Bachelor thesis

Artificial Intelligence

Radboud University

Routing Mechanisms in Convolutional Neural Networks for EEG Decoding

Author:
Diego García Cerdas
S1020485

First supervisor:
Dr. R.S. van Bergen
Donders Institute for Brain, Cognition and Behaviour
rubenvanbergen@gmail.com

Second supervisor:
Dr. T.C. Kietzmann
Donders Institute for Brain, Cognition and Behaviour
tim.kietzmann@donders.ru.nl

June 18, 2021

Abstract

Convolutional neural networks (CNNs) are becoming increasingly popular tools for decoding electroencephalographic (EEG) data. They alleviate the need for domain expertise in a typical decoding pipeline by being able to learn feature representations from raw input. However, the impact of new deep learning research ideas on EEG decoding is yet to be fully explored. This bachelor thesis introduces the idea of routing mechanisms in CNNs for EEG decoding. We present two mechanisms: routing through optimization and routing through attention. Each mechanism is implemented within the first layer of a CNN specifically designed to handle EEG data. Through our experimental setup, we aim to answer the following question: *What is the impact of each routing mechanism on EEG decoding accuracies?* To do so, we look at our models' learning process and test for statistically significant differences in performance between our networks and a baseline implementing regular convolution. Our experimental results show that both mechanisms function effectively and our models reach at least 89% decoding accuracy. However, no statistically significant impact is found for either mechanism. The work here presented sets a framework that future research can build upon in order to create an efficient flow of information through the network.

Contents

1. INTRODUCTION

Machine learning (ML) methods play an important role in the extraction of relevant features from electroencephalographic (EEG) recordings within typical brain-signal decoding pipelines [1, 2, 3]. These methods have been successfully applied on rehabilitation after stroke [4], sleep stage classification [5], and operation of prosthetic devices [6], among others. However, each new decoding pipeline relies on application-specific signal preprocessing, feature selection, and classification stages [7, 8, 9]. Thus, it requires significant expertise and prior knowledge about the EEG data to be decoded [10].

Deep learning alleviates these requirements by combining feature extraction and classification stages within a single model used on minimally-processed or raw data [11]. This method, known as *end-to-end* learning, has proven to be successful in domains such as computer vision [12, 13] and natural language processing [14, 15]. For example, convolutional neural network (CNN) models are recognized for their success in several challenging image classification tasks [16, 17, 18].

Recent studies have explored the use of deep learning techniques for EEG decoding. Specifically, different CNN models have been developed for motor and imagery tasks [19, 20, 21], emotion recognition [22], and seizure detection [23, 24], among others. Some of these models incorporate domain-specific knowledge in their design and reach performances at least as good as popular ML feature extractors such as filter bank common spatial patterns (FBCSP) [25]. Moreover, these studies highlight the benefits of incorporating recent advances in CNN research, such as batch normalization [26] or exponential linear units [27], to these models [19]. Hence, the relevance of new deep learning ideas in the task of brain-signal decoding is a research area yet to be fully explored.

In this study, we introduce the idea of *routing* in convolutional neural networks for EEG decoding. Originally, routing is described in the context of capsule networks [28], where units in lower-level layers direct their output towards a single higher-level unit through the use of *routing weights*. These weights are updated iteratively, depending on the level of *agreement* between the representations at each layer. For further details on capsule networks, see [28, 29].

In the context of CNNs, routing can be used to constrict the information flow between increasingly higher-level feature maps [30, 31]. As described in detail in the Methods section, a routing mechanism involves two consecutive convolutional layers. A source element in the earlier layer (i.e., a characteristic of its feature maps) is directed, via a set of routing weights, towards a target characteristic in the second layer. Routing provides a way for the CNN to optimize the forward pass of a (mini-)batch of input data, given the current state of the network. Moreover, the flexibility in designing a routing mechanism permits the use of domain-specific knowledge to constrict the flow of information in a way that is logical, given the characteristics of the input data.

