ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für Integrierte Systeme
Integrated Systems Laboratory

DEPARTMENT OF
INFORMATION TECHNOLOGY AND ELECTRICAL ENGINEERING

Fall Semester 2023

Self-supervised EEG Representation for Motor-Imagery Brain-Computer Interfaces

Semester Project

Sebastian Jäger
jaegeseb@student.ethz.ch

14.12.2023

Advisors: Thorir Mar Ingolfsson, thoriri@iis.ee.ethz.ch
Dr. Xiaying Wang, xiaywang@iis.ee.ethz.ch
Dr. Yawei Li, yawei.li@vision.ee.ethz.ch

Professor: Prof. Dr. L. Benini, lbenini@iis.ee.ethz.ch

Abstract

Transformer-based foundation models have yielded state of the art results in many areas using electroencephalogram (EEG) data, such as seizure detection. In this work, we attempt to bring this success to the setting of EEG Motor-Imagery (MI) tasks for Brain-Computer Interfaces (BCI). To this end, we create a variety of transformer-based foundation models starting from BrainBERT [1]. They feature the Short-Time Fourier Transform (STFT) and convoultional embeddings as well as classical transformers and vision transformers. During pretraining, we reconstruct the spectrum of the input data, or the input data itself, from masked versions. We examine the architectures in pretraining and the quality of the extracted features in downstream classification, where we use a linear classifier. There, the convolutional embedding, as it was used in EEGNet [2], helps increase performance. At the same time, we seem to struggle to benefit from EEG foundation model approaches and the transformer. In a binary downstream classification task, we are unable to reach the state of the art 84% accuracy of EEGNet, but achieve up to 68% accuracy. This puts the performance significantly above the random guessing accuracy of 50%. Our analysis indicates the need for compact models and large pretraining datasets. Finally, we illustrate possible ways of improving foundation models for EEG-based MI BCIs further in the future.

Acknowledgments

I want to thank my advisors, Thorir Mar Ingolfsson, Dr. Xiaying Wang and Dr. Yawei Li for their guidance and the many interesting and helpful discussions.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Self-supervised EEG Representation for Motor-Imagery Brain-Computer Interfaces

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Jäger	Sebastian

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, 14.12.2023	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

.

Contents

Chapter 1

Introduction

Motor-Imagery (MI) Brain-Computer Interfaces (BCI) process brain signals to predict actions imagined by the subject. To this end, classical approaches use a form of feature extraction followed by a classification method. Current state of the art approaches use a single network to address both challenges. This results in networks specific to the task, e.g. distinguishing imagining the actions *opening the left fist* and *closing the left fist*. Thus, such a model needs to be retrained to solve a different task.

In contrast to this, developments in speech and language processing demonstrate the strength of large foundation models, which separately perform feature extraction. These models are trained by optimizing them on a surrogate pretraining task, using unlabeled data. After pretraining, they serve as general feature extractors. The resulting features are input into a classification network to solve the task at hand. This classification network is trained on the task, while the foundation model is frozen after pretraining. If a foundation model can produce expressive features, a small classification network trained on few labeled data is sufficient to make accurate predictions. When considering a different task, the foundation model can be reused, and only the prediction network is retrained.

1.1 Focus of this Work

In this work, we attempt to bring the advantages of foundation models to MI BCI classification tasks based on electroencephalogram (EEG) data. There, our goal is to predict the action imagined by a subject from EEG recordings. To this end, we adapt and combine the existing networks BrainBERT [1], EEGNet [2] and TSD [3] to create transformer-based foundation models. This leads to the exploration of architectures differing in the method of feature extraction and channel aggregation. For assessment, they are trained and evaluated on the PhysioNET dataset [4].

1.2 Organization of this Work

In Chapter 2, we introduce EEG and the general ideas of foundation models and transformers. Concrete EEG foundation models and classification architectures are introduced in Chapter 3 in detail. Chapter 4 describes the architectures we constructed and introduces the pretrainig and downstream classification schemes. Implementation details, results and analyses of our approach are found in Chapter 5.

Chapter 2

Background

2.1 Electroencephalogram

In EEG, brain signals are recorded using electrodes that monitor the electrical activity of the brain. The individual electrodes, or channels, measure the electrical potential. In neuroscience and clinical research, EEG is employed for detection of seizures and brain injuries as well as sleep monitoring [5]. To help with such analyses, the EEG data is often combined with video recordings of the subjects. Furthermore, EEG is used in BCIs, where the action imagined by a subject is predicted. This allows to act based on this prediction, e.g. steer a wheelchair or move a prosthetic. Typically, the electrodes are placed on the subjects head. In contrast to this, intracranial EEG (iEEG) uses electrodes that are surgically inserted into the subjects skull and placed on the surface of the brain. This reduces noise and allows to monitor the signals in a more localized way, allowing to more accurately detect activity in a specific area of the brain.

2.2 Foundation Models

A variety of different approaches to apply foundation models on EEG data have been made recently. They are categorised as Auto-Encoders (AEs). During pretraining, a prediction head is placed after the feature extraction network. The combined network is optimized to reconstruct the inputs to the feature extractor network at the output of the prediction head. Often, portions of the inputs are masked and the the predicted reconstruction is optimized to be close to the unmasked input. The output of the feature extraction network, termed the latent representation, is meant to encode meaningful information about the input. It is meant to not only be helpful in reconstructing the input, but also useful as a general feature set for solving other tasks. To this end, after pretraining, the prediction head is no longer used, but a classification network is applied after the feature extractor. In what we term downstream training, the classification network is trained to solve the task at hand. During this phase, the feature extractors

weights are either frozen or fine-tuned jointly with the classification network. An advantage of foundation models is that the feature extractor network can be reused for any task based on the same type of input. In the literature, the features extracted by EEG foundation models are used in a variety of different task. These include MI classification [6], emotion prediction [7] and sleep stage classification [8].

2.3 Transformer and Vision Transformer

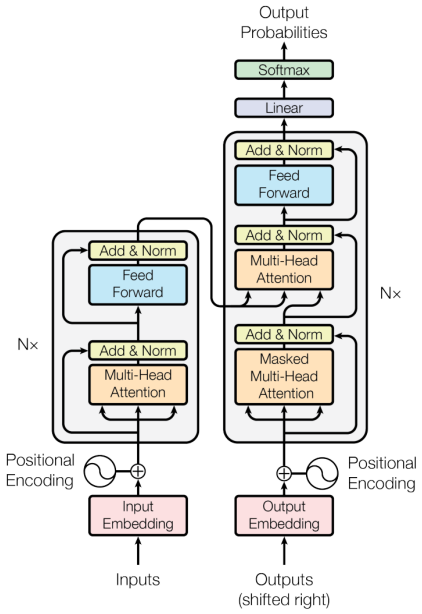

Figure 2.1: Transformer network, figure from [9]

The transformer is a network component popular in deep learning, especially in natural language processing (NLP). It considers a series of input vectors, named tokens, and processes them in a multi-head self-attention mechanism. The detailed model is introduced in [9] and shown in Fig. 2.1. We will consider transformers without the embedding blocks in Fig. 2.1. A key characteristic is the use of attention layers. There, relations between the input tokens, e.g. correspondences between words in NLP, are estimated. Multiple such filters can be used in parallel, referred to as multiple heads. The transformer is built from two parts, the encoder on the left and the decoder on the right side of Fig. 2.1. The encoder is used to find a representation expressing the relations between tokens in the input. The decoder processes this representation in combination with the previous decoder outputs to predict the next output token.

In this work, we consider transformer stacks, i.e. multiple layers of multi-head transformers. We refer to these simply as transformers. They are characterized by the

number of layers n, the number of attention heads per layer h and the hidden layer dimension.

For computer vision applications, [10] proposes splitting an image into patches, and creating tokens from each patch individually. This way of applying a transformer to an image is commonly referred to as a vision transformer.

Chapter 3

Related Work

3.1 Foundation Models for EEG

Wav2Vec [11] is an AE speech recognition network, that is pretrained to predict future speech signal given the current input. The architecture features a convolution feature extraction network and a Multilayer Perceptron (MLP) for the classifier. The similarity in speech and EEG data both being time series with large amounts of noise, lead to the development of EEG2Vec [12, 13]. There, a similar approach is taken to transfer the idea to the realm of EEG. The downstream task considered was sentiment detection (positive, neural, or negative), given the EEG input. In contrast to this, Speech2EEG [6] directly takes an encoder network from a speech recognition model, pretrained on a speech dataset. This network is applied on EEG data, separately on each channel. The latent representations of the channels are aggregated in a convolutional network and thereafter input into an MLP for classification. During downstream training, the encoder is fine-tuned and the network is optimized to solve motor imagery tasks. A core idea of this approach is that the model can benefit from pretraining on a very large speech dataset, and is not limited to the size of available EEG data sets. [14] further makes use of the similarities between EEG and speech to build a model for translation of neural activity to speech.

In BENDR [8], a convolutional encoder is used, while the classifier is built as a transformer. The transformer is used during pretraining to reconstruct the full embeddings produced by the feature extractor from masked versions of thereof. The pretraining loss is augmented by a contrastive loss term, ensuring that non-corresponding inputs lead to largely different latent representations. TS-MoCo [7] uses a linear layer and a transformer for feature extraction, and a second linear layer for classification. It employs momentum contrast to increase the robustness of the model to augmentations of the input data. In downstream classification, the task of emotion prediction from EEG is considered. The authors of [15] attempt to solve the task of classifying subjects as obese or not obese based on EEG data recorded from them. It applies a convolutional feature extractor and a MLP for classification. In MIN2NET [16], a convolutional feature

extraction is applied before a classifier composed of convolutional layers and a linear layer. The optimized loss features a contrastive loss term and the downstream tasks are motor imagery classification.

There are also approaches to combine AEs for EEG with graph-based embeddings [17], and Recurrent Neural Networks (RNN) [18]. Furthermore, [19] considers computing the optimal latent representation size for EEG based AEs.

The AE model on which architectures considered in this work are based, BrainBERT [1] is presented in detail in the following.

3.1.1 BrainBERT

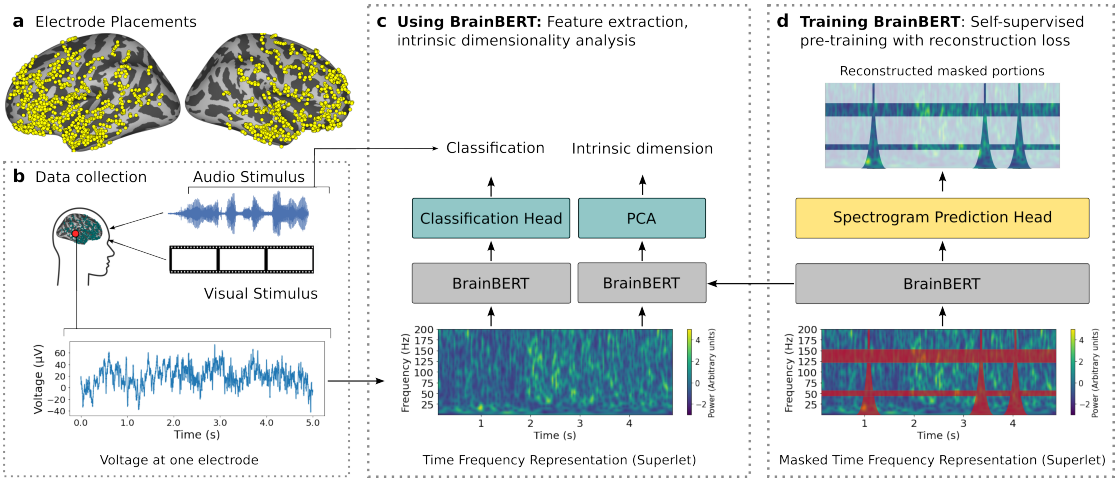

Figure 3.1: Architecture of BrainBERT, figure from [1]

BrainBERT [1] is a foundation model applied to intracranial EEG signals. The general setup is shown in Fig. 3.1. It uses a transformer achitecture for feature extraction and is pretrained to reconstruct masked spectra of the input signals. In downstream classification, a single layer MLP is used.

The iEEG data is collected while showing the subjects movies, i.e. audio and visual stimuli. These stimuli are used to define the downstream tasks. The data is recorded at 2'000 Hz. The model takes a single channel as input. As a preprocessing step, the Short-Time Fourier Transform (STFT) of the input signal is computed. For this, the signal is divided into segments of 400 data points, overlapping by 350 data points. Then, the Discrete Fourier Transform (DFT) of each segment is computed. The intensity values of 40 frequencies evenly spaced between 0 and 200 Hz are chosen. This yields a 2D time frequency representation of the signal. Since low frequencies contain most of the energy, but not necessarily most information, the values of the spectrum are reweighted. To this end, Mean-Standard Deviation Normalization, also known as z-Scoring, is applied along the time dimension. Up to this point, only fixed preprocessing and no learnable transformation is applied.

During pretraining, certain frequency bands and time slots are chosen at random to be masked. In a linear input encoding layer, the 40 frequency values per time slot are encoded into 768 values. Then a transformer stack containing 6 layers, 12 attention heads and hidden layer size 768 is applied. The output of the final layer are the extracted features.

During pretraining a prediction head MLP containing containing 2 layers, hidden dimension 768 and output dimension 40, is applied. Thus, its output is of the same size as the input spectrum. The L1-loss of the masked portions is augmented by weighting the values with large absolute value higher. The model is optimized to minimize this loss using the LAMB optimizer.

In downstream classification, four different tasks are considered: Sentence onset detection, speech vs. non-speech classification, volume classification and classification estimation, all with respect to the video shown to the subjects. A single linear layer is used to classify the features extracted by BrainBERT. For each task and each channel, an individual classifier is trained. The information provided by the different channels is not explicitly aggregated. Instead, the 10 best classifiers per task are selected and thus only the 10 most accurate classifiers are considered.

3.2 EEG Classification Models

3.2.1 Transformers for Seizure Detection

TSD [3] is a seizure detection model for EEG signals based on STFT. Its main differences from BrainBERT are the use of extracranial EEG data, the use of a vision transformer, it not being a foundation model but a direct classifier and that the signals from the different channels are aggregated early on in the network. This means that, instead of channel-specific features, a single set of features is obtained for the entire multi-channel input. Furthermore, the application of a vision transformer allows to work along temporal and channel-wise dimensions, rather than processing features across time only.

The STFT of all input channels is computed. Then, as illustrated in Fig. 3.2, the resulting 2D spectrum is divided into patches and each patch is vectorized and projected to a vector of size 16 using a linear layer. This is input into a transformer stack of length 4, using 4 attention heads in each layer and hidden dimension 16. The transformer is augmented with a classification token. The model is trained to classify the input as seizure or no seizure in this token at the output of the transformer stack.

3.2.2 EEGNet

EEGNet [2] is a convolutional EEG classification network for BCI tasks. The architecture, illustrated in Fig. 3.3, is compromised of two blocks. In block 1, the multi-channel EEG input is convolved in time dimension, then in channel dimension. Multiple convolution kernels are learned, leading to multiple representations. Then, average pooling in time dimension is applied. In block 2, a separable convolution along time dimension and the different representations of the different kernels is applied. Finally, a linear classification

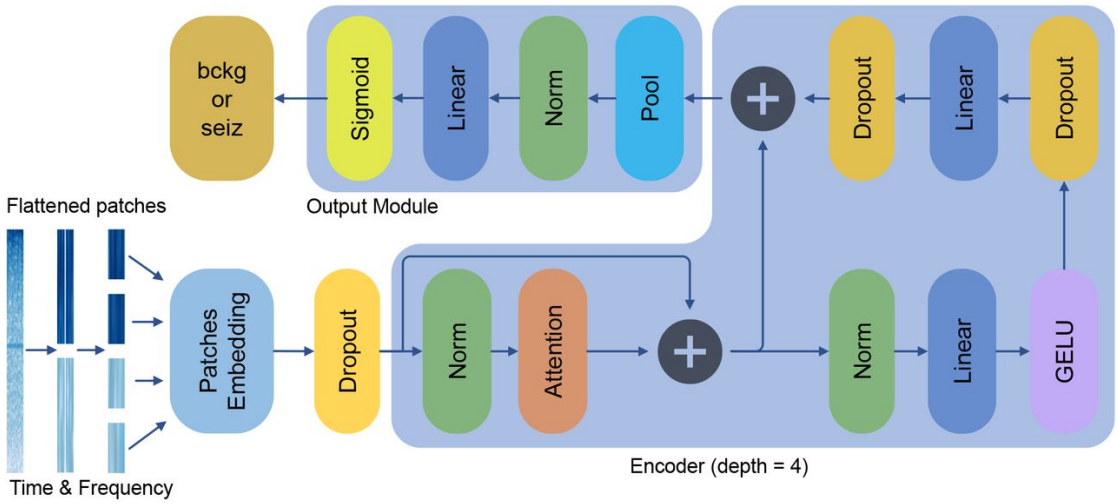

Figure 3.2: Architecture of TSD, figure from [3]

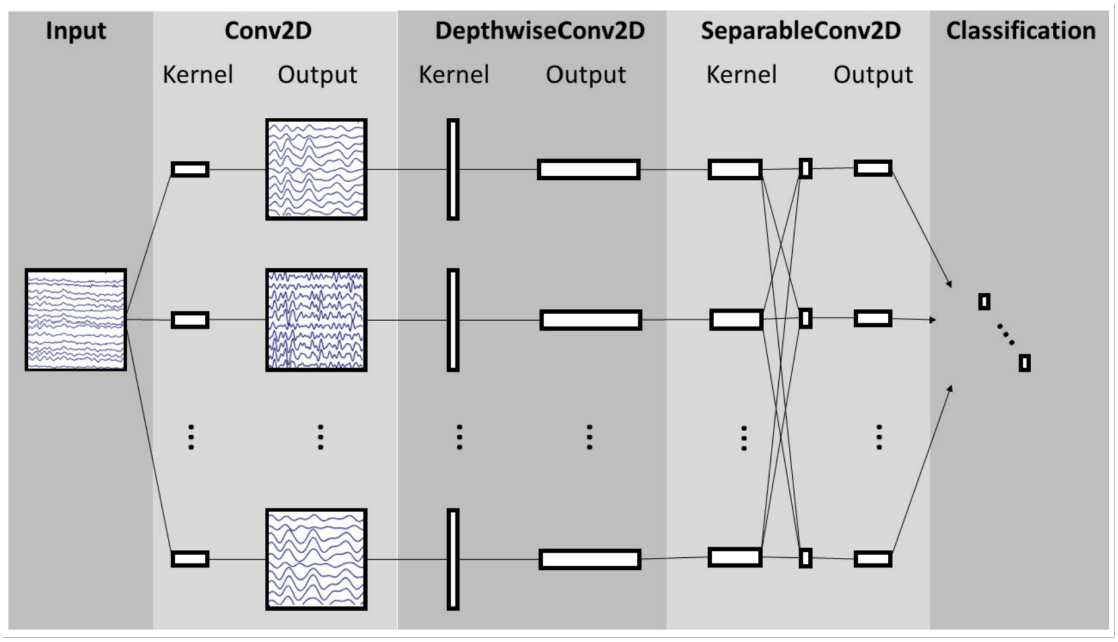

Figure 3.3: Architecture of EEGNet, figure from [2]

layer combines the outputs of block 2 into into a vector of the same size as the number of classes. A softmax activation is used, such that this output can be interpreted as class probabilities.

EEGNet is designed to be compact, employing a small number of trainable parameters. It is considered state of the art in MI BCI classification tasks. Recently, [20] has combined EEGNet with a transformer for the task of seizure detection. We will consider the original

implementation as a reference for the results we achieve in this work on MI BCI tasks.

3.2.3 EEGformer

Another EEG classification model based on the transformer architecture is EEGformer [21]. It is a compact transformer model for seizure detection, designed to run in real-time on small processing units. A second model by the same name is introduced in [22]. It employs a convolutional encoder to produce 3D data tensors, followed by three transformers. The transformers are set up to process data across different encoding dimensions, such that all three are covered. As downstream tasks, emotion prediction and depression classification are considered.

Chapter 4

Method

In this chapter, the different architectures we use are described. In Section 4.1, we describe how we adapt the architecture from BrainBERT [1] to the use in our extracranial EEG MI classification setting. Section 4.2 shows how we changed the STFT to a learnable convolutional embedding, as it is used in block 1 of EEGNet [2]. Finally, architectures where we apply a vision transformer can be found in Section 4.3. There, we consider both the STFT and convolutional embedding.

We consider inputs of 1280 time steps, which corresponds to 8s in the PhysioNET dataset, which we use for pretraining and downstream classifier training. The dataset and the input formation procedure are described in Section 5.1. The number of weights in the networks is shown in Table 5.6.

4.1 BrainBERT

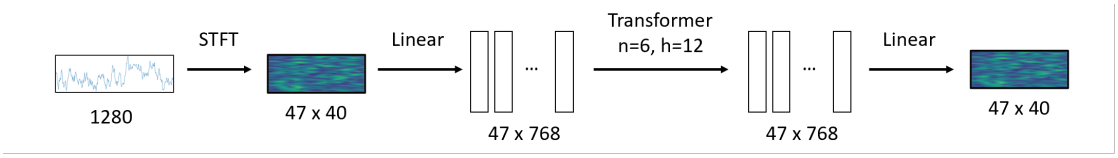

Figure 4.1: Adapted BrainBERT architecture.

Fig. 4.1 shows the architecture of BrainBERT [1], see also Section 3.1.1, adapted to our setup. We input a single channel into the STFT, where we use time slots of 213 data points per DFT calculation, which overlap by 190 data points. We obtain 47 time slots, and calculate the intensity values for 40 frequencies, spaced evenly between 0 and 60 Hz. This change of frequency range is made since for extracranial EEG, typically this lower frequency range is considered. The reason for this is that higher frequencies are mostly associated with noise. Afterward, as in the original setup, We use a linear layer to obtain tokens of size 768, apply a transformer with $n = 6$ layers, $h = 12$ attention heads

and hidden layer dimension 768. During pretraining, the outputs of the last layer are reduced to size 40 with a linear layer, such that the input spectrum can be reconstructed. We pretrain the model on input data from all 64 available channels of the PhsyioNET dataset. During downstream classification, we use the outputs of the last transformer layer as the extracted features. Then, for each channel, we train an individual classifier.

4.2 Architectures using Convolutional Embedding

In this section, we describe the approach of using the BrainBERT [1] architecture, but replacing the STFT with a learned convolutional embedding. The reasoning behind this is that convolutions allow us to aggregate the information from the different input channels already in the feature extraction network, before the transformer. This means that with this embedding, we can use all channels simultaneously as input and make use of the recorded spatial information. Moreover, the use of convolutions for EEG signal processing has led to success in the EEGNet [2] BCI classification network, described in Section 3.2.2. We use the convolutional embedding of block 1 of EEGNet. Additionally, we also test architectures where we adjust the embedding by introducing temporal downsampling, through pooling or through the use of stride in the temporal convolution. We use the same transformer stack dimensions as in the BrainBERT architecture: $n = 6$ layers, $h = 12$ attention heads and hidden layer dimension 768.

During pretraining, the architectures reconstruct the masked raw EEG input signal. In downstream classification, we again use the outputs of the last transformer layer as the extracted features, which we input into a classifier.

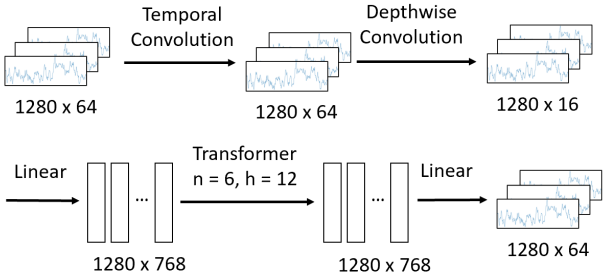

Figure 4.2: Architecture with convolutional embedding, no downsampling.

Fig. 4.2 shows the architecture where we apply the convolutional embedding of EEGNet directly, without adding a way of downsampling. We first apply same convolution with kernel size 64 in temporal dimension, keeping the number of samples in time unchanged. We use 8 such filters, defined by different kernels. Afterward, batch normalization is applied before the depth-wise valid convolution in channel dimension. The kernel size is the same as the number of channels, so 64 in our case, yielding output size 1 in channel dimension. Since we have 2 depth-wise convolution kernels per temporal convolution filter, we end up with a total of 16 embeddings per time point. A batch

normalization, a ReLU activation and a dropout layer with dropout probability 0.5 are used. Contrary to the original EEGNet implementation, we do not use the average pooling layer in this architecture. This allows us to only consider the effect of the convolutions at first, while we introduce a version with pooling in the following. A linear layer brings the dimension per time point to 768, before the same transformer stack as in the adjusted BrainBERT architecture is used. Contrary to before, the linear reconstruction layer appended to the transformer now has output dimension equal to the number of channels, to reconstruct the raw input signal.

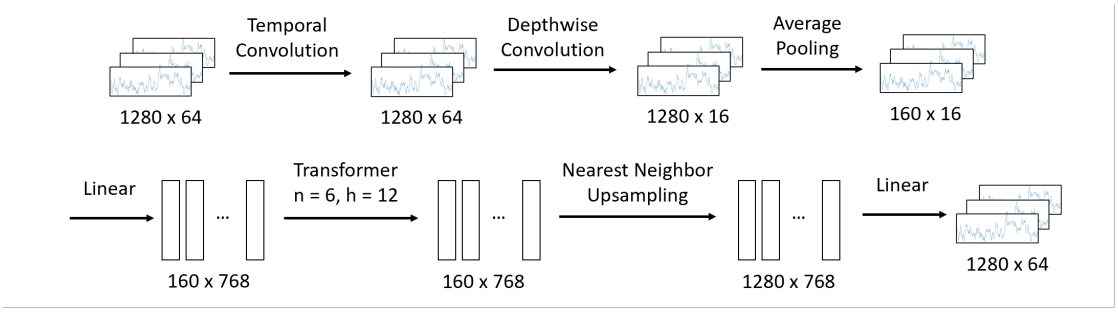

Figure 4.3: Architecture with convolutional embedding, pooling.

In Fig. 4.3, we adjusted the architecture in Fig. 4.2 by introducing an average pooling layer directly after the depth-wise convolution. It has kernel size 8, meaning that 8 consecutive values in time are replaced by their average. This reduces the number of tokens input into the transformer by a factor of 8, from 1280 to 160. The output of the transformer is upsampled back to 1280 data points by means of nearest neighbor upsampling before being input into the linear reconstruction layer. In this case, each sample is simply repeated 8 times. The nearest neighbor upsampling is used only during pretraining for input reconstruction, not during downstream classification.

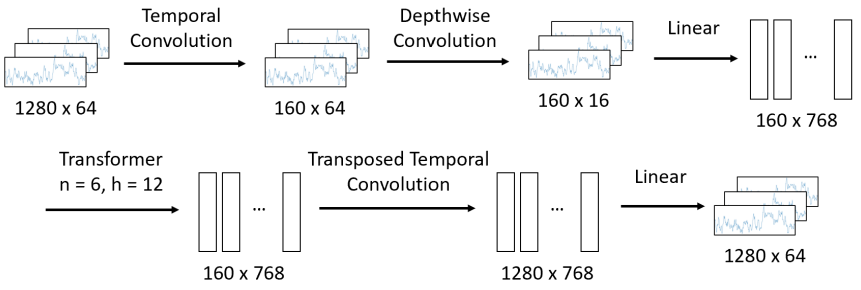

Figure 4.4: Architecture with convolutional embedding, stride.

Fig. 4.4 shows a second way of introducing temporal downsampling to the architecture of Fig. 4.2. Instead of using a pooling layer, we adjust the temporal convolution by setting the stride to 8. This reduces the number of samples in time by a factor of 8, thus the downsampling ratio is the same as for the architecure with pooling. During pretraining,

we increase the number of samples in time back to 1280 after the transformer. To this end, we apply transposed convolution with stride 8.

4.3 Architectures using Vision Transformers

In this section, we describe architectures using a vision transformer. One is an implementation of TSD [3], which uses STFT embedding, the other replaces the STFT with temporal convolution.

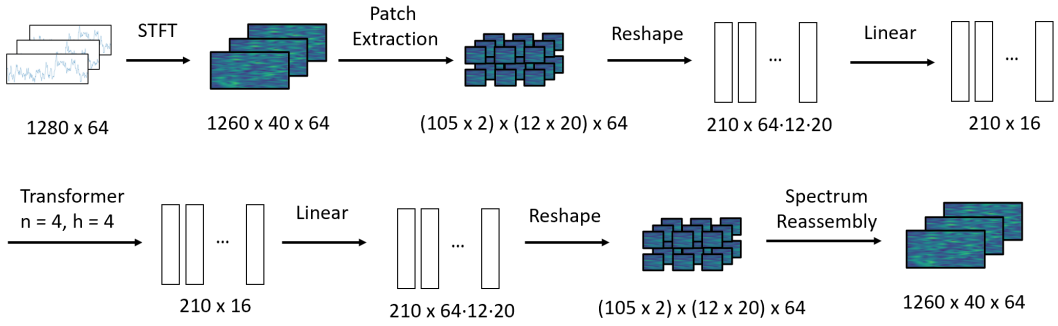

Figure 4.5: Vision Transformer architecture with STFT embedding.

In the architecture shown in Fig. 4.5, we apply the STFT using time slots of 213 data points, overlapping by 212 data points. We discard the part of the spectrum corresponding to the 10 first and the 10 last time slots. The reason for this is that in these time slots, the STFT shows effects that stem from a lack of data near the edge of the input. This gives 1260 time slots, for which we calculate intensities for frequencies evenly distributed from 0 to 60 Hz. This procedure is executed for each channel. We apply the vision transformer regarding the spectrum as a 2D image, with the electrode channels taking the role of the RGB channels in computer vision. We split the spectrum into patches of size 12 in time dimension and size 20 in frequency dimension. Each patch is reshaped to a vector and a linear layer with output dimension 16 is applied. The transformer stack has the same dimensions as in TSD [3]: $n = 4$ layers with $h = 4$ attention heads each and hidden layer size 16 is used. The outputs of the last transformer layer are the extracted features. During pretraining, we use a linear layer, followed by reshaping, to obtain a reconstruction of the spectrum patches. From these, the input spectrum is reassembled.

The architecture of Fig. 4.6 replaces the STFT with a convolutional embedding. Same convolution in the time direct is applied using kernel size 64. We employ 8 different such kernels, and interpret the result as a 2D image in time and filter dimension, with the channels taking the role of the RGB channels in computer vision. It is split into patches of size 16 in time dimension and size 4 in filter dimension, which are reshaped to vectors. A linear layer reduces their size down to 16, before the same transformer stack as in TSD[3] with $n = 4$, $h = 4$ and hidden layer dimension 16 is applied. The outputs of the last transformer layer are the extracted features. During pretraining, they are processed

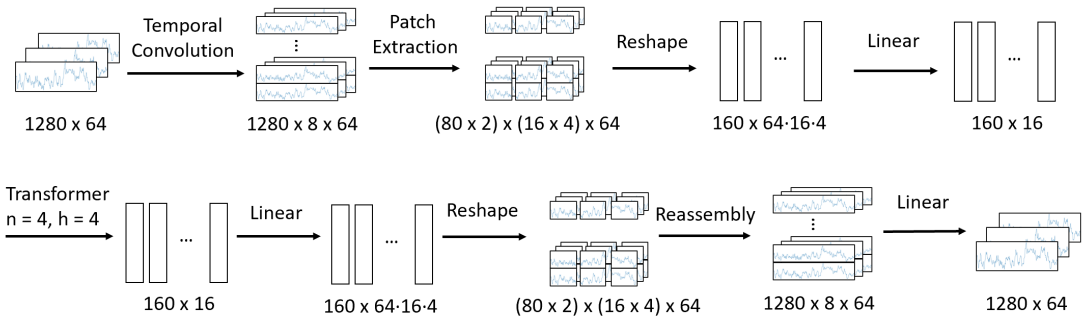

Figure 4.6: Vision Transformer architecture with convolutional embedding.

by a linear layer and reshaped to reconstruct the patches, from which a layer of the same dimensions as before patch extraction is assembled. Finally, a linear layer is used to reduce the number of features per temporal data point and channel from 8 to 1, giving an output of the dimension of the raw input signal.

4.4 Partial Architectures

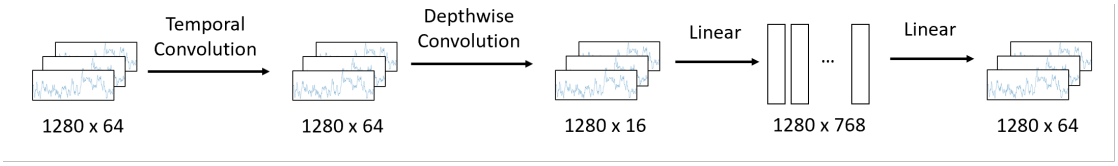

Figure 4.7: Architecture using only convolutional embedding.

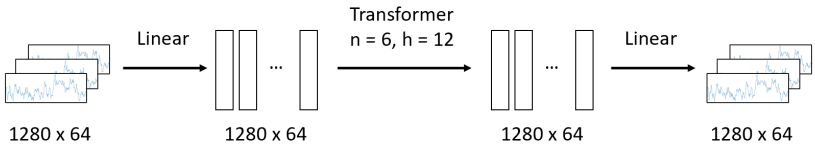

Figure 4.8: Architecture using only transformer.

To analyze the contribution of the convolutional embedding and the transformer toward the extraction of useful features, feature extractors using these components individually are created. Concretely, we start from the model using convolutional embedding and no downsampling, see Fig. 4.2. Using only the convolutional embedder as feature and leaving out the transformer, we obtain the architecture shown in Fig. 4.7. On the other hand, only using a transformer and no convolutional embedding yields the architecture in Fig. 4.8.

4.5 Pretraining

4.5.1 Masking Procedure

During pretraining, we apply a masking procedure. If STFT is used, masking is applied after the STFT, if a convolutional embedding is used, masking is applied on the raw input singal. In this way the signal is masked directly before the first embedding transformation using learned parameters. This prevents the network from learning an embedding that is simply easy to reconstruct, and potentially not expressive. For all architectures, we employ the masking strategy of BrainBERT [1]. We randomly select time intervals to be masked. Each interval has a random width in the range $[\text{step}_t^{min}, \text{step}_t^{max}]$. Similarly, random frequency or channel intervals are selected, depending on whether the STFT or the raw input is masked. These intervals are chosen with random widths in $[\text{step}_{fc}^{min}, \text{step}_{fc}^{max}]$. Each time point, frequency or channel is the start of an interval with probability p_{mask}. The interval width is chosen uniformly in the ranges described above. Then, the values in the interval are left unchanged with probability p_{ID}, replaced by a random interval from the input with probability $p_{replace}$ or set to 0 otherwise.

4.5.2 Pretraining Loss

Let X_{ij} denote the input before masking and $\hat{X}_{ij}$ the reconstruction, where i indexes time and j indexes the channel. Using the set M to denote all masked locations, the loss for all architectures where we reconstruct the raw input is the L1-loss of all masked locations

$$\mathcal{L} = \frac{1}{|M|} \sum_{(i,j) \in M} |X_{ij} - \hat{X}_{ij}|. \tag{4.1}$$

Using the notation Y_{ij} for the spectrum before masking, $\hat{Y}_{ij}$ for the reconstructed spectrum, where i denotes the time index and j the frequency index, and letting M be the set of masked positions, the loss for all architecture where the STFT is masked is

$$\mathcal{L} = \frac{1}{|M|} \sum_{(i,j) \in M} |Y_{ij} - \hat{Y}_{ij}| + \mathcal{L}_{ca}. \tag{4.2}$$

This is the augmentation with the content aware loss

$$\mathcal{L}_{ca} = \alpha \frac{1}{|\{(i,j) : Y_{ij} > \gamma\}|} \sum_{(i,j):(i,j) \in M, Y_{ij} > \gamma} |Y_{ij} - \hat{Y}_{ij}| \tag{4.3}$$

proposed in [1]. The introduction of the second loss term is meant to encourage accurate reconstruction of signals where neural processes are likely to be happening.

4.6 Downstream Classification

In downstream classification, we use the features extracted by the pretrained feature extractor to solve the binary classification task introduced in Section 5.1. The features

are extracted as the output of the last transformer layers for all models. We employ a linear layer with input size equal to the dimension of the output tokens and output size 2. Then we apply the Softmax activation function, such that the outputs can be interpreted as class probabilities. Such a set of probabilities is obtained for each output token. We take the mean of these to obtain the predicted probabilities of classifying the input as left fist or right fist. In the downstream training procedure we minimize the Binary Cross-Entropy loss.

Chapter 5

Results

5.1 PhysioNET Dataset

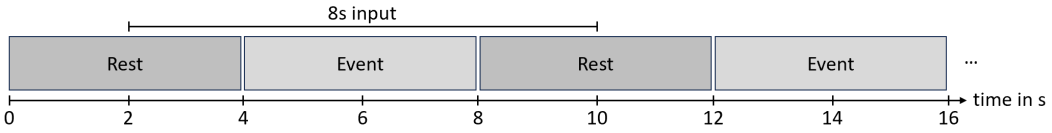

Figure 5.1: Formation of input segments from the PhysioNET dataset.

In this work, we use the PhysioNET dataset [4] for both pretraining and downstream training. This dataset contains extracranial EEG data from 109 subjects, recorded at 160 Hz with 64 channels. We discard 4 subjects due to variability in the number of runs, as in [23]. Different runs with varying tasks are performed with each subject. We consider a binary MI classification task. Here, the subjects imagine opening and closing either their left or their right fist, depending on an indication shown to them on a screen. There are three two-minute runs recorded for this task per subject. Each run alternates between 4 seconds of rest and 4 seconds corresponding to an event, i.e. left or right fist. For all of our architectures, we consider 8 second segments as input, which corresponds to 1280 data points in time per channel. The inputs are formed from the 4 seconds of the event, the 2 seconds of rest directly before and the 2 seconds of rest directly after the event, see Fig. 5.1.

5.2 Implementation Details

We split the the PhysioNET dataset into a train and a test split. The train split is formed by the first 85 subjects, the test split includes the remaining 21. From the training split, 1% of the data is randomly chosen and reserved for validation during pretraining. We pretrain the model on the remaining portion of the train split. In our experiments, we set

Architecture	$\mathcal{L}^{train}$	$\mathcal{L}^{val}$	$\mathcal{L}^{val}_{ca}$
BrainBERT	**0.4078**	**0.3503**	**0.2637**
STFT + Vision Transformer	2.8413	2.785	1.9631

Table 5.1: Pretraining losses of architectures with STFT embedding. BrainBERT is reconstructing a single channel, STFT + Vision Transformer is reconstructing all channels at once. For the latter, the loss is the average across all channels.

the ranges in which the masking interval lengths are chosen to $[\text{step}_t^{min}, \text{step}_t^{max}] = [1,5]$ and $[\text{step}_{fc}^{min}, \text{step}_{fc}^{max}] = [1,2]$. The masking probabilities are $p_{mask} = 0.05$, $p_{ID} = 0.1$, and $p_{replace} = 0.1$. The adjustable parameters of the pretraining loss when applying STFT are $\alpha = 2$ and $\gamma = 1$. We use the LAMB optimizer with learning rate 10^{-4} and unless stated otherwise, we perform 200′000 updates.

The downstream classifier is trained on the first 82 subjects of the same train split that was used in pretraining. The remaining three subjects are kept as the validation set. This is data seen by the network during pretraining, but not during downstream classifier training. The test set remains unchanged compared to pretraining, meaning that no part of the network was ever trained on it. We use the AdamW optimizer with learning rate 10^{-4} and perform 1′000 training updates. During downstream training, we keep the weights of the pretrained feature extractor unchanged. When experimenting with simultaneously training the downstream classifier and finetuning the feature extraction network, we could not improve results, contrary to [1].

5.3 Pretraining

The training and validation losses of the architectures with STFT embedding are shown in Table 5.1. The BrainBERT architecture manages to reduce the loss to a much smaller value than the architecture using the vision transformer. While the architectures are trained on the same dataset, the BrainBERT architecture takes a single channel as input, and not all channels at once. Thus, it cannot use channel information to solve the pretraining task. On the other hand, this means that the number of training examples is increased by a factor of 64, as we have 64 channels in the PhysioNET dataset. The loss computed for the architecture using STFT and a vision transformer is the average loss across all channels.

The increased number of training samples possibly contributes to the improved results in the pretraining task. But the large differences in architectures also have to be considered: The hidden layer dimension in the vision transformer is significantly smaller (16 compared to 768 in BrainBERT). Such a bottleneck increases difficulty of the pretrainng task, but can potentially lead to more expressive extracted features. Afterall, the goal is not to solve the pretraining task well, but to extract features helpful in downstream classification.

Table 5.2 shows the pretraining losses for the architectures with convolutional em-

Architecture	$\mathcal{L}^{train}$	$\mathcal{L}^{val}$
Conv., no Downsampling	**24.4167**	**26.8212**
Conv., Pooling	25.9124	27.9238
Conv., Strided Convolution	25.5206	31.0337
Conv. + Vision Transformer	27.2257	28.8223

Table 5.2: Pretraining losses of architectures with convolutional embedding.

bedding. We obtain the smallest losses for the convolutional architecture without downsampling. Looking at the proposed downsampling methods, the use of strided convolution yields a smaller training loss, but in validation, it is outperformed by the simpler pooling approach. Finally, the application of a vision trainsformer gives the largest training loss.

The architecture without downsampling likely benefits from not having to reconstruct from a decreased number of temporal features. The strided convolution approach is expected to converge to a lower training loss than the pooling approach, as it generalizes the latter: The srided convolution kernel can implement the average pooling operation by converging to equal weights in all kernel entries. In validation however, the strided convolution does not generalize as well as the pooling. As before, the vision transformer again poses a much smaller bottleneck since it has a much smaller hidden layer dimension as the other architectures in this table (16 compared to 768), increasing the difficulty of the pretraining reconstruction task.

Some example reconstructions for the different convolutional architectures are shown in Fig. 5.2. The example input is taken from the PhysioNET dataset, subject 1, run 5, the first segment. We visualize the middle 3 seconds of channel 6 in the figure. Here, no masking was applied on the input. Thus, the plots are not representative of the pretraining loss, but illustrate the different architectures' approaches to reconstruction. The top plot shows reconstruction for the architecture without downsampling. Since no temporal downsampling is used, this model is not forced to compress temporal information like the others, and therefore able to more accurately reconstruct high frequency components of the input. This enables the most accurate reconstruction out of the three architectures. The pooling architecture uses nearest neighbor upsampling in the reconstruction network, yielding the piece-wise constant reconstruction of the middle plot. Finally, the bottom plot shows the use of strided convolution. The reconstruction by transposed strided convolution gives a smoothed signal compared to the input. Both downsampling approaches yield reconstructions that appear to follow the running average of the input signal. On the other hand, they fail to recreate high-frequency components due to the forced temporal compression of the input information.

Table 5.3 shows the pretraining losses for the partial architectures. The one using convolutional embedding without a transformer gives a lager training loss than the architectures with transformer in Table 5.2, but a lower validation loss. The architecture using only the transformer gives worse training and validation losses than the architec-

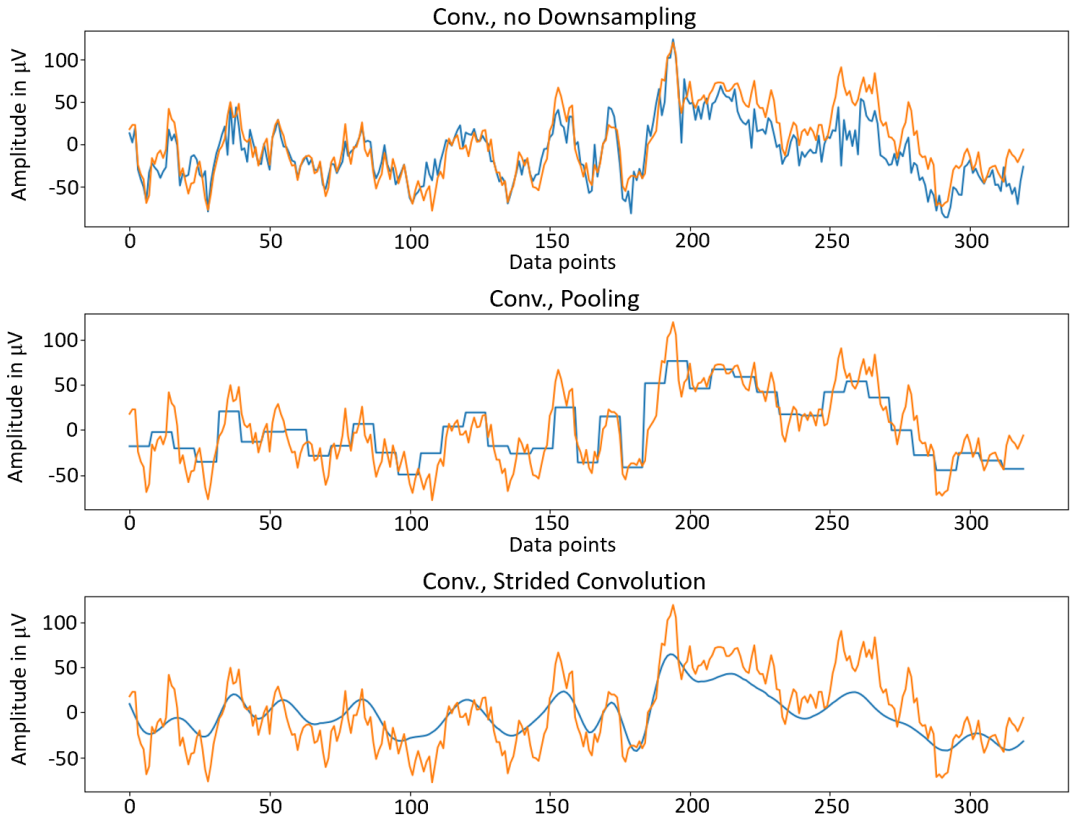

Figure 5.2: Example input (red) and reconstruction (blue) when using the convolutional architecture without downsampling (top), with pooling (middle) or with strided convolution (bottom).

tures in Table 5.2 when it is trained for the same number of updates (200'000). However, increasing the number of training updates decreases the loss significantly, giving a comparable training loss to and the lowest validation loss out of the architectures with convolutional embedding.

Is is noteworthy that the convolutional embedding architecture gives a much lower validation than training loss, which is generally not expected in neural network training. In this experiment, the effect may be attributed to the very small validation set. We already employ a comparably small training set, see also Sections 5.1 and 5.5, and only 1% of the pretraining data is used in validation. Thus, the occurrence may be explained as an outlier due to small validation set size. The architecture using only the transformer appears to need a larger number of pretraining samples to converge and solve the pretraining task well. This slow rate of convergence is further illustrated in Fig. 5.3, where reconstruction of an example input without masking is shown. As in Fig. 5.2, the example input is taken from The PhysioNET dataset, subject 1, run 5,

Architecture	$N_{pretrain}$	$\mathcal{L}^{train}$	$\mathcal{L}^{val}$
Conv. only	200'000	31.3775	25.8156
Transformer only	200'000	32.2890	35.6024
Transformer only	400'000	29.1659	25.3832
Transformer only	800'000	**25.9633**	**22.5604**

Table 5.3: Pretraining losses of partial architectures.

Architecture	A_{ds}^{train}	A_{ds}^{val}	A_{ds}^{test}
BrainBERT	52%	53%	52%
Conv., no Downsampling	67%	66%	63%
Conv., Pooling	**70%**	**69%**	**68%**
Conv., Strided Convolution	62%	65%	58%
STFT + Vision Transformer	55%	53%	53%
Conv. + Vision Transformer	63%	64%	54%

Table 5.4: Downstream classification accuracies.

the first segment. We visualize the middle 3 seconds of channel 6 in the figure. In the top, middle, and bottom plot we use the architecture after 200'000, 400'000 and 800'000 updates, respectively. While the reconstruction in the top plot is close to 0 everywhere, the reconstructions obtained after more updates follow the input signal more closely. When comparing to Fig. 5.2, this indicates that the use of convolutional embedding decreases the number of updates needed to accurately reconstruct the unmasked input. It thus appears that the transformer is reliant on long taining and potentially also a large training data set. This is supported by observations made in [3, 24]. Furthermore, for the architectures using convolutional embedding and transformer together, the question of how the different parts contribute to solving the pretraining task arises. The slow convergence of the transformer could imply that the pretraining task is mostly solved by the convolutional architecture of these networks.

5.4 Downstream Classification

In the following, we present the results of training the downstream classifier network on the features extracted by the various architectures. As the dataset and all splits are balanced with respect to the labels, the accuracy of random guessing is 50% for this task. The EEGNet model, see Section 3.2.2 and [2], achieves a test accuracy of 84% on this train-test split. We consider this as the state of the art result to compare to.

The training, validation and test accuracies of the considered architectures achieved in downstream classification are presented in Table 5.4. Looking at the results from top to bottom, we see that our adapted version of BrainBERT fails to significantly improve on

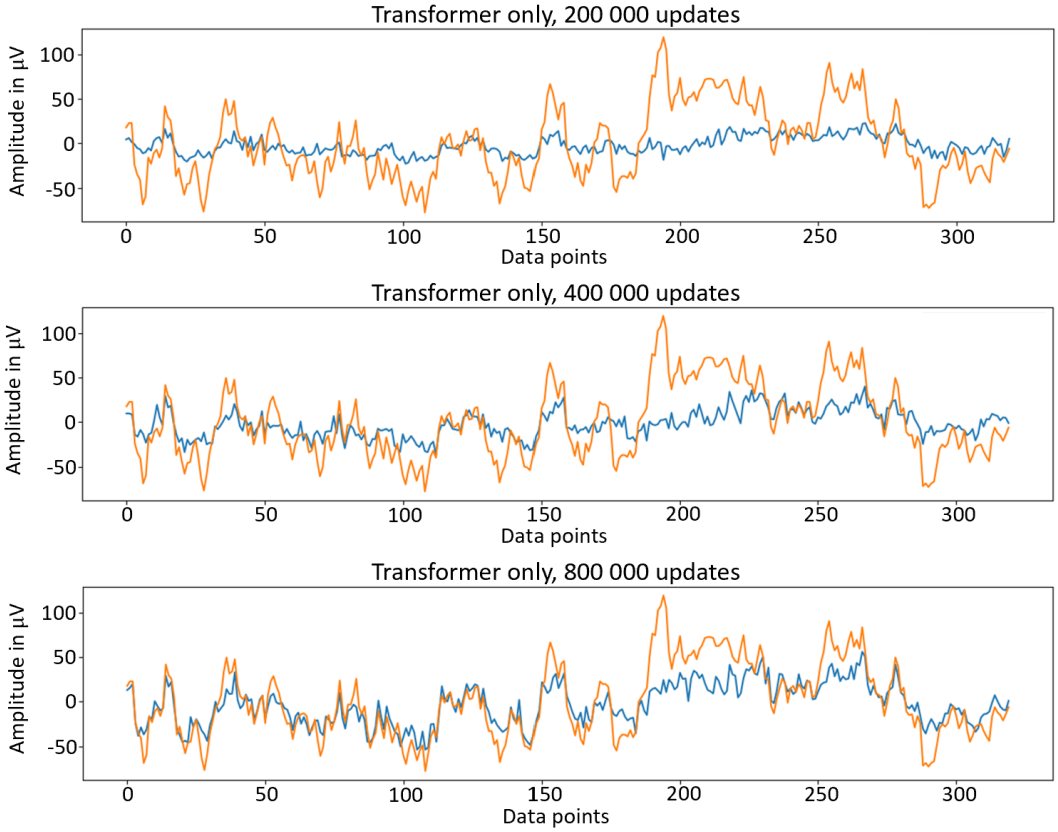

Figure 5.3: Example input (red) and reconstruction (blue) when using only the transformer, after 200'000 (top), 400'000 (middle) or 800'000 (bottom) updates.

the baseline provided by random guessing. When swapping the STFT embedding for the learned convolutional embedding, we observe test accuracies between 58% and 68% for all three considered approaches. The most accurate among them is the one using pooling for temporal downsampling, followed by the one without downsampling. The one with strided convolution performs worst out of the three. Looking at the two architectures with vision transformers, we see that the one using STFT embedding reaches accuracies around the 50% mark, only being marginally better than random guessing. When using the convolutional embedding together with the vision transformer, we achieve 63% accuracy on the training split, but only reach 54% on the test split.

The BrainBERT network is shown to extract meaningful features that yield promising downstream classification results in [1]. However, our approach to transfer BrainBERT to the extracranial EEG setting and the PhysioNET dataset has not given the desired results. This may partially be explained by the increased noise in extracranial EEG data and the transfer from tasks based on videos shown to subjects during recording to MI tasks. We think that another important component to this result is the lack of an elaborate channel aggregation scheme. In the original iEEG setting, electrodes are placed

Architecture	$N_{pretrain}$	A_{ds}^{train}	A_{ds}^{val}	A_{ds}^{test}
Conv. only	200'000	**65%**	**65%**	**62%**
Transformer only	200'000	50%	47%	52%
Transformer only	400'000	51%	57%	52%
Transformer only	800'000	53%	57%	52%

Table 5.5: Downstream classification accuracies for partial models.

at areas of interest on the brain. In our PhysioNET setup however, electrodes are placed to cover the skull. Networks successfully working with extracranial EEG data, such as [2, 6, 24], implement an elaborate form of channel aggragation using convolutional layers or transformers. This leads us to the introduction of explicit channel aggregation before application of the transformer, by integrating block 1 of the EEGNet architecture, see Section 3.2.2 and [2].

The results obtained from the architectures with convolutional embedding show that they manage to learn features that allow downstream classification with up to 68% testing accuracy. This significantly improves on the baseline set by random guessing, indicating that the learned features express information helpful for solving the motor imagery task. The accuracy is however not in the realm of the state of the art 84% achieved by EEGNet. Applying pooling has a positive effect on the downstream classification accuracy compared to the model without downsampling. This could be attributed to the imposed reduction of the number of features in team. It forces the network to learn features that are helpful for the reconstruction network during pretraining, while only having access to the averaged values. Interestingly, the network using strided convolutions cannot profit from the reduction of features in time, despite being a generalization of the pooling operation and thus potentially more expressive.

The architecture using STFT and a vision transformer is modeled after TSD [3]. It combines the use of STFT as in BrainBERT with the idea of channel aggregation before the transformer. Despite this promising approach and the successful application as a classifier in the original paper, our adaption only yields slightly improved results compared to random guessing. The network was already not performing well in pretraining when compared to BrainBERT, see Table 5.1. As mentioned in Section 5.3, this may be due to the reduction to a small number of features in the hidden layers of the transformer. If the network is unable to learn representations useful for improving on the pretraining loss, the representations are likely not helpful for downstream tasks either. The second architecture employing a vision transformer is using temporal convolution before it. It manages to achieve training and validation accuracies comparable to the other architectures with convolutional embedding, but performs worse on the test split. As this approach uses the same vision transformer as the one with STFT embedding, it potentially also suffers from the small hidden layer size.

Table 5.5 shows the downstream classification results for the partial architectures. The architecture using convolutional embedding and no transformer achieves results closely

comparable to its counterpart with transformer, see Table 5.4. The architecture using only the transformer on the other hand gives accuracies close to the guessing accuacy of 50% after 200′000 updates. Increasing the number of updates slightly increases the training accuracy, improves classification on the validation split and leaves the test accuracy unchanged.

The similarity of the results when only using the convolutional embedding and the results when using an additional transformer suggests that the extraction of expressive features can probably be attributed mostly to the convolutional embedding. This was already implied by the results of Table 5.2 and Table 5.3. Furthermore, using the transformer without the convolutional embedding gives accuracies not significantly above guessing accuracy. The only exception from this is the increase in validation accuracy for an increased number of training steps. It was not to be expected that these are significantly larger than the respective training accuracies. This effect may be attributed to the small validation split of only three subjects, as the small validation set size increases the likelihood of outliers in accuracy estimation.

5.5 Neural Scaling Law

In this section, we consider the Chinchilla Scaling Law (CSL) introduced in [25]. It is a relation describing the optimal relation of the following parameters in neural network training: network size in number of trainable weights N, number of tokens for training D, training effort in FLOPs C and training loss L. It assumes the relations

$$C = C_0 N D \tag{5.1}$$

where $C_0 = 6$ and

$$L = \frac{A}{N^\alpha} + \frac{B}{D^\beta} + L_0. \tag{5.2}$$

The parameters A, α, B and β are statistically estimated from training on the Chinchilla network introduced in the paper, a large language model employing a transformer. The estimation is performed using Chinchilla model versions with 70 Million to 16 Billion parameters and data sets containing 5 Billion to 500 Billion tokens. Plugging in the estimated values and solving for minimal loss L gives

$$N_{opt}(C) = 0.6C^{0.45} \tag{5.3}$$

and

$$D_{opt}(C) = 0.3C^{0.55} \tag{5.4}$$

for fixed C. Combining Eq. (5.3) and Eq. (5.4) we obtain

$$D_{opt}(N) = 0.3\left(\frac{N}{0.6}\right)^{\frac{11}{9}}. \tag{5.5}$$

Architecture	D	N	$D_{opt}(N)$	$N_{opt}(D)$
BrainBERT	10'631'174	43'183'144	1'203'151'971	901'534
Conv., no Downsampling	4'523'904	42'593'152	1'183'091'534	448'106
Conv., Pooling	565'488	42'593'152	1'183'091'534	81'750
Conv., Strided Convolution	565'488	42'593'216	1'183'093'707	81'750
STFT + Vision Transformer	742'203	917'376	10'859'945	102'121
Conv. + Vision Transformer	565'488	546'201	5'762'214	81'750
Conv. only		42'593'152		
Transformer only	4'523'904	2'492'224	36'840'083	448'106

Table 5.6: Number of input tokens and parameters, as well as optimal number of input tokens and parameters according to CSL for the different architectures.

and

$$N_{opt}(D) = 0.6\left(\frac{D}{0.3}\right)^{\frac{9}{11}}. \tag{5.6}$$

We now attempt to coumpute the optimal model size using the number of tokens in our pretraining dataset. To this end, we first calculate the number of tokens D we obtain from the PhysioNET dataset. The results of the following calculations are summarized in Table 5.6. The value of D depends on the model we use. In the case of BrainBERT, we have 85 subjects, 3 runs per subject used in pretraining, and 14 events per run. This gives 3570 events in total. Each one provides 64 input segments (one per channel). From all such segments, we remove 1% for validation. The remainng ones are transformed into 47 tokens each using STFT. In total, we thus have $D = 10'631'174$ tokens. When using the convolutional embedding without downsampling, we have the same number of events, each provinding one input segment. After removing the validation split, each remaining segment gives 1280 input tokens (one per input data point), which means we have $D = 4'523'904$ tokens in total. The same holds for the architecture using the transfomer without the convolutional embedding. The architectures with downsampling however reduce the number of input tokens by a factor of 8, due to the reduction of samples in time. This reduces the number of tokens to $D = 565'488$. For the architecture using STFT and a vision transformer, we have 210 tokens per event in the training split (one per extracted patch), which yields $D = 742'203$. Similarly, for the architecture using convolutional embedding and a vision transformer, we have 160 tokens per event in the training split, giving $D = 565'488$.

As comparison, the data set used to pretrain BrainBERT contains 4551 electrode-hours of data recoreded at 2'000 Hz. In preprocessing, the data is split into 5-second segments, and the STFT is applied with a window moving 25 data points between time slots, see Section 3.1.1. Since each time slot's DFT forms a token , we calculate 1'310'688'000 tokens for this dataset.

With the above calculations and Eqs. (5.5) and (5.6), we obtain the results in Table 5.6.

N represents the actual number of parameters in the architectures. For the architecture using only a convolutional embedding and no transformer, no input tokens can be calculated, as it does not feature a transformer. Comparing the calculated optimal network sizes N_{opt} to the actual sizes N_{actual}, one can see that the architectures used in this work contain much more parameters than CSL considers optimal. Similary, comparing D_{opt} to D, we see that CSL suggests the use of larger data sets than PhysioNET to train our architectures. Thus, applying this neural scaling law to our setting implies that our architectures are too large, our dataset is too small, or both. In Sections 5.3 and 5.4, we discuss that the architecture using only the transformer is taking a larger number of pretraining updates to minimize the loss on the pretraining task, and cannot yield results above guessing accuracy in downstream classification. When using a convolutional embedding before the transformer, however, the pretraining loss is optimized in fewer updates and downstream classification yields improved results. This could suggest that the number of tokens in the training set is too small for the application of these transformer architectures. The results obtained from the application of CSL - which was estimated for a network using transformers - support this argument.

Looking specifically at the BrainBERT architecture, we see that the predicted number of tokens D_{opt} matches the number of tokes used in [1] (1'310'688'000) quite closely. This implies that we may attribute some of the difficulties we encounter in transfering BrainBERT to the PhysioNET dataset to the the small size thereof. In our experiments, we have access to over 100 times less pretraining tokens.

The problem of low data set size for pretraining could be solved by using a large, publicly available dataset that is not recorded in the MI setting. Afterall, we pretrain under self-supervision, i.e. no labels are needed. The Temple University Hosplital (TUH) EEG Data Corpus [26], for example, contains 29.1 electrode-years of data, compared to the 0.5 electrode-years used to pretrain BrainBERT in the original paper. It is sampled at frequencies between 250 Hz and 500 Hz, compared to the 2'000 Hz iEEG signals used in BrainBERT. To estimate a lower bound on the number of training tokens this dataset provides, we assume a samling frequency of 250 Hz and 30 channels on average. In the case of the convolutional architecture without downsampling, we obtain 7.64 Billion tokens, and for the architectures with pooling or strided convolution, we get 956 Million tokens. This means that it would overall give a far larger number of training tokens than the PhysioNET dataset. For no downsampling, it surpasses the size CSL predicts to be necessary, for pooling and strided convolution, it gets close. However, while the pretraining procedure indeed does not require labeled data, it is still assumed that the data is of the same or very similar characteristics as the one considered in downstream classification. Pretraining on the TUH dataset therefore is likely to improve performance compared to our approach, but may not lead to the results we could achieve if we had the same amount of data recorded from subjects performing MI tasks. Furthermore, the TUH dataset considers varying sampling frequencies and electrode configurations. If one uses this data in pretraining, these aspects have to be considered, e.g. by adequate preprocessing.

It is important to note that the results in this section are to be taken with caution. They may only provide a rough indication of the optimal network size in our case, since the

setting in which the parameters of CSL were estimated differs much from ours. Namely, the scaling law was computed for much larger model and dataset sizes. Furthermore, the architecture of the Chinchilla model differs to ours both in the transformer dimensions and the layers surrounding it. Finally, the meaning of the number of tokens differs largely across the two settings. In natural language processing, a token typically is a word, a syllable, or as in the case of Chinchilla, a fixed number of characters. The number of tokens D in the training set is thus very indicative of the the size of the text it contains. In our case however, D is arguably a weaker indication of the training set size, as it varies considerably depending on the embedding we apply.

Chapter 6

Conclusion and Future Work

The experiments conducted in this work show that our approach of transfering BrainBERT to the extracranial EEG MI setting did not work as desired. Exchanging the STFT for a learned convolutional embedding increases classification accuracy, although not to the state of the art achieved by EEGNet. Still, the embedding seems to provide the network with the temporal and depth-wise processing needed to learn meaningful latent representations. The same is observed when applying a vision transformer: We cannot solve the MI classification task with an STFT embedding, but improve when introducing a convolutional architecture.

Overall, our efforts to bring transformer-based foundation models to the realm of MI EEG tasks show most success when building on EEGNet [2]. Thus, it may be beneficial for future work to attempt to build a model based on EEGNet. Then transformers and foundation model approaches could be introduced, which potentially enable the extraction of more meaningful features.

Using partial architectures, we obtain results that suggest that the convolutional architecture contributes to the extraction of expressive features, whereas the transformer architecture on its own struggles to do so. Considering the pretraining procedure, it appears that the transformer converges slowly compared to the convolutional architecture. Thus, one needs a longer pretraining time and possibly more pretraining data, to optimize it. We conclude that the architectures suffer from the small amount of pretraining data we use - or have too many parameters for the small pretraining dataset. This is supported by calculating optimal dataset and model sizes by means of CSL. This leads us to suggest the use of a larger pretraining dataset, such as the TUH corpus we discuss. With such an increased pretraining dataset, one may be able to extract more expressive features using transformer-based foundation models.

List of Acronyms

AE . . . . . .Auto-encoder

BCI . . . . . .Brain-Computer Interface

CSL . . . . . .Chinchilla Scaling Law

EEG . . . . . .Electroencephalography

iEEG . . . . . .Intracranial electroencephalography

MI . . . . . .Motor Imagery

MLP . . . . . .Multilayer Perceptron

NLP . . . . . .Natural Language Processing

RNN . . . . . .Recurrent Neural Network

STFT . . . . . .Short-Time Fourier Transform

TUH . . . . . .Temple University Hospital

List of Figures

List of Tables

Bibliography

[1] C. Wang, V. Subramaniam, A. U. Yaari, G. Kreiman, B. Katz, I. Cases, and A. Barbu, "Brainbert: Self-supervised representation learning for intracranial recordings," 2023.

[2] 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

[3] Y. Ma, C. Liu, M. S. Ma, Y. Yang, N. D. Truong, K. Kothur, A. Nikpour, and O. Kavehei, "Tsd: Transformers for seizure detection," *bioRxiv*, 2023. [Online]. Available: https://www.biorxiv.org/content/early/2023/01/25/2023.01.24.525308

[4] G. Schalk, D. McFarland, T. Hinterberger, N. Birbaumer, and J. Wolpaw, "Bci2000: a general-purpose brain-computer interface (bci) system," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1034–1043, 2004.

[5] M. Soufineyestani, D. Dowling, and A. Khan, "Electroencephalography (eeg) technology applications and available devices," *Applied Sciences*, vol. 10, no. 21, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/21/7453

[6] J. Zhou, Y. Duan, Y. Zou, Y.-C. Chang, Y.-K. Wang, and C.-T. Lin, "Speech2eeg: Leveraging pretrained speech model for eeg signal recognition," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 31, pp. 2140–2153, 2023.

[7] P. Hallgarten, D. Bethge, O. Özdenizci, T. Grosse-Puppendahl, and E. Kasneci, "Ts-moco: Time-series momentum contrast for self-supervised physiological representation learning," 2023.

[8] D. Kostas, S. Aroca-Ouellette, and F. Rudzicz, "Bendr: using transformers and a contrastive self-supervised learning task to learn from massive amounts of eeg data," 2021.

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

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[11] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," 2020.

[12] D. Bethge, P. Hallgarten, T. Grosse-Puppendahl, M. Kari, L. L. Chuang, O. Özdenizci, and A. Schmidt, "Eeg2vec: Learning affective eeg representations via variational autoencoders," 2022.

[13] Q. Zhu, X. Zhao, J. Zhang, Y. Gu, C. Weng, and Y. Hu, "Eeg2vec: Self-supervised electroencephalographic representation learning," 2023.

[14] Y. Guo, X. Zhang, Z. Gong, A. Wang, and W. Wang, "End-to-end translation of human neural activity to speech with a dual-dual generative adversarial network," 2022.

[15] Y. Yue, J. D. Deng, D. D. Ridder, P. Manning, and D. Adhia, "Variational autoencoder learns better feature representations for eeg-based obesity classification," 2023.

[16] 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," *IEEE Transactions on Biomedical Engineering*, vol. 69, no. 6, p. 2105–2118, Jun. 2022. [Online]. Available: http://dx.doi.org/10.1109/TBME.2021.3137184

[17] T. Behrouzi and D. Hatzinakos, "Graph variational auto-encoder for deriving eeg-based graph embedding," *Pattern Recognition*, vol. 121, p. 108202, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320321003848

[18] X. Li, Z. Zhao, D. Song, Y. Zhang, J. Pan, L. Wu, J. Huo, C. Niu, and D. Wang, "Latent factor decoding of multi-channel eeg for emotion recognition through autoencoder-like neural networks," *Frontiers in Neuroscience*, vol. 14, 2020. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2020.00087

[19] T. Ahmed and L. Longo, "Examining the size of the latent space of convolutional variational autoencoders trained with spectral topographic maps of eeg frequency bands," *IEEE Access*, vol. 10, pp. 107 575–107 586, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:252796174

[20] Y. Zhu and M. D. Wang, "Automated seizure detection using transformer models on multi-channel eegs," in *2023 IEEE EMBS International Conference on Biomedical and Health Informatics (BHI)*, 2023, pp. 1–6.

[21] P. Busia, A. Cossettini, T. M. Ingolfsson, S. Benatti, A. Burrello, M. Scherer, M. A. Scrugli, P. Meloni, and L. Benini, "Eegformer: Transformer-based epilepsy detection on raw eeg traces for low-channel-count wearable continuous monitoring devices," in *2022 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2022, pp. 640–644.

[22] Z. Wan, M. Li, S. Liu, J. Huang, H. Tan, and W. Duan, "Eegformer: A transformer–based brain activity classification method using eeg signal," *Frontiers in Neuroscience*, vol. 17, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257744035

[23] X. Wang, M. Hersche, B. Tömekce, B. Kaya, M. Magno, and L. Benini, "An accurate eegnet-based motor-imagery brain–computer interface for low-power edge computing," in *2020 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, 2020, pp. 1–6.

[24] D. Zhang, Z. Yuan, Y. Yang, J. Chen, J. Wang, and Y. Li, "Brant: Foundation model for intracranial neural signal," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. [Online]. Available: https://openreview.net/forum?id=DDkl9vaJyE

[25] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de Las Casas, L. A. Hendricks, J. Welbl, A. Clark, T. Hennigan, E. Noland, K. Millican, G. van den Driessche, B. Damoc, A. Guy, S. Osindero, K. Simonyan, E. Elsen, J. W. Rae, O. Vinyals, and L. Sifre, "Training compute-optimal large language models," 2022.

[26] I. Obeid and J. Picone, "The temple university hospital eeg data corpus," *Frontiers in Neuroscience*, vol. 10, 2016. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2016.00196