymbol{w}_{\mathcal{T}}) = \sum_{\tau \in \mathcal{T}} \frac{1}{2 \cdot c_\tau^2} \cdot \mathcal{L}_\tau(\boldsymbol{x}_\tau, \boldsymbol{y}_\tau, \hat{\boldsymbol{y}}_\tau; \boldsymbol{w}_\tau) + \ln(1 + c_\tau^2) \tag{4.1}$$

In our case, $\mathcal{L}$ is the negative log-likelihood (NLL) loss. The coefficients c_τ are added to the learnable parameters of the model. It should be noted that in our case the inputs $\boldsymbol{x}_\tau$ are also task-specific in contrast to the inputs in the two publications mentioned above.

4.4 Transfer learning

To assess whether our backbone learns a more general representation compared to single-task or pooled models, we evaluate the performance when transferring knowledge from the previously learned backbone to new subjects. Uran et al [33] proposed the three methods of Distributed Learning, Split Learning and Frozen Learning for transfer learning based on the BCI Competition IV 2a dataset.

Split Learning "The model is trained on data from sessions of all but one user and then retrained on the first session of the last user with the previously initialized parameters. Finally the model is tested on the second session of the same user." - Uran et al. [33]

Frozen Learning "Similar to Split Learning but when the model is re- trained, the lower layers will be frozen, thus reducing the amount of parameters to be trained. The number of layers to be frozen can be manually selected." - Uran et al. [33]

In our Multitask-Learning architecture we can transfer knowledge with distributed and frozen learning. We train the model with all but one subjects and add a new head to the resulting model. The same parameters will be used and trained with the first session of the remaining subject and tested on the second session of the same user. This will further fine-tune the backbone and train a complete new head. For a comparison with the Frozen Learning approach, the backbone will be frozen. In the MTL architecture, a new head has not seen any data yet and thus is not trained. For a better comparison of the features learned in the backbone, we will also run an experiment where the non-frozen layers will be reset in the baseline model. For the results, *TL* stands for transfer learning.

4.5 Visualizations

For the purpose of better interpretability of our models, we have used two different visualization methods, which are described in the following sections.

4.5.1 Input-perturbation network-prediction correlation maps

Schirrmeister et al. proposed a visualization method to help see if a trained model actually focuses on the relevant brain signals. The visualization method in the original paper involves perturbing input frequencies and measuring the correlation with the change in the predictions of the model. We used a slightly modified version from their repository, where the correlation is not measured with the change in the outputs of the model, but with the change in the gradients in the model [6].

4.5.2 Most-activating input windows

Using the most-activating input window method, Hartmann et al. showed for the first time that CNNs not only learn different sinusoids from EEG data but also exhibit complex oscillatory patterns, especially in the last convolutional layer [41]. With this method, inputs are passed through the network and the activations of the neurons are stored after each layer. These activations can then be sorted, and the input windows associated with the highest activations can be analyzed. We want to use this method to show potential differences between subjects, which could be informative regarding which subjects are suitable to train in parallel. Furthermore, we want to identify possible differences in the learned features of MTL models compared to single-task models.

4.6 Data set

We used the 2a Continuous Multi-class Motor Imagery data set from Brunner et al. provided for BCI competition IV. It is therefore also referenced as BCIC IV 2a. This data set consists of 9 subjects with a cue-based BCI paradigm consisting of four different motor imagery tasks. These are imagining movement of the left hand, right hand, both feet and tongue. Each subject has two recorded sessions on different days, consisting of 6 runs each with short breaks in between. One run comprises 12 trials for each class, resulting in 48 trials per run or a total of 288 trials per session [42, 43]. To load the data sets we are using the MOABB library [44].

4.7 Preprocessing

The unmodified preprocessing steps used by Schirrmeister et al. are used. They include a bandpass filter for frequencies below 4Hz and above 38Hz and an exponential moving standardization [6]. We used MNE-Python to preprocess the data [45].

4.8 Hyperparameter optimization

While there are many public benchmarks for these CNNs on the BCIV Competition 2a dataset, the hyperparameters used are not published. To compare the performance of our architecture, we did our own hyperparameter optimization for all three models and used them as our hyperparameters for all experiments.

For the hyperparameter selection, the tool Weights & Biases [46] was used in a bayesian optimization configuration. It uses the principles of Bayes rules 4.2 and a Gaussian process to model the relationship between parameters and the metric that is being minimized or maximized. It chooses parameters to optimize the probability of improvement. The parameters in table 4.2 were optimized. For defining the parameters a grid search was done on each model first to narrow down the range for the values.

$$\mathcal{P}(metric|hyperparameter) = \frac{\mathcal{P}(hyperparameter|metric)\mathcal{P}(metric)}{\mathcal{P}(hyperparameter)} \tag{4.2}$$

Parameter	batch size	drop prob	layer split	learning rate	weight decay
Values	12,32,64	0.1-0.9	1,2,3,4[1]	0.005-0.02	0.0001-0.1

Table 4.2: **Hyperparameter Search Configuration.**

4.9 Baselines

We used the respective single-task counterparts to the models as baselines. For the baselines, we both trained each subject individually (referred to as *single*) and all subjects together (referred to as *pooled*). For methods that aim to improve the performance of our MTL models (such as pretraining or task weighting) we compared the results to the same MTL model trained without these methods.

[1] For ShallowFBCSPNet only 1,2 was configured.

4.10 Experiments

For all experiments, we used a cropped training strategy that was adapted by Schirrmeister et al. [6], which uses crops, i.e. sliding input windows within a trial. This leads to many more training examples for the network. We used the training session of the data set to train the model and test it with the evaluation session. For each experiment, 10 runs were executed with random seeds and the results were averaged.

Hyperparameter optimization To maximize performance in our environment, a hyperparameter optimization will be run, in particular to find out where to split each model in a MTL architecture to maximize performance. (See 4.8)

Multi-task learning To evaluate the performance of each model in the MTL architecture, we trained the models with all 9 subjects, thus having 9 heads.

Forwarding to all heads The heads will see all samples, but the ones not corresponding to the subject will be labeled as *irrelevant*. (See 4.1.1)

Transfer Learning To evaluate the transferability of the learned features, the model will be trained with 8 subjects. For the remaining subject, a new head will be added to the model. For one experiment the backbone will be fine tuned and the new head trained, and one time the backbone will be frozen so only the new head is trained (See 4.4). For comparison, we will train the single task equivalent of the model with 8 subjects and fine tune it with the remaining subject, freeze the same layers as in the backbone of the MTL model and once with also resetting the not frozen layers.

Pretrained models We will train the MTL model with loaded pretrained backbone, heads, per subject heads and both backbone and heads, which are based on the single task counterpart of the MTL model. (See 4.2)

Task weighting All experiments will be run once with task weighting turned on and once turned off. (See 4.3)

Layer split sensitivity To evaluate the models' sensitivity to layer splits, we will train the MTL architecture from layer split 0, which is equivalent to separate models per subject, up to layer split 5 for MTL Deep4Net and MTL EEGNet and layer split 3 for MTL ShallowFBCSPNet, which is equivalent to pooled training.

4.11 Performance evaluation & statistics

To evaluate model performance we will use multiple metrics: a confusion matrix, validation accuracy, visualization of input windows and correlation maps. We will split the data set between the two sessions and use the first one as test and the second session as validation. We will not mix sessions, as it is not best-practice in BCI [47].

Chapter 5

Results

In this chapter, the results of the experiments are shown. Unless stated otherwise, results and visualizations of the *plain* models are shown (i.e., without pretraining, task weighting, etc., thus differing from the baseline only in architecture).

5.1 Validation of multi-task learning implementation

We used the braindecode library [6] for the Deep4Net, EEGNet and ShallowFBCSPNet models. We implemented our MTL architecture and their evaluation ourselves. To ensure a fair comparison, we first validated our implementation. We used the same hyperparameters for the models as in MTL and trained each subject in a single task experiment for 10 runs. Using the same preprocessing (4Hz - 38Hz), time window (4s, offset -0.5) and cropped windows, we compared the models to our MTL implementation in a single head configuration for each subject. Table 5.1 shows that for all three models, we reached similar performance with statistically insignificant differences.

Model	Single Task	MTL	p
Deep4Net	60.78	60.93	$p = .98$
EEGNet	68.91	68.94	$p = .432$
ShallowFBCSPNet	69.53	68.86	$p = .084$

Table 5.1: **Results for validation of multi-task learning implementation.** The results are given in percent. Single Task refers to the base model implementation by braindecode[6]. MTL refers to our implementation.

5.2 Result 1: Deep4Net

5.2.1 Layer split sensitivity

The MTL Deep4Net performs best with a layer split at 1. The accuracy rises significantly from the layer split 0, which corresponds to the single subject trained Deep4Net, to 1 and from there decreases in performance up to layer split 4. With layer split 5, the model reaches the level of performance of the pooled training strategy. As layer split 1 is the best performing split in MTL Deep4Net, this indicates that more complex heads perform better. (See 'Layer Sensitivity' in Figure 5.1)

5.2.2 Comparison of multi-task architecture to single-task architecture performance

Overall, the MTL Deep4Net performed better than the baseline Deep4Net on single and pooled training on absolute accuracy (see 5.2). The accuracies per subject can be found in 'Multihead' from Figure 5.1. Compared to the pooled training, the MTL Deep4Net achieved worse accuracy on subject 5 and 9, but better on every subject compared to the single trained model.

Dataset	Mode	Deep4Net	MTL Deep4Net
BCIC IV 2a	single	60.78	+7.70*
BCIC IV 2a	pooled	64.91	+3.56**

Table 5.2: **Decoding accuracy of Deep4Net baseline and of MTL Deep4Net.** The Deep4Net results are given in percent and MTL Deep4Net in absolute percent increase. BCIC IV 2a: BCI Compeition Dataset IV 2a. Mode: single refers to single subject training and pooled to all subjects trained. (Wilcoxon signed-rank test, *:$p = .002$, **:$p = .019$)

5.2.3 Forwarding all subjects to all heads

As visible in the "Forward to all heads" plot in figure 5.1, forwarding all data to every head underperforms in the MTL Deep4Net enormously, i.e., 47.48% vs 68.48% (Wilcoxon signed-rank test, $p = .002$).

5.2.4 Transfer learning performance

For the Deep4Net, the frozen and reset layers perform worse with -11.58% and -12.49% than the Deep4Net TL baseline. This might be because the frozen

layers, which represent the backbone in the MTL architecture, do not generalize enough in a pooled training setting for an unseen subject. The MTL Deep4Net TL approach reached an +2.42% in accuracy compared to the baseline and an even higher accuracy with a frozen backbone of +6.96% (See figure 5.3). With a frozen backbone the MTL Deep4Net performs only -0.37%, which is statistically insignificant (Wilcoxon signed-rank test, $p = .86$), worse in transfer learning than the MTL Deep4Net with the subject already trained. This hints at the fact that the backbone of MTL Deep4Net learns more generalized features for EEG data, which can be better shared amongst subjects than its single task counterpart.

Deep4Net TL	Deep4Net TL frozen	Deep4Net TL frozen reset layers	MTL Deep4Net TL	MTL Deep4Net TL frozen
61.14	49.57	48.65	63.56	68.11
	-11.58*	-12.49*	+2.42**	+6.96*

Table 5.3: **Transfer Learning Results for MTL Deep4Net.** The results in the first row are given in percent and the values of the second row are in absolute percent increase compared to Deep4Net TL. Stars indicate statistically significant differences. (Wilcoxon signed-rank test, *:$p < .001$, **:$p = .036$)

5.2.5 Pretrained models

Figure 5.4 shows how using a pretrained model for the MTL Deep4Net has not caused any significant performance change, except for the pretrained backbone with the pretrained head based on a pooled Deep4Net. While the result is statistically significant, it might also be achievable with more training epochs.

MTL Deep4Net	backbone	head	per subject head	backbone + head	backbone + per subject head
68.47	+0.33	+0.27	-0.11	+1.03*	+0.29

Table 5.4: **Pretrained Results for MTL Deep4Net.** The MTL Deep4Net result is given in percent and the values for each pretraining type in absolute percent increase. Stars indicate statistically significant differences. (Wilcoxon signed-rank test, *:$p = .01$)

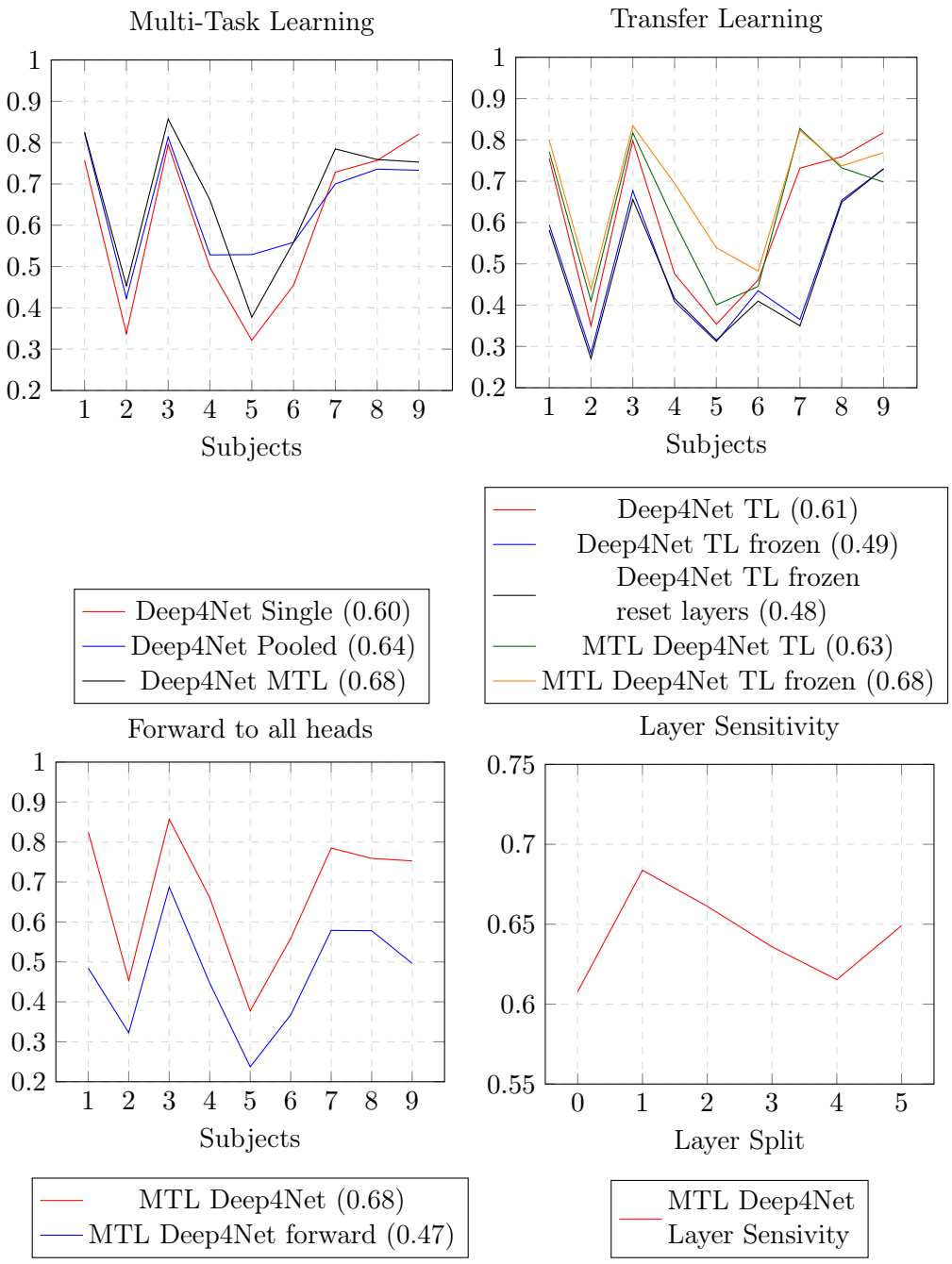

Figure 5.1: **MTL Deep4Net Results:** Multi-task learning, transfer learning, forward to all heads and layer sensitivity. Values in brackets are average accuracies.

5.3 Result 2: EEGNet

5.3.1 Layer split sensitivity

The MTL EEGNet performs best with a layer split at 0, which corresponds to the single subject trained EEGNet. The accuracy is significantly reduced from layer split 1 to 4, and from layer split 5 onward, performance improves, reaching the same level of the performance of the pooled training strategy. Layer split 1 being the best performing split in MTL EEGNet indicates that more complex heads perform better. (See 'Layer Sensitivity' in Figure 5.2)

5.3.2 Comparison of multi-task architecture to single-task architecture performance

The MTL EEGNet performed overall worse than the baseline EEGNet on single training on absolute accuracy but higher compared to pooled training (see 5.2). The accuracies per subject can be found in figure 'Multihead' 5.1. Compared to the pooled training, the MTL EEGNet achieved a better accuracy on subject 7, 8 and 9, but worse on every subject, except 8, compared to the single trained model.

Dataset	Mode	EEGNet	MTL EEGNet
BCIC IV 2a	single	66.80	-1.28*
BCIC IV 2a	pooled	62.97	+2.55**

Table 5.5: **Decoding accuracy of EEGNet baseline and of MTL EEGNet.** The EEGNet results are given in percent. BCIC IV 2a: BCI Compeition Dataset IV 2a. Mode: single refers to single subject training and pooled to all subjects trained. (Wilcoxon signed-rank test, *:$p = .002$, **:$p = .032$)

5.3.3 Forwarding all subjects to all heads

As visible in the "Forward to all heads" plot in figure 5.2, forwarding all data to every head performs slightly better on subject 2, 3 and 7 with the MTL EEGNet but drops in accuracy for every other subject. On average forwarding to all heads results in accuracy of 63.66% vs 65.52% with the default MTL EEGNet model. (Wilcoxon signed-rank test, $p = .002$).

5.3.4 Transfer learning performance

For the EEGNet the frozen and reset layers perform worse with -4.35% and -4.29% than the EEGNet TL baseline. This might be because the frozen layers,

which represent the backbone in the MTL architecture, do not generalize enough in a pooled training setting for an unseen subject. The MTL EEGNet TL approach reached an -0.66% in accuracy compared to the baseline and an even worse accuracy with a frozen backbone of +4.32% (See figure 5.6). The MTL EEGNet TL performs +1.1% better than the default MTL EEGNet model, which is statistically insignificant (Wilcoxon signed-rank test, $p = .731$). This hints at the fact that that the backbone of MTL EEGNet for transfer learning learns similar generalized features for EEG data as the MTL EEGNet.

EEGNet TL	EEGNet TL frozen	EEGNet TL frozen reset layers	MTL EEGNet TL	MTL EEGNet TL frozen
67.29	62.94	62.99	66.62	62.97
	-4.35*	-4.2*	-0.66	-4.32*

Table 5.6: **Transfer Learning Results for MTL EEGNet.** The results in the first row are given in percent and the values of the second row are in absolute percent increase. Stars indicate statistically significant differences. (Wilcoxon signed-rank test, *:$p < 0.001$)

5.3.5 Pretrained models

Figure 5.7 shows the use of a pretrained model for the MTL EEGNet has not caused any significant performance change, rather resulted in a decrease in accuracy. While the result is statistically significant, it might also be achievable with more training epochs.

MTL EEGNet	backbone	head	per subject head	backbone + head	backbone + per subject head
65.52	-0.32	-0.32	-0.44	-0.14	-0.83

Table 5.7: **Pretrained Results for MTL EEGNet.** The MTL EEGNet result is given in percent and the values for each pretraining type in absolute percent increase. There is no statistically significant difference. (Wilcoxon signed-rank test, *:$p > .13$)

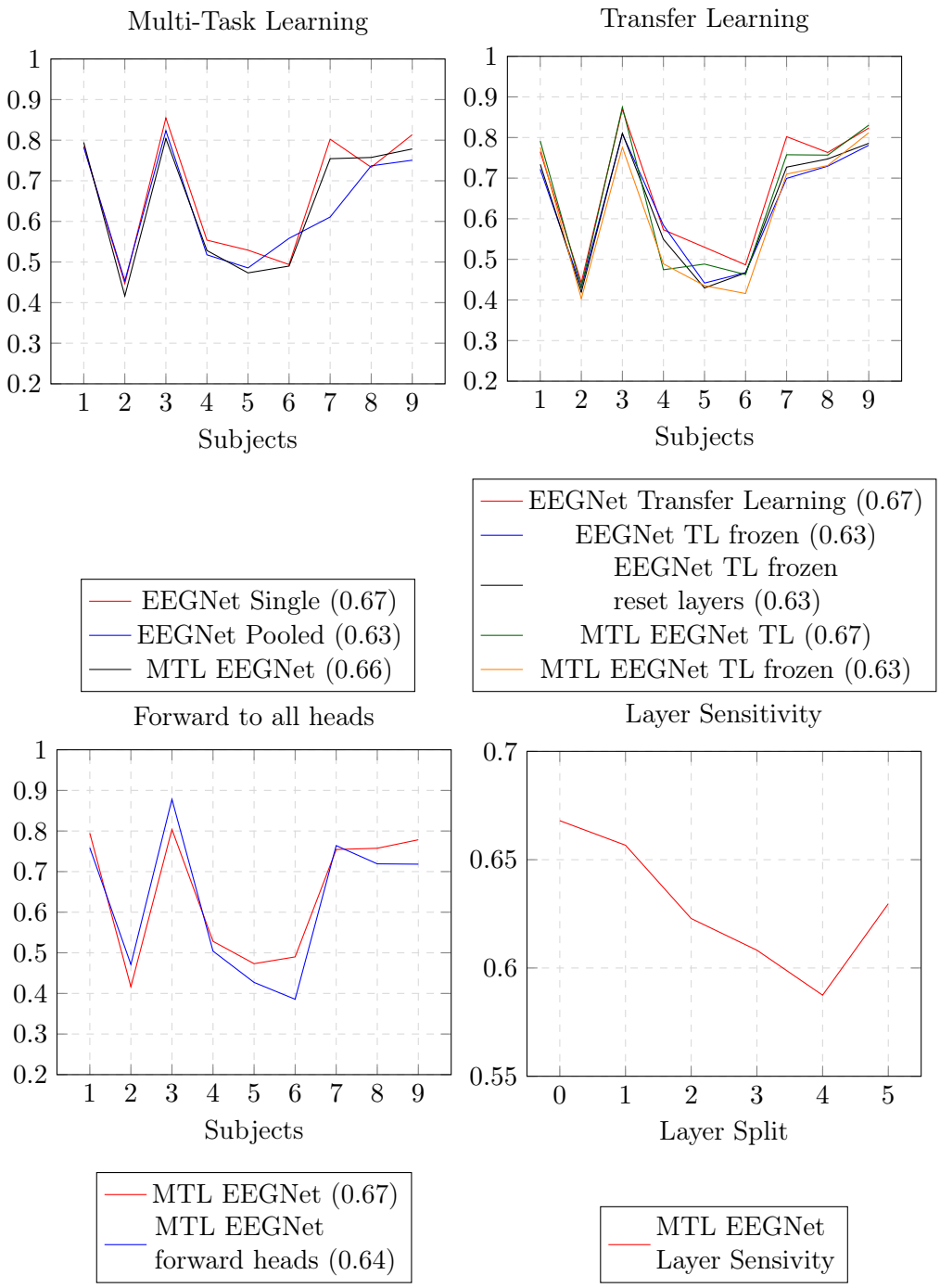

Figure 5.2: **MTL EEGNet Results:** Multi-task learning, transfer learning, forward to all heads and layer sensitivity. Values in brackets are average accuracies.

5.4 Result 3: ShallowFBCSPNet

5.4.1 Layer split sensitivity

The MTL ShallowFBCSPNet performs best with a layer split at 0, which corresponds to the single subject trained ShallowFBCSPNet. The accuracy decreases significantly from the layer split 1 to 2, and from there increases in performance by layer split 3, which is the performance of the pooled training strategy. Layer split 1 is the best performing split in MTL ShallowFBCSPNet, which indicates that more complex heads perform better. (See 'Layer Sensitivity' in Figure 5.3)

5.4.2 Comparison of multi-task architecture to single-task architecture performance

The MTL ShallowFBCSPNet performed slightly worse than the baseline ShallowFBCSPNet on single and better compared to pooled training on absolute accuracy (see 5.2). The accuracies per subject can be found in figure 'Multihead' 5.1. Compared to the pooled training, the MTL ShallowFBCSPNet achieved a better accuracy on subject 7, but failed to perform better on every other subject. The single task trained ShallowFBCSPNet had a higher accuracy on every subject compared to the MTL architecture. This hints that for a MTL architecture to perform well, more complex networks are needed where the ShallowFBCSPNet for its shallow design is not suited.

Dataset	Mode	ShallowFBCSPNet	MTL ShallowFBCSPNet
BCIC IV 2a	single	69.53	-1.90*
BCIC IV 2a	pooled	61.72	+5.92**

Table 5.8: **Decoding accuracy of ShallowFBCSPNet baseline and of MTL ShallowFBCSPNet.** The ShallowFBCSPNet results are given in percent. BCIC IV 2a: BCI Compeition Dataset IV 2a. Mode: single refers to single subject training and pooled to all subjects trained. Stars indicate statistically significant differences. (Wilcoxon signed-rank test, *:$p = .335$, **:$p < .001$)

5.4.3 Forwarding all subjects to all heads

As visible in the "Forward to all heads" plot in figure 5.3, forwarding all data to every head performs in MTL ShallowFBCSPNet slightly worse on some subjects. No subject could benefit from the added samples and the model had difficulties to identify samples of subject 3, which results in the poor accuracy. On average, forwarding to all heads results in accuracy of 59.42% vs 67.63% with the default MTL ShallowFBCSPNet model. (Wilcoxon signed-rank test, $p = .002$).

5.4.4 Transfer learning performance

For the ShallowFBCSPNet, the frozen and reset layers perform worse with -24.5% and -24.29% than the ShallowFBCSPNet TL baseline. This might be because the frozen layers, which represent the backbone in the MTL architecture, do not generalize enough in a pooled training setting for an unseen subject. The MTL ShallowFBCSPNet TL approach reached a -7.47% in accuracy compared to the baseline and an even worse accuracy with a frozen backbone of -8.94% (See figure 5.9). The MTL ShallowFBCSPNet TL performs +1.1% better than the default MTL ShallowFBCSPNet model, which is statistically insignificant (Wilcoxon signed-rank test, $p = .731$). This hints at the fact that the backbone of MTL ShallowFBCSPNet for transfer learning learns similar generalized features for EEG data as the MTL ShallowFBCSPNet.

ShallowNet TL	ShallowNet TL frozen	ShallowNet TL frozen reset layers	MTL ShallowNet TL	MTL ShallowNet TL frozen
69.42	44.87	45.13	61.95	60.48
	-24.5	-24.29	-7.47	-8.94

Table 5.9: **Transfer Learning Results for MTL ShallowFBCSPNet.** The results in the first row are given in percent and the values of the second row are in absolute percent increase. (Wilcoxon signed-rank test, $p < .001$)

5.4.5 Pretrained models

Figure 5.4 shows the use of a pretrained model for the MTL ShallowFBCSPNet has not caused any significant performance change, rather resulted in a decrease in accuracy. The result are statistically insignificant.

MTL ShallowNet	backbone	head	per subject head	backbone + head	backbone + per subject head
67.64	-0.08	-0.08	-0.14	+0.65	+0.21

Table 5.10: **Pretrained Results for MTL ShallowFBCSPNet.** The MTL ShallowFBCSPNet result is given in percent and the values for each pretraining type in absolute percent increase. There is no statistically significant difference. (Wilcoxon signed-rank test, *:$p > .13$)

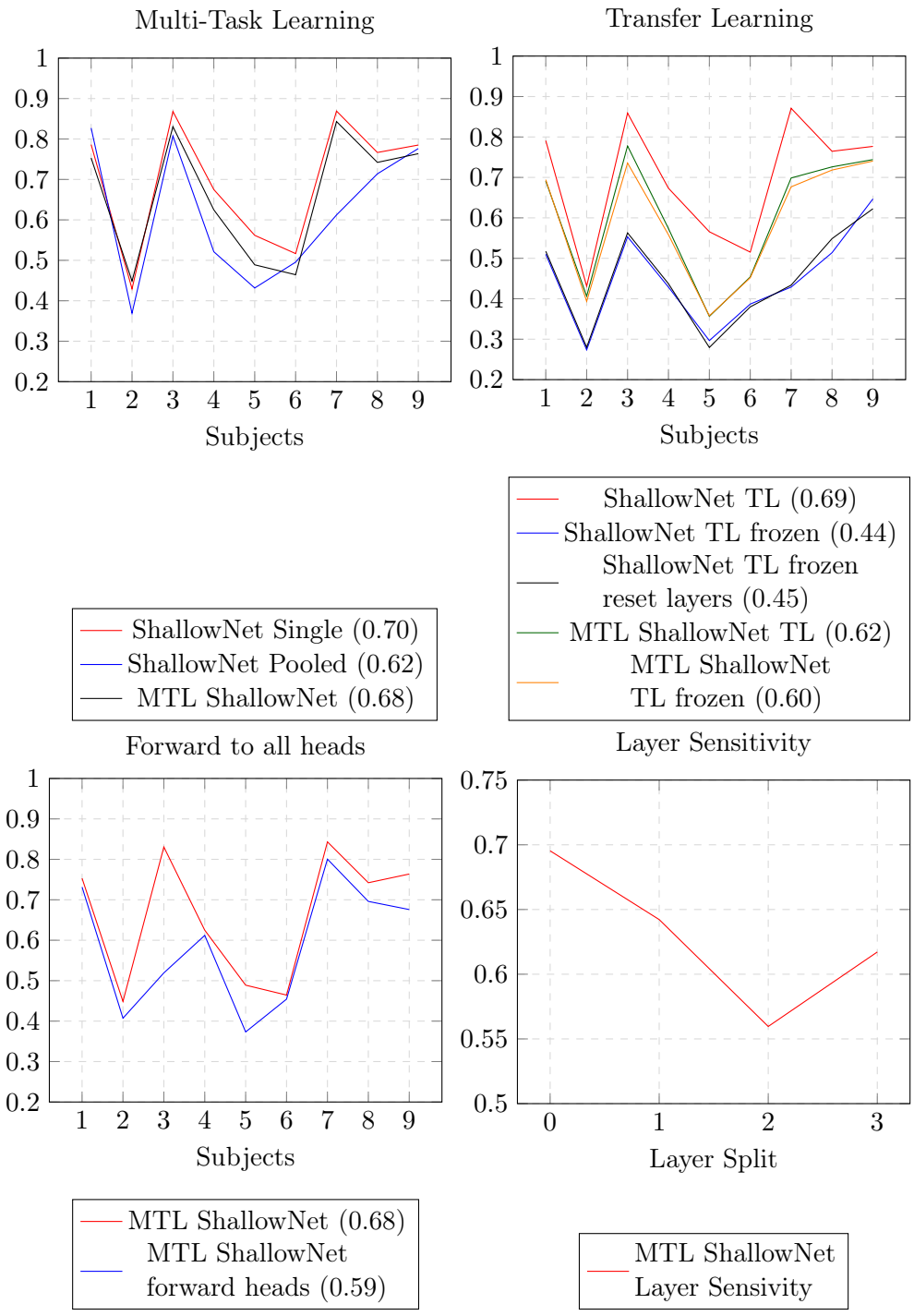

Figure 5.3: **MTL ShallowFBCSPNet Results:** Multi-task learning, transfer learning, forward to all heads and layer sensitivity. Values in brackets are average accuracies. For layout reasons ShallowFBCSPNet is shortened to ShallowNet in this figure.

5.5 Result 4: Task weighting

The level of performance with task weighting decreased slightly on average and the learned coefficients were similar for all subjects. Some subjects benefitted slightly, while others saw their performance decrease. As seen with forwarding to all heads for MTL EEGNet (see 5.3.3) the model has the best capabilities of the three models to differentiate between subjects. The learned coefficients for MTL EEGNet demonstrated the biggest difference between subjects, but had no big effect on training performance. When setting weights manually, one could observe that accuracy for single subjects improved, but to the detriment of the accuracy of other subjects. 5.11

	Coefficients	change in accuracy
Subject 1	0.6763	-0.84%
Subject 2	0.8713	+0.61%
Subject 3	0.6448	-1.65%
Subject 4	0.8487	+0.49%
Subject 5	0.8217	-1.04%
Subject 6	0.8686	-0.82%
Subject 7	0.7117	+0.39%
Subject 8	0.7037	-0.31%
Subject 9	0.5775	+0.73%

Table 5.11: **Weight coefficients of MTL EEGNet.** The learned coefficients for each task by the MTL EEGNet model. Change in accuracy is in comparison to the MTL EEGNet without weights. There is no statistically significant difference (Wilcoxon signed-rank test, $p > .492$).

5.6 Result 5: Confusion matrices

To examine the performance of the models with respect to the different classes, we will present their confusion matrices below. Models that performed above average (compared to other runs of the same model) were used to obtain the confusion matrices, so that potential differences between architectures would be more visible. In Figure 5.4, we compare the confusion matrices yielded by the models that were trained using the pooled training strategy to the corresponding MTL models. Overall, classification accuracy is balanced across classes for all models. While the confusion matrices are generally similar, slight improvements in classification accuracy can be observed for the MTL models. Both versions of Deep4Net seem to have a bias towards one of the hands, in the sense that they tend to confuse other labels with one of the hands more often. With the EEGNet, it can be seen that the MTL counterpart mistakes the feet or tongue for one of the two hands less often, while the tongue and feet are still mixed up approximately equally often. The MTL ShallowFBCSPNet confuses the two hands less frequently compared to the single-task trained ShallowFBCSPNet. The tongue is less confused with the feet, while the reverse is not true and in fact happens to a worse degree.

Figure 5.5 shows the confusion matrices of the MTL models trained with the strategy of forwarding the data of all subjects to all heads. As already apparent from the respective section of results 1-3, the Deep4Net performs significantly worse with this strategy compared to the other two models. It seems that the Deep4Net has difficulty associating the subject's data from a new session with the subject (and therefore classifies the samples as belonging to a different subject). It is particularly noticeable that the tongue is almost never correctly identified. In contrast, samples containing motor imagery of the right hand are correctly classified in nearly all cases.

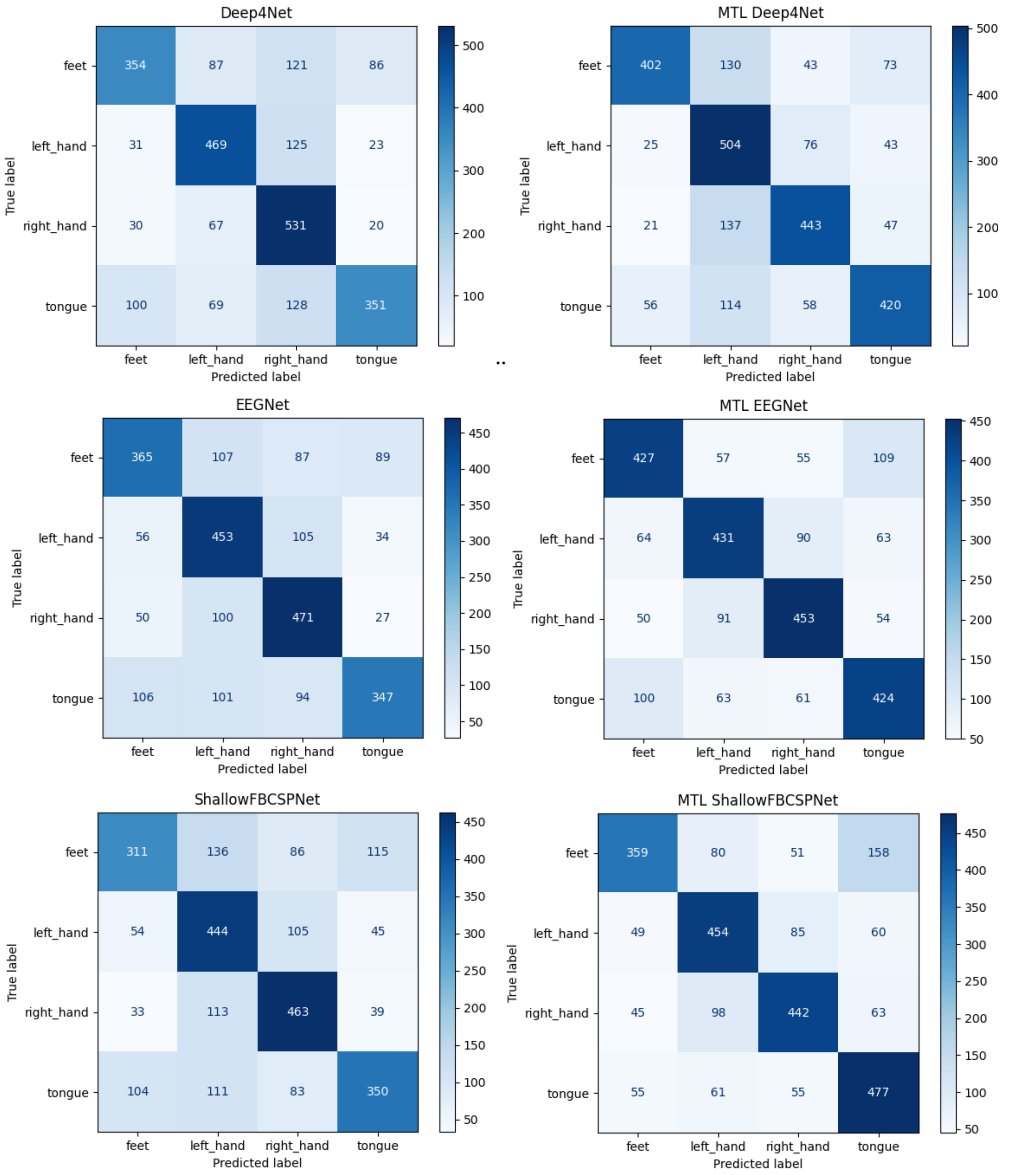

Figure 5.4: **Confusion matrices for all models trained on the pooled subjects (left) and their MTL counterparts (right).** Note that the scale of the Deep4Net differs from those of the other models, hence comparisons across these models should not be made based on color intensity.

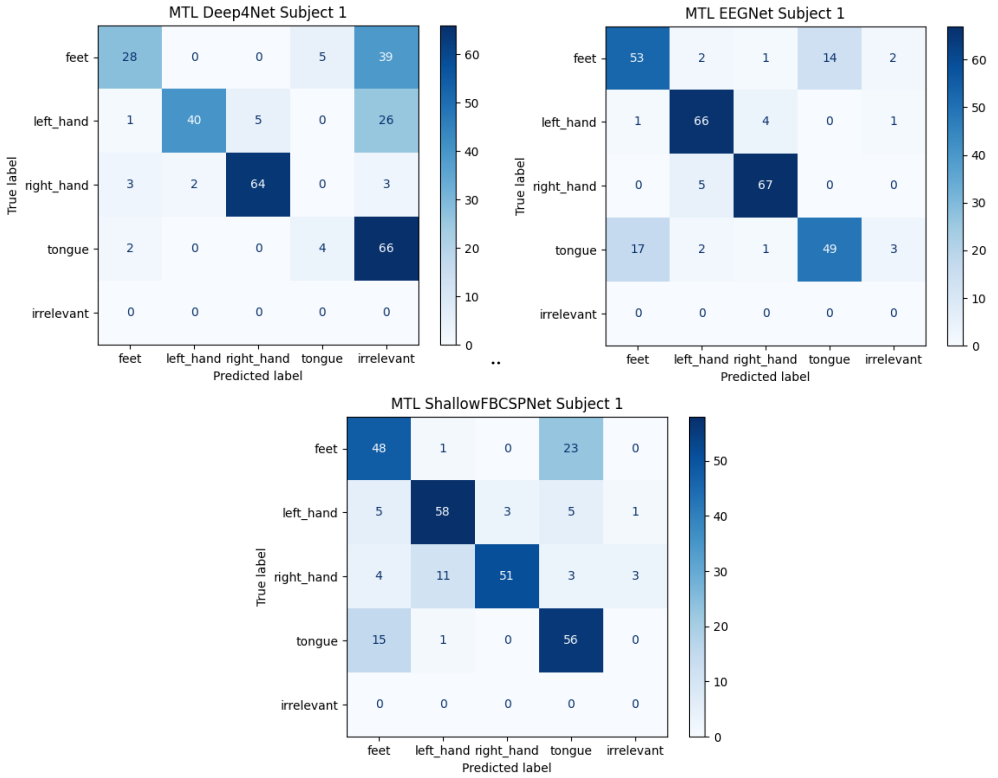

Figure 5.5: **Confusion matrices for the three MTL models when using the strategy to forward the data of all subjects to all heads.** As labels change depending on the subject, subject 1 (performance above average compared to other subjects) is shown as an example. The matrices were obtained by using subject-specific data only; as a result, the row for the label *irrelevant* (to be predicted when a sample does not belong to the given subject) is empty.

5.7 Result 6: Input-perturbation network-prediction correlation maps

The MTL Deep4Net seems to rely more on frequencies in the beta band for classification than the single-task model. The mean absolute gradient in the classifier layer that arises when the amplitude of the input frequencies is perturbed shows higher and narrower peaks in the beta band for most subjects, except for poorly performing subjects such as subject 2 and subject 5. The mean absolute gradient in the classifier layer for subject 1 is shown in Figure 5.6. A similar effect can be observed for the MTL EEGNet, but not for the MTL ShallowFBCSPNet, where the mean absolute gradient remained similar to its single-task counterpart.

In Figure 5.7, one can see that the frequencies in the beta band that are relevant for classification also originate at the expected source (at least for the right and left hand). For the MTL model, clearly defined spatial correlations with the electrodes C3 and C4 that are placed over the sensorimotor regions associated with hand movement are apparent in the beta band, while for the pooled Deep4Net this is only visible for the right hand. In return, a spatial correlation with the electrode CPz (associated with movement of feet) for the classification of feet can be seen in the alpha band. Much less spatially defined correlations are seen for the Deep4Net that was trained on only one subject.

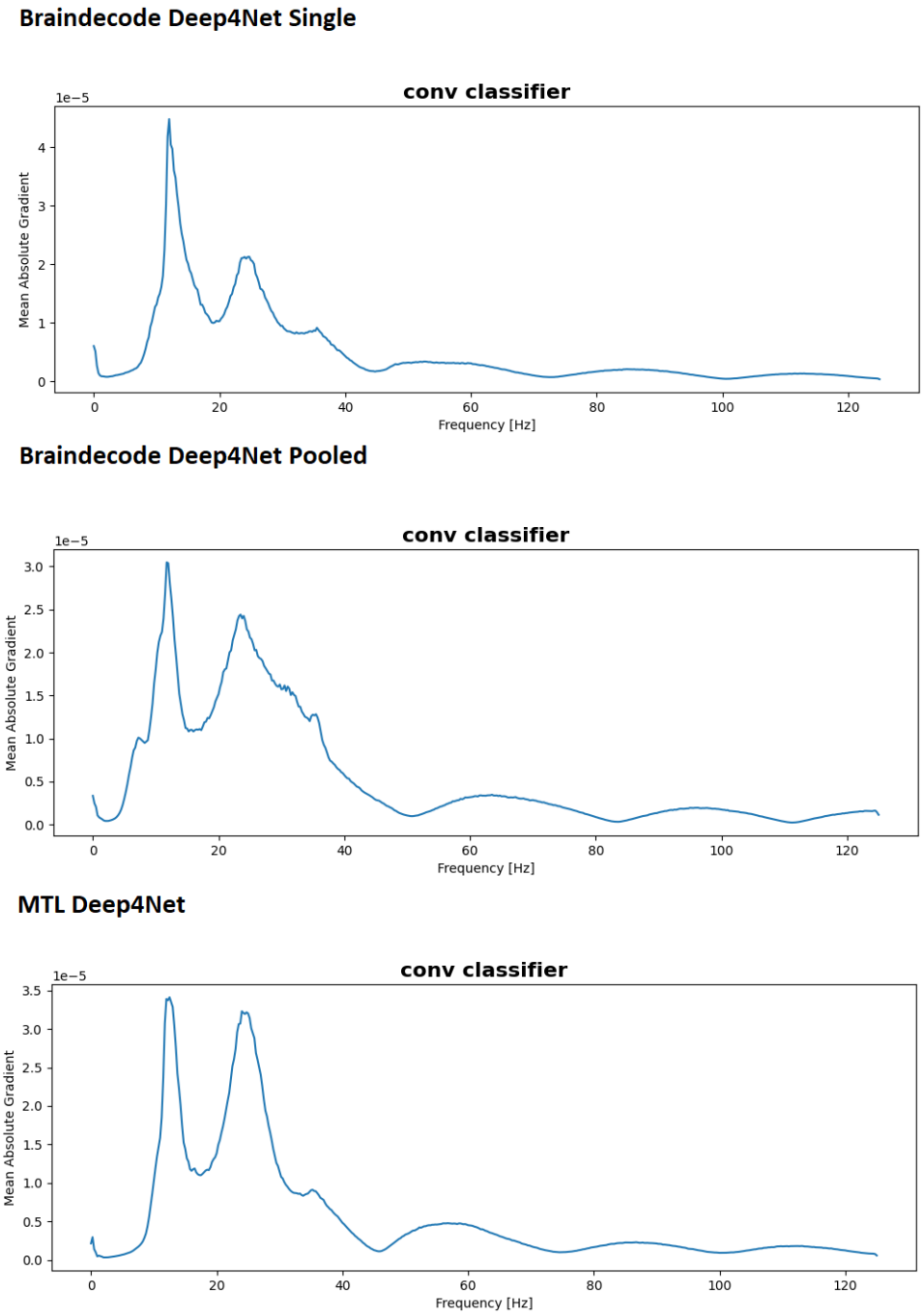

Figure 5.6: **Mean absolute gradient in the classifier layer** when perturbing amplitude of input frequencies for subject 1.

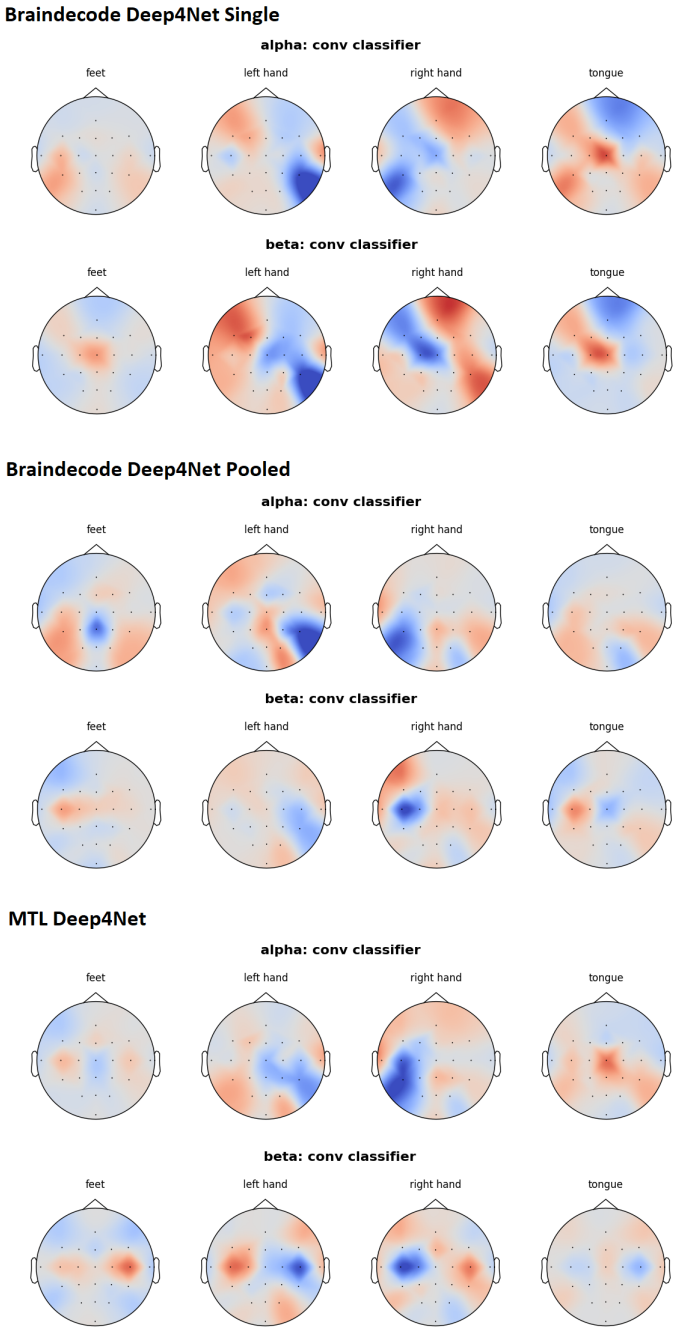

Figure 5.7: **Scalp plots of input-perturbation network-prediction correlation maps for the classifier layer** in the alpha and beta bands for subject 1. The colors indicate the correlation coefficient (min: blue, max: red).

5.8 Result 7: Most-activating input windows

The features used by the models become more complex from lower to higher layers, which is expected when using ConvNets. The respective single most-activating input windows for the Deep4Net (for electrode C4) when using data from subject 1 are shown in Figure 5.8 as an example. One can see that features evolve across layers from parts of simple sinusoids into complex oscillatory patterns. The ShallowFBCSPNet, due to the small number of layers, was not able to learn such complex features. However, the features in the first layer are already slightly more complex compared to the features in the first layer of the Deep4Net. The characteristics of the features in the second layer are similar to those of the third layer of the deeper model. Unfortunately, due to implementation challenges relating to architectural differences, we were not able to obtain these plots for the EEGNet.

We hypothesized that the features would reflect differences between the subjects. While the exact features varied from subject to subject, the characteristics of those are very similar across subjects. Quantifying the differences is difficult; unfortunately, we could not use these feature visualizations to make statements about the similarity of subjects to each other. However, for subject 4 (which benefited the most from MTL with Deep4Net), we analyzed the features in the first layer. We estimated the frequency of the features by fitting a sinusoidal function. We could see that the preferred frequency differed between training strategies. Namely, for example for the Beta band, the frequency increased from just under 13Hz for the single model to 15.6Hz for the pooled model to almost 17Hz for the MTL model. Due to time constraints, an analysis for other subjects was omitted.

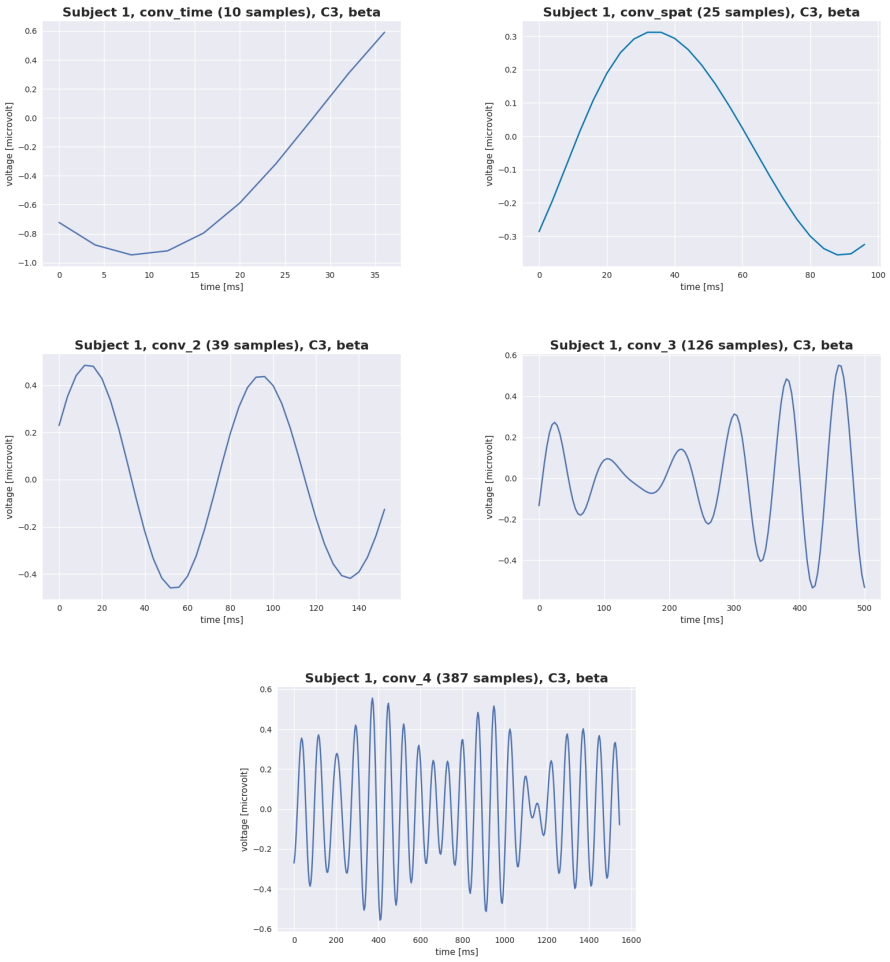

Figure 5.8: **The respective most-activating input window for the MTL Deep4Net** with data from subject 1 in the beta band for all layers up to the last one before the classifier layer.

Chapter 6

Discussion and outlook

6.1 Conclusions

Our results show that MTL is able to improve the performance of CNNs on decoding MI-EEG signals, but whether performance does improve is dependent on the underlying architecture. In the following section, we will discuss possible reasons for the difference in results between architectures.

Of the models evaluated, only the Deep4Net was able to achieve higher accuracy when using MTL compared to both single and pooled strategies (result 1). The two other models, EEGNet and ShallowFBCSPNet, both perform best when a separate model is used per subject (result 1-2). Sharing the first layer, however, performs better compared to pooled training (i.e. sharing all layers). One reason for the Deep4Net being able to profit from MTL could be its ability to learn complex features. This is supported by Zhang et al., according to which deep MTL models perform better on images than shallow models, since they learn more powerful feature representations [30]. In contrast, Caruana finds that the usefulness of MTL is independent of network size [28]. Therefore, it cannot be ruled out that the gain in performance was not achieved by sharing knowledge, but possibly only by a regularization effect. Whereby, one could discuss whether such a regularization effect represents an implicit sharing of knowledge. The apparent greater emphasis on frequencies in the beta band in the Deep4MTL compared with the single and pooled models (result 6), and the increased performance, may indicate knowledge sharing between subjects. We believe that more focus being laid on the beta band might be due to event related potentials in the beta band frequencies being more consistent across subjects compared to frequencies in the alpha band, and that the MTL model is able to exploit this better. A similar effect was observed with the EEGNet, and yet, the desired outperformance of the single model was not achieved with MTL. It did, however, improve performance over the pooled model. The Deep4Net is also the only model for which pooled training achieves better results than training subjects individually. It has more learnable parameters than the other two models, and therefore requires more examples to avoid overfitting on the training data. Thus, it seems reasonable

that it benefits from more samples. With using MTL we show that apparently it is sufficient and even improves performance when only lower layers are provided with the additional training samples. The number of parameters of the shallow model is closer to the number of parameters of the deep model than to that of the EEGNet (Deep4Net: 170'000, ShallowFBCSPNet: 100'000, EEGNet: 1'000; approximate values from [31]). Therefore, based only on the capacity of the models, one might expect more similar results between the Deep4Net and the ShallowFBCSPNet.

Another notable difference from the Deep4Net to the other two models is the following: The Deep4Net was intended as a general decoder for brain signals. It is inspired by computer vision models and is not specifically designed for decoding EEG data (except that the input layers have been adapted for EEG input). In contrast, the other two models incorporate expert knowledge in EEG decoding. The ShallowFBCSPNet was implemented analogously to the FBCSP pipeline, one of the most successful methods in decoding EEG data. EEGNet was also designed specifically for this task. In result 5, it can be seen that the two models had significantly less difficulty distinguishing different subjects compared to the Deep4Net. This suggests that the internal representations of subjects in the two models specialized for EEG are more distinct from each other. We believe it is possible that this leads to more conflicting gradients when sharing layers (and through this to negative transfer), which could be the reason why these two models perform worse with the pooled training strategy compared to single-subject training. When sharing less layers (e.g. by using MTL with layer split 1), the performance is again increased.

Regarding the sensitivity of the models to the layer split, it can be confirmed that the features in the lowest layers (where temporal and spatial convolutions are performed by the evaluated models) have the highest transferability.

The transfer learning performance for the Deep4Net could be improved by using MTL. This indicates that the learned backbone is more general compared to the same layers of the single and pooled trained Deep4Nets. However, the single and pooled trained models might perform better with a different approach, such as transferring other layers. No improvement was achieved in this area for the other models.

Task weighting did not provide an advantage for any of the MTL models. We suspect that it is due to the fact that our tasks, despite the high inter-subject variability, are still more similar to each other compared to tasks in typical use cases of MTL. This is supported by the fact that the learned weights were more different for models that performed better with the *forward all* training strategy.

6.2 Limitations

We were not able to achieve the performances of the models published in the respective papers. We attribute this to less extensive hyperparameter optimization. Since we make our comparisons among models with the same hyperparameters, this does not affect the validity of our comparisons. The data set we used is very small. With more subjects, different results may possibly be obtained. The use of visualization methods resulted in a very large number of plots that had to be considered. We looked for changes across subjects and models that seemed consistent, and showed them using an example. However, most of these differences found are subject to our perception and have not yet been quantified.

6.3 Future work

Our conclusions suggest, above all, a need to prevent negative transfer of knowledge. We see several ways to address this. One could try a different MTL architecture, which also learns *what* to share, e.g. sluice networks. Another way would be to cluster the subjects so as not to train subjects with together that negatively affect each other. However, we are not sure which metric would be suitable for this. If this were achieved, heads could also be used for groups for subjects instead of only individual subjects. Finally, the use of an optimizer specifically designed for MTL could be beneficial, such as the recently proposed PCGrad [48].

Bibliography

[1] T. C. Major and J. M. Conrad, "A survey of brain computer interfaces and their applications," in *IEEE SOUTHEASTCON 2014*, 2014, pp. 1–8.

[2] J. Zhang and M. Wang, "A survey on robots controlled by motor imagery brain-computer interfaces," *Cognitive Robotics*, vol. 1, pp. 12–24, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S266724132100001X

[3] M. Rashid, N. Sulaiman, A. P. P. Abdul Majeed, R. M. Musa, A. F. Ab. Nasir, B. S. Bari, and S. Khatun, "Current status, challenges, and possible solutions of eeg-based brain-computer interface: A comprehensive review," *Frontiers in Neurorobotics*, vol. 14, p. 25, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnbot.2020.00025

[4] H.-J. Yoo, "Deep convolution neural networks in computer vision: a review," *IEIE Transactions on Smart Processing and Computing*, vol. 4, pp. 35–43, Feb 2015.

[5] X. Zhang, L. Yao, X. Wang, J. Monaghan, and D. McAlpine, "A survey on deep learning based brain computer interface: Recent advances and new frontiers," *CoRR*, vol. abs/1905.04149, 2019. [Online]. Available: http://arxiv.org/abs/1905.04149

[6] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball, "Deep learning with convolutional neural networks for eeg decoding and visualization," *Human Brain Mapping*, Aug 2017. [Online]. Available: http://dx.doi.org/10.1002/hbm.23730

[7] A. Azab, M. Arvaneh, J. Toth, and L. Mihaylova, *A review on transfer learning approaches in brain–computer interface.* The Institution of Engineering and Technology, 09 2018.

[8] K. Zhang, G. Xu, X. Zheng, H. Li, S. Zhang, Y. Yu, and R. Liang, "Application of transfer learning in eeg decoding based on brain-computer interfaces: A review," *Sensors*, vol. 20, no. 21, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/21/6321

[9] L. Liebel and M. Körner, "Multidepth: Single-image depth estimation via multi-task regression and classification," *CoRR*, vol. abs/1907.11111, 2019. [Online]. Available: http://arxiv.org/abs/1907.11111

[10] G. Crichton, S. Pyysalo, B. Chiu, and A. Korhonen, "A neural network multi-task learning approach to biomedical named entity recognition," *BMC Bioinformatics*, vol. 18, 08 2017.

[11] R. G. Bickford, "Physical control of the mind. toward a psychocivilized society. josé m. r. delgado. harper and row, new york, 1969. xxii, 282 pp., world perspectives, vol. 41," *Science*, vol. 169, no. 3946, pp. 666–666, 1970. [Online]. Available: https://science.sciencemag.org/content/169/3946/666.1

[12] E. Fetz, "Operant conditioning of cortical unit activity," *Science*, vol. 163, pp. 955 – 958, 1969.

[13] R. P. N. Rao, *Brain-Computer Interfacing: An Introduction*. Cambridge University Press, 2013.

[14] I. Iturrate, R. Chavarriaga, and J. del R. Millán, "Chapter 23 - general principles of machine learning for brain-computer interfacing," in *Brain-Computer Interfaces*, ser. Handbook of Clinical Neurology, N. F. Ramsey and J. del R. Millán, Eds. Elsevier, 2020, vol. 168, pp. 311–328. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444639349000238

[15] 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://doi.org/10.1088/1741-2552/ab260c

[16] R. Leeb, C. Keinrath, D. Friedman, C. Guger, R. Scherer, C. Neuper, M. Garau, A. Antley, A. Steed, M. Slater, and G. Pfurtscheller, "Walking by thinking: The brainwaves are crucial, not the muscles!" *Presence*, vol. 15, pp. 500–514, 10 2006.

[17] M. Lotze and U. Halsband, "Motor imagery," *Journal of Physiology-Paris*, vol. 99, no. 4, pp. 386–395, 2006, brain Imaging in Neurosciences - An Interdisciplinary Approach. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0928425706000210

[18] G. Pfurtscheller, C. Guger, G. Müller, G. Krausz, and C. Neuper, "Brain oscillations control hand orthosis in a tetraplegic," *Neuroscience Letters*, vol. 292, no. 3, pp. 211–214, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304394000014713

[19] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958. [Online]. Available: http://dx.doi.org/10.1037/h0042519

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

[21] A. K. et al. Cs231n convolutional neural networks for visual recognition. [Online]. Available: cs231n.github.io/neural-networks-1/

[22] P. C. Kainen, V. Kůrková, and M. Sanguineti, *Approximating Multivariable Functions by Feedforward Neural Nets.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 143–181. [Online]. Available: https://doi.org/10.1007/978-3-642-36657-4_5

[23] F. Bre, J. M. Gimenez, and V. D. Fachinotti, "Prediction of wind pressure coefficients on building surfaces using artificial neural networks," *Energy and Buildings*, vol. 158, pp. 1429–1441, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378778817325501

[24] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," in *2017 International Conference on Communication and Signal Processing (ICCSP)*, 2017, pp. 0588–0592.

[25] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *Artificial Neural Networks and Machine Learning – ICANN 2018*, V. Kůrková, Y. Manolopoulos, B. Hammer, L. Iliadis, and I. Maglogiannis, Eds. Cham: Springer International Publishing, 2018, pp. 270–279.

[26] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[27] C. Sammut and G. I. Webb, Eds., *Generalization Performance.* Boston, MA: Springer US, 2010, pp. 454–454. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_329

[28] R. Caruana, *Multitask Learning.* Boston, MA: Springer US, 1998, pp. 95–133. [Online]. Available: https://doi.org/10.1007/978-1-4615-5529-2_5

[29] E. Hüllermeier, T. Fober, and M. Mernberger, *Inductive Bias.* New York, NY: Springer New York, 2013, pp. 1018–1018. [Online]. Available: https://doi.org/10.1007/978-1-4419-9863-7_927

[30] Y. Zhang and Q. Yang, "A survey on multi-task learning," *CoRR*, vol. abs/1707.08114, 2017. [Online]. Available: http://arxiv.org/abs/1707.08114

[31] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance, "Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces," *Journal of Neural Engineering*, vol. 15, no. 5, p. 056013, Jul 2018. [Online]. Available: http://dx.doi.org/10.1088/1741-2552/aace8c

[32] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" 2014.

[33] A. Uran, C. V. Gemeren, R. van Diepen, R. Chavarriaga, and J. del R. Millán, "Applying transfer learning to deep learned models for EEG analysis," *CoRR*, vol. abs/1907.01332, 2019. [Online]. Available: http://arxiv.org/abs/1907.01332

[34] M. Alamgir, M. Grosse-Wentrup, and Y. Altun, "Multitask learning for brain-computer interfaces," in *JMLR Workshop and Conference Proceedings Volume 9: AISTATS 2010*, Max-Planck-Gesellschaft. Cambridge, MA, USA: JMLR, May 2010, pp. 17–24.

[35] P. Autthasan, R. Chaisaen, T. Sudhawiyangkul, P. Rangpong, S. Kiatthaveephong, N. Dilokthanakul, G. Bhakdisongkhram, H. Phan, C. Guan, and T. Wilaiprasitporn, "Min2net: End-to-end multi-task learning for subject-independent motor imagery eeg classification," 2021.

[36] Y. Song, D. Wang, K. Yue, N. Zheng, and Z.-J. M. Shen, "Eeg-based motor imagery classification with deep multi-task learning," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–8.

[37] R. Mormont, P. Geurts, and R. Maree, "Multi-task pre-training of deep neural networks for digital pathology," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 2, p. 412–421, Feb 2021. [Online]. Available: http://dx.doi.org/10.1109/JBHI.2020.2992878

[38] T. Gong, T. Lee, C. Stephenson, V. Renduchintala, S. Padhy, A. Ndirango, G. Keskin, and O. H. Elibol, "A comparison of loss weighting strategies for multi task learning in deep neural networks," *IEEE Access*, vol. 7, pp. 141 627–141 632, 2019.

[39] R. Cipolla, Y. Gal, and A. Kendall, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7482–7491.

[40] L. Liebel and M. Körner, "Auxiliary tasks in multi-task learning," *CoRR*, vol. abs/1805.06334, 2018. [Online]. Available: http://arxiv.org/abs/1805.06334

[41] K. G. Hartmann, R. T. Schirrmeister, and T. Ball, "Hierarchical internal representation of spectral features in deep convolutional networks trained for EEG decoding," *CoRR*, vol. abs/1711.07792, 2017. [Online]. Available: http://arxiv.org/abs/1711.07792

[42] C. Brunner, R. Leeb, G. R. Müller-Putz, A. Schlögl, and G. Pfurtscheller, "Bci competition 2008 graz data set a experimental paradigm." 2008. [Online]. Available: http://www.bbci.de/competition/iv/desc_2a.pdf

[43] M. Tangermann, K.-R. Müller, A. Aertsen, N. Birbaumer, C. Braun, C. Brunner, R. Leeb, C. Mehring, K. Miller, G. Mueller-Putz, G. Nolte,

G. Pfurtscheller, H. Preissl, G. Schalk, A. Schlögl, C. Vidaurre, S. Waldert, and B. Blankertz, "Review of the bci competition iv," *Frontiers in Neuroscience*, vol. 6, p. 55, 2012. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2012.00055

[44] V. Jayaram and A. Barachant, "MOABB: trustworthy algorithm benchmarking for BCIs," *Journal of Neural Engineering*, vol. 15, no. 6, p. 066011, Sep 2018. [Online]. Available: https://doi.org/10.1088/1741-2552/aadea0

[45] A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen, and M. Hämäläinen, "MEG and EEG data analysis with MNE-Python," *Frontiers in Neuroscience*, vol. 7, p. 267, 2013. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2013.00267

[46] L. Biewald, "Experiment tracking with weights and biases," 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[47] E. Thomas, M. Dyson, and M. Clerc, "An analysis of performance evaluation for motor-imagery based BCI," *Journal of Neural Engineering*, vol. 10, no. 3, p. 031001, May 2013. [Online]. Available: https://doi.org/10.1088/1741-2560/10/3/031001

[48] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *CoRR*, vol. abs/2001.06782, 2020. [Online]. Available: https://arxiv.org/abs/2001.06782

List of Figures

2.1 **Processing steps of a brain-computer interface**: The acquired signal is preprocessed and relevant features are selected. A decoder receives the features as an input and infers an action. The user receives feedback either explicitly or via their own senses [14]. 4

2.2 **(A) Subject wearing a 32-electrode EEG cap. (B) International 10–20 system for standardized EEG electrode locations on the head** [13]. 5

2.3 **Examples of EEG rhythms and their frequency range** [13]. 6

2.4 **A representation of a biological neuron (left) and its mathematical model (right)** [21]. 7

2.5 **Rectified Linear Unit (ReLU) activation function**, which is zero when $x < 0$ and then linear with slope 1 when $x > 0$ [21]. . . 7

2.6 **Artificial Neural Network** [23] 8

2.7 **Different learning processes between (a) traditional machine learning and (b) transfer learning** [26]. 9

2.8 **Illustration of a multi-task neural network** [28]. Unlike a single-task model, a multi-task model has outputs that are associated with multiple tasks. In this illustration, only one output per task is shown, but there can be (and usually will be in practice) multiple outputs. 10

4.1 **Multi-task architecture proposed by Mormont et al.**: From the input $\boldsymbol{x}$ a shared representation $\boldsymbol{\theta_s}$ is computed. Samples from this representation are forwarded to the respective heads. The loss is calculated per task and aggregated [37]. 16

4.2 **MTL Layer Splits** 17

5.1 **MTL Deep4Net Results:** Multi-task learning, transfer learning, forward to all heads and layer sensitivity. Values in brackets are average accuracies. 26

5.2 **MTL EEGNet Results:** Multi-task learning, transfer learning, forward to all heads and layer sensitivity. Values in brackets are average accuracies. 29

5.3 **MTL ShallowFBCSPNet Results:** Multi-task learning, transfer learning, forward to all heads and layer sensitivity. Values in brackets are average accuracies. For layout reasons ShallowFBCSPNet is shortened to ShallowNet in this figure. 32

5.4 **Confusion matrices for all models trained on the pooled subjects (left) and their MTL counterparts (right).** Note that the scale of the Deep4Net differs from those of the other models, hence comparisons across these models should not be made based on color intensity. 35

5.5 **Confusion matrices for the three MTL models when using the strategy to forward the data of all subjects to all heads.** As labels change depending on the subject, subject 1 (performance above average compared to other subjects) is shown as an example. The matrices were obtained by using subject-specific data only; as a result, the row for the label *irrelevant* (to be predicted when a sample does not belong to the given subject) is empty. 36

5.6 **Mean absolute gradient in the classifier layer** when perturbing amplitude of input frequencies for subject 1. 38

5.7 **Scalp plots of input-perturbation network-prediction correlation maps for the classifier layer** in the alpha and beta bands for subject 1. The colors indicate the correlation coefficient (min: blue, max: red). 39

5.8 **The respective most-activating input window for the MTL Deep4Net** with data from subject 1 in the beta band for all layers up to the last one before the classifier layer. 41

C.1 **ShallowFBCSPNet architecture** [6]. C-1

C.2 **Deep4Net architecture** [6]. C-2

C.3 **EEGNet architecture** [31]. C-3

List of Tables

4.2 **Hyperparameter Search Configuration.** 20

5.1 **Results for validation of multi-task learning implementation.** The results are given in percent. Single Task refers to the base model implementation by braindecode[6]. MTL refers to our implementation. 23

5.2 **Decoding accuracy of Deep4Net baseline and of MTL Deep4Net.** The Deep4Net results are given in percent and MTL Deep4Net in absolute percent increase. BCIC IV 2a: BCI Competion Dataset IV 2a. Mode: single refers to single subject training and pooled to all subjects trained. (Wilcoxon signed-rank test, *:$p = .002$, **:$p = .019$) 24

5.3 **Transfer Learning Results for MTL Deep4Net.** The results in the first row are given in percent and the values of the second row are in absolute percent increase compared to Deep4Net TL. Stars indicate statistically significant differences. (Wilcoxon signed-rank test, *:$p < .001$, **:$p = .036$) 25

5.4 **Pretrained Results for MTL Deep4Net.** The MTL Deep4Net result is given in percent and the values for each pretraining type in absolute percent increase. Stars indicate statistically significant differences. (Wilcoxon signed-rank test, *:$p = .01$) 25

5.5 **Decoding accuracy of EEGNet baseline and of MTL EEGNet.** The EEGNet results are given in percent. BCIC IV 2a: BCI Competion Dataset IV 2a. Mode: single refers to single subject training and pooled to all subjects trained. (Wilcoxon signed-rank test, *:$p = .002$, **:$p = .032$) 27

5.6 **Transfer Learning Results for MTL EEGNet.** The results in the first row are given in percent and the values of the second row are in absolute percent increase. Stars indicate statistically significant differences. (Wilcoxon signed-rank test, *:$p < 0.001$) . 28

5.7 **Pretrained Results for MTL EEGNet.** The MTL EEGNet result is given in percent and the values for each pretraining type in absolute percent increase. There is no statistically significant difference. (Wilcoxon signed-rank test, *:$p > .13$) 28

5.8 **Decoding accuracy of ShallowFBCSPNet baseline and of MTL ShallowFBCSPNet.** The ShallowFBCSPNet results are given in percent. BCIC IV 2a: BCI Compeition Dataset IV 2a. Mode: single refers to single subject training and pooled to all subjects trained. Stars indicate statistically significant differences. (Wilcoxon signed-rank test, *:$p = .335$, **:$p < .001$) 30

5.9 **Transfer Learning Results for MTL ShallowFBCSPNet.** The results in the first row are given in percent and the values of the second row are in absolute percent increase. (Wilcoxon signed-rank test, $p < .001$) 31

5.10 **Pretrained Results for MTL ShallowFBCSPNet.** The MTL ShallowFBCSPNet result is given in percent and the values for each pretraining type in absolute percent increase. There is no statistically significant difference. (Wilcoxon signed-rank test, *:$p > .13$) 31

5.11 **Weight coefficients of MTL EEGNet.** The learned coefficients for each task by the MTL EEGNet model. Change in accuracy is in comparison to the MTL EEGNet without weights. There is no statistically significant difference (Wilcoxon signed-rank test, $p > .492$). 33

Appendix A

Project Description

Zürcher Hochschule
für Angewandte Wissenschaften

School of Engineering

Transfer Learning of Deep Neural Network Representations of Brain Activity

BA21_stdm_03

BetreuerInnen:	Thilo Stadelmann, stdm Ricardo Chavarriaga, chav
Fachgebiete:	Datenanalyse (DA) Software (SOW)
Studiengang:	IT
Zuordnung:	Institut für angewandte Informationstechnologie (InIT)
Gruppengrösse:	2

Kurzbeschreibung:

There is an increasing interest of using machine learning and artificial intelligence techniques to analyse brain activity and provide insights on the neurological basics of cognitive processes. Building up on the success of deep neural networks in the analysis and recognition of other types of signals (e.g. image or speech processing), this project is oriented towards their use for analysing biological neural data.

ZHAW students and researchers have been successful on the development of deep learning methods in multiple applications and this project is aimed at applying these tools to the analysis of electrical activity of the brain. Specifically, the field of brain machine interfaces (BMI) aims at decoding this activity in real-time, allowing interaction with external devices. Application of deep neural networks in this field has been challenging since the amount of available data is considerable smaller than in other fields like computer vision and natural language processing.

The goal of this thesis is to implement and evaluate deep learning approaches suitable for the characteristics of brain-machine interfacing. Namely, large signal variability, low signal-to-noise ration and small data sets. Of particular interest is the use of explainable methods that help understanding the underlying cognitive processes reflected by the brain signal. During the project, students should perform the following tasks:

- a literature study on current state-of-the-art systems
- setting up the development envirnoment (locally and on the InIT GPU cluster)
- designing and carrying out experiments to compare different classification approaches, comprising non-DL classification systems (e.g. LDAs or SVMs) and DL-based methods, including group analysis of the data, training/testing individual classifiers per individual, transfer learning approaches
- writing a scientific report with a focus on motivation, argumentation & results

Voraussetzungen:

No prior knowledge of neuroscience or machine learning / AI is expected at this stage. Necessary, however, is a passion for the project topic, a curiosity for research work (creating and evaluating hypotheses and striving to find the reasons behind observations), a pragmatic approach to experimentation, a very good command of programming in general, and a very good previous track record in the study programme. Data and code from previous BMI experiments are available.

Details for above:

- Group analysis of the data (i.e., pooling together data from several people)
- Training/testing individual classifiers per individual
- Transfer learning approaches(i.e., training on a set of individuals ; test on a different person)

Useful references:

- A Beginners Guide to Brain-Computer Interface and Convolutional Neural Networks
- Chaudhary, U., Birbaumer, N. & Ramos-Murguialday, A. Braincomputer interfaces for communication and rehabilitation. Nat Rev Neurol 12, 513525 (2016). https://doi.org/10.1038/nrneurol.2016.113
- Yannick Roy et al. Deep learning-based electroencephalography analysis: a systematic review. 2019 J. Neural Eng. 16 05100. https://doi.org/10.1088/1741-2552/ab260c

With good results, the joint publication of a scientific paper together with the supervisors is likely.

Die Arbeit ist vereinbart mit:
Benjamin Bertalan (bertaben)
Gian Andri Hess (hessgia1)

Appendix B

Contact information and code access

If you have any questions regarding this work, you can contact us via gian.hess@gmail.com or benjamin@brtln.com.

Appendix C

Figures

C.1 Architectures

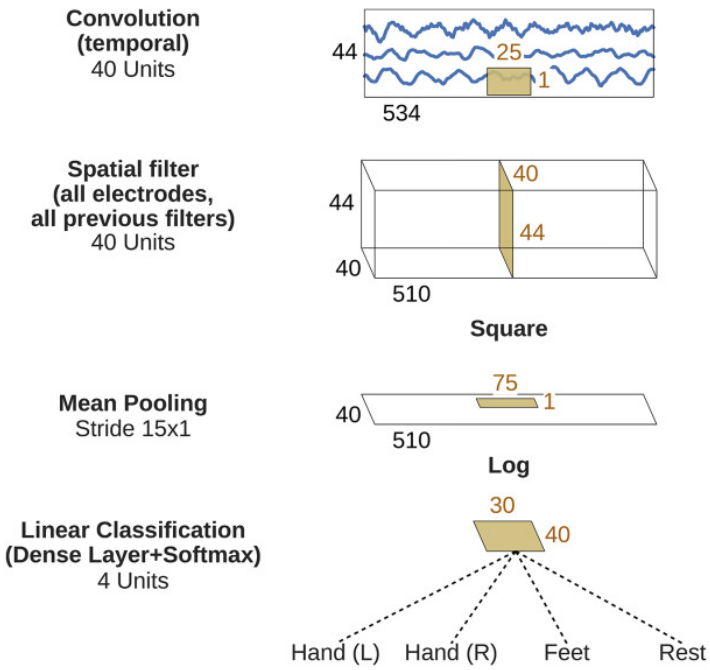

Figure C.1: **ShallowFBCSPNet architecture** [6].

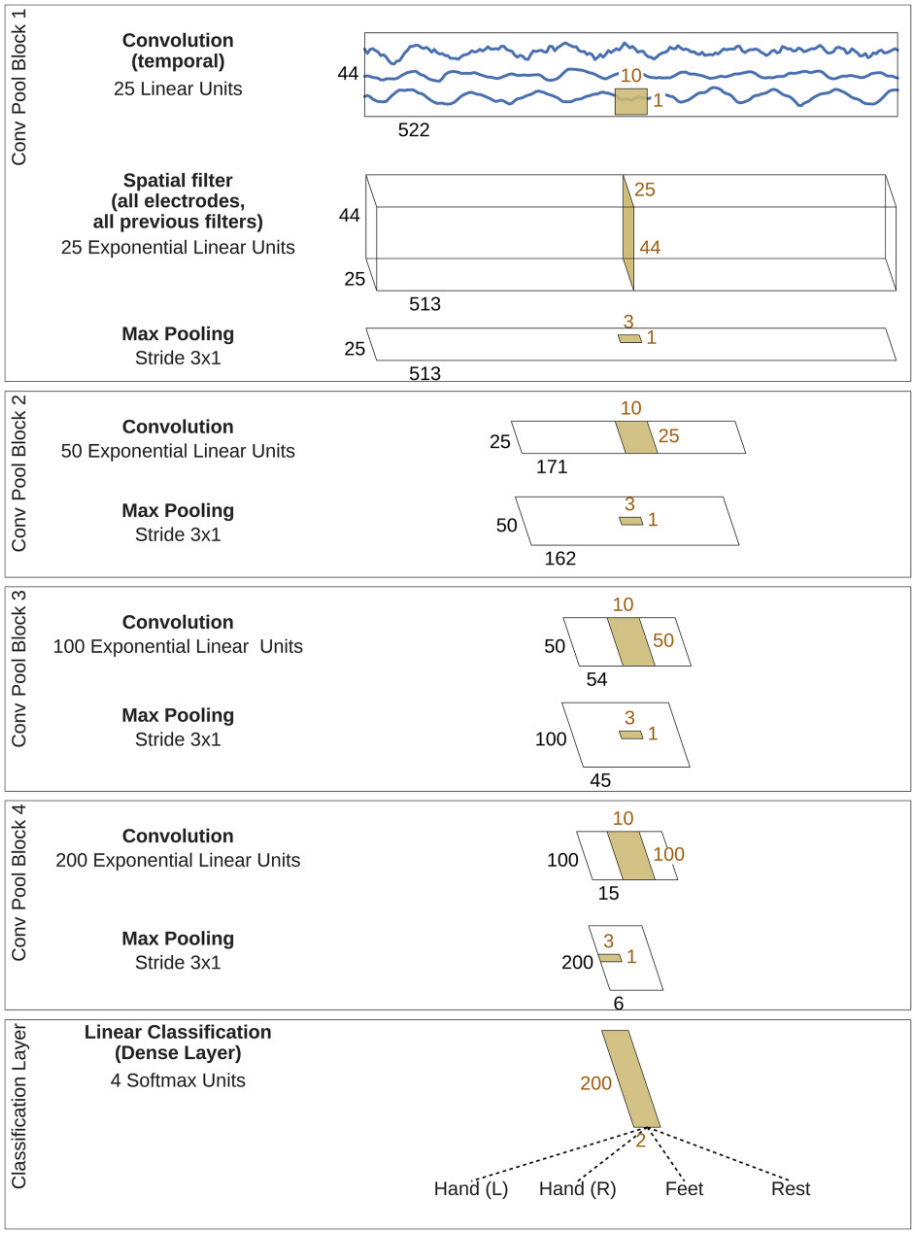

Figure C.2: **Deep4Net architecture** [6].

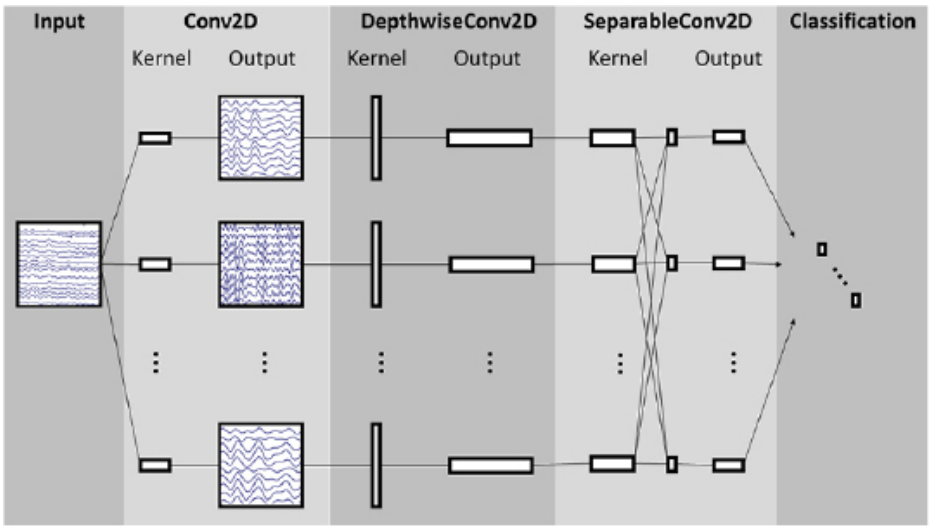

Figure C.3: **EEGNet architecture** [31].