In light of this, we present two routing mechanisms for the purpose of EEG decoding. The first one, denoted *routing through optimization* directs input EEG channels towards learned temporal filters. It updates the routing weights by optimizing a mean squared error (MSE) loss within the earlier layer. The second one, referred to as *routing through attention*, directs a single time point in the raw EEG input towards a single unit in the later layer. It updates the routing weights by calculating the dot product between the earlier layer's feature map and a top-down prediction, reminiscent of the *self-attention* method introduced in [32]. Consequently, in this study, we address two key questions:

- What is the impact of *routing through optimization* on EEG decoding accuracies?
- What is the impact of *routing through attention* on EEG decoding accuracies?

To do so, we introduce a base CNN architecture inspired by previous work [19, 16] and designed specifically for handling raw EEG data. From this base architecture, we stem two variants, denoted *OptimConvNet* and *AttConvNet*, each implementing a different routing mechanism within its first layer. We first analyze the learning process for each variant in order to ensure the mechanisms work correctly. Then, to address whether the routing mechanisms have an impact in decoding accuracies, we analyze the offline decoding performance of each variant in a motor execution EEG dataset (refer to section 2.1.2 for details). To control for implementational differences, we compare these results with those obtained by a version of each variant without routing.

In the following section, we introduce the problem tackled and data used in our analysis, as well as the models we use and our experimental setup. Section 3 presents the results of our analysis. We discuss these results in Section 4, and present a conclusion in Section 5.

2. METHODS

As a first step, we formally introduce the task of EEG decoding as a multi-class classification problem and present the dataset used in this study. Then, we describe the use of convolutional neural networks and routing mechanisms for EEG decoding, including the architecture and variants that we evaluated. Finally, our experimental setup is described in detail, including the protocol, training procedure, and implementational details. The code used in this study is available under https://github.com/diegogcerdas/routing-cnn-eeg.

2.1. EEG Decoding

2.1.1. Problem Definition

The task of brain-signal decoding carried out in this study is formally defined as a supervised multi-class classification problem [19]. We assume that, for each subject i, we are given a dataset $D^i = \{(X_1, y_1), ..., (X_{N_i}, y_{N_i})\}$ where:

- $X_j \in \mathbb{R}^{N_c \cdot N_t}$ is a multivariate signal measured by N_c EEG channels at N_t discrete time samples. It denotes the input matrix of trial j, with $1 \leq j \leq N_i$.
- $y_j \in L$ is the class label of trial j, where the label space $L = \{0, 1, ..., (K-1)\}$ defines a set of K class labels, with $K > 1$.
- N_i is the number of recorded trials for subject i.

A classifier $f(X^j; \theta) : \mathbb{R}^{N_c \cdot N_t} \rightarrow L$, parameterized by weights θ, is trained to correctly assign the label y_j to a given input trial X_j. In this study, the classifier is represented by a convolutional neural network, as described in Section 2.2.

2.1.2. Dataset

This work uses the High Gamma Dataset (HGD) [19], which contains recordings for 14 healthy subjects (of which we only use 9 subjects in this study due to limited computing power). It is based on 128 electrodes, out of which only 44 sensors covering the motor cortex are used. The dataset consists of a motor execution task: at every trial, subjects are instructed to execute one of four movements: left hand, right hand, feet, and rest. Trials consist of the four-second segment [-0.5, 3.5]s relative to cue. The data is originally sampled at 500 Hz, and further resampled to 250 Hz.

Given our problem definition, HGD thus provides input trials $X_j \in \mathbb{R}^{44 \cdot 1000}$ with class labels $y_j \in \{0, 1, 2, 3\}$. The number of recorded trials is on average 963 per subject (standard deviation 150.9).

2.2. Convolutional Neural Networks

In this section, we provide a basic description of CNNs and routing mechanisms. Then, we introduce our base architecture, from which we derive the two variants used in our study.

2.2.1. CNN Basics

CNNs are neural network models designed to process matrix data [11]. Prototypical CNNs extract features through three basic components:

1. *Convolutional layer*. Units in this layer are organized in feature maps. Each unit's receptive field comprises a set of neighboring units in the previous layer, to which it is connected through a set of weights called a 'kernel'. Kernels are shared by all units within the same feature map, but vary across feature maps. The result of each local weighted sum can be passed through an element-wise non-linearity [33].
2. *Pooling layer*. This layer reduces the resolution of its input feature maps in order to merge semantically similar features into a single one. To do so, a unit in this layer typically preserves the average or maximum value within its receptive field [34].
3. *Fully-connected layer*. This layer connects all units from the previous layer to every single unit in the current layer.

By stacking a series of convolutional and pooling layers, CNNs extract increasingly higher-level feature representations. These in turn get passed to

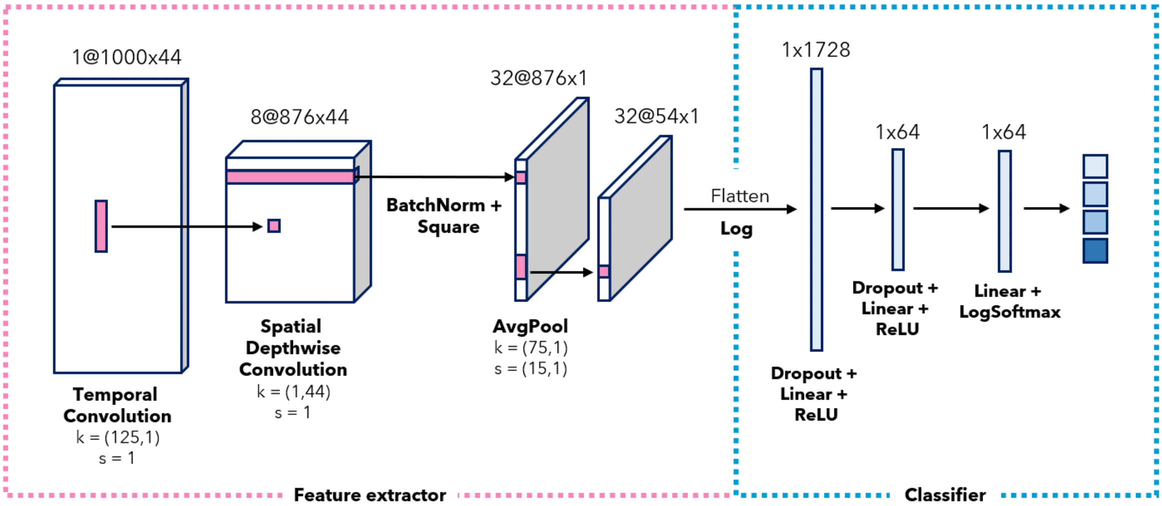

Figure 1: Base architecture. Raw EEG trials pass through a feature extractor block and a classifier block. Cuboids denote feature maps; pink regions are convolution/pooling layers; blue rectangles are 1-D feature vectors. Sizes belong to the HGD dataset. Note: proportions of feature maps and kernels are only approximate.

fully-connected layers in order to perform high-level reasoning towards a goal (e.g., classification) [17, 35].

A CNN is trained through backpropagation [36] to minimize a loss function, according to the type of problem it aims to solve. To do this, it is common to employ an optimizer, such as Adam [37] or SGD [38]. For more details on CNNs, see [11, 39].

2.2.2. Routing Mechanisms

A routing mechanism for CNNs is used to constrict information flow through the network. It involves two consecutive convolutional layers, l_1 and l_2. A given *source* characteristic of the feature maps in l_1 is routed, via a set of routing weights R, towards a *target* characteristic in l_2. Routing weights get updated throughout N_R iterations within a single forward pass of the input data through the network. This allows for the CNN to process a (mini-) batch of data optimally, given the current state of the network.

Therefore, the design of an effective routing mechanism concerns three main choices:

1. Which characteristic of the feature maps in l_1 defines the routing *source*.

2. Which characteristic of the feature maps in l_2 defines the routing *target*.

3. How to update the routing weights R.

The flexibility in defining these three design-choices permits the use of domain-specific knowledge, in order to provide a routing mechanism that is useful given the characteristics of the input data.

2.2.3. Base Architecture

Our base architecture is illustrated in Figure 1. It is a compact CNN architecture divided in two blocks as follows:

1. *Feature extractor*. This part is based on the Shallow ConvNet model proposed in [19]. The first convolutional layer operates along the time axis, allowing the kernels to extract relevant frequencies of the input signals. This layer uses F kernels of size (125,1), corresponding to a time duration of 0.5s. This size differs from the original implementation in [19], as larger window sizes led to increased accuracies in preliminary experiments). Moreover, this layer implements the different routing mechanisms explored in this study (described in the next Section 2.2.4).

   The second layer performs spatial convolution using kernels of size $(1,N_c)$. In contrast to [19], we use Depthwise Convolution [40], allowing the network to learn D spatial filters for each temporal filter. This choice enables the extraction of frequency-specific spatial filters [20].

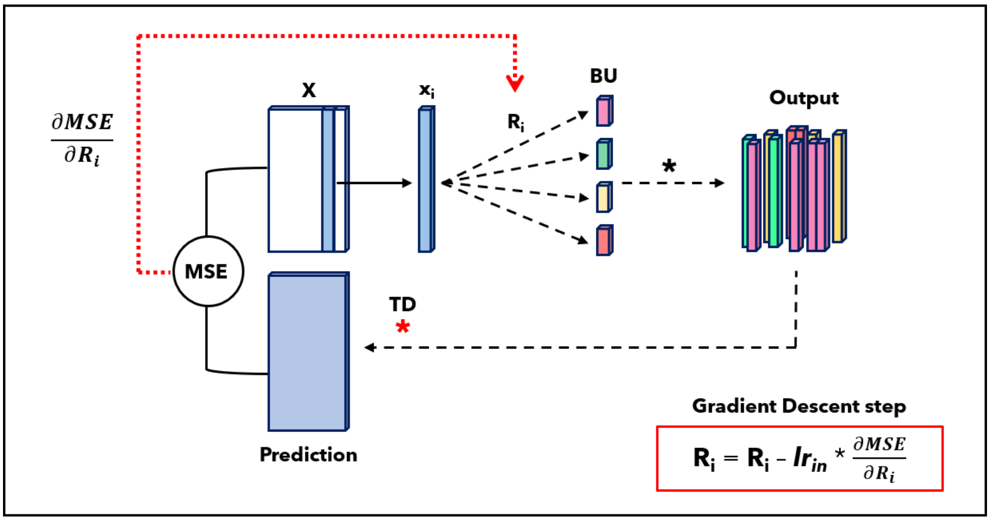

Figure 2: Routing through optimization. An EEG channel x_i is routed via routing weights R_i towards a single kernel. The routed input is convolved with kernels BU to produce the layer's output. Through transposed convolution with weights TD, a prediction is calculated. The gradient of an MSE loss (between the prediction and the actual input) with respect to the routing weights is used to update them via N_r iterations of gradient descent. * (black) denotes convolution; * (red) denotes transposed convolution. Red arrows denote the gradient descent process. Note: proportions, number of channels, and number of features are only illustrative.

The network then performs batch normalization and a squaring non-linearity. Finally, an average pooling layer with kernel size (75,1) and stride (15,1) is followed by a logarithmic activation function. This design is inspired by the trial log-variance computation of the FBCSP algorithm [25].

2. *Classifier*. The feature representations are flattened and passed through a block containing a Dropout layer [41] with p=0.75, a fully-connected (FC) layer that shrinks the representation to $F * F$ units, and a ReLU activation [39]. A second block with the same shape (Dropout + FC + ReLU) preserves the $F * F$ units. Finally, the features are passed through a FC layer and LogSoftmax activation that outputs K units. This design is inspired by the classifier block in AlexNet [16].

2.2.4. Variants

Two variants are derived from our base architecture, each implementing a different routing mechanism within the first convolutional layer:

1. *Routing through optimization*. This mechanism is illustrated in Figure 2. The routing sources are EEG channels in the input trial, and the routing targets are the temporal filters learned by the first layer. In other words, this mechanism ensures that the information from each one of the N_c EEG channels in the input trial ends up in a single output feature map, out of F possibilities. We use routing weights $R_{i,j}$ from EEG channel i to feature map j such that:

$$\sum_{j=1}^{F} R_{i,j} = 1$$

This condition enforces the assumption that only one feature map receives the full information pertaining to EEG channel i, after proper optimization of the routing weights. In order to optimize the weights, the layer performs top-down prediction by performing transposed convolution on its output feature maps (obtained by convolving the routed input with the layer's set of kernels BU) via an additional set of weights TD; an MSE loss is calculated between the layer's prediction and its actual input; then, the gradient of the loss with respect to $R_{i,j}$ is used to update the routing weights via gradient descent (running for N_r iterations and learning rate lr_{in}); when gradient descent ends, the final output feature maps are calculated and passed on to the next layer to continue the forward pass.

This approach is inspired by previous at-

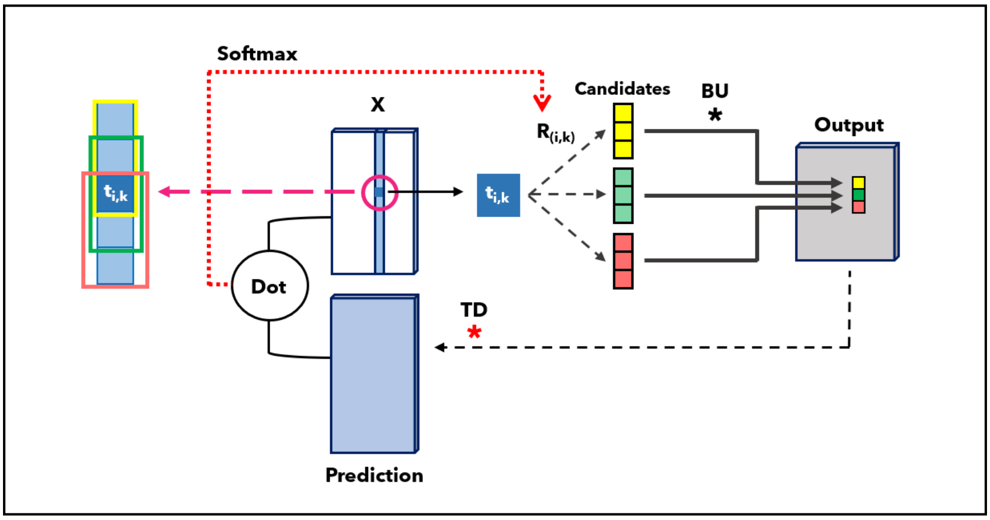

Figure 3: Routing through attention. An timepoint $t_{i,k}$ in the recording of channel i is routed via routing weights $R_{(i,k)}$ towards a single candidate sliding block. The routed input is convolved with kernels BU to produce the layer's output. Through transposed convolution with weights TD, a prediction is calculated. The dot product between the prediction and the actual input is passed to a softmax function in order to set the routing weights. This procedure is repeated for N_r iterations. Note: we display a kernel of size (3,1) for illustrative purposes. Sizes and proportions are illustrative only.

tempts to extract optimal frequency components from EEG data, in the context of Brain-Computer Interfaces [42, 43, 44]. The variant implementing this mechanism is denoted OptimConvNet-N_r.

2. *Routing through attention.* As illustrated in Figure 3, this mechanism routes a single timepoint $t_{i,k}$ from the recording of EEG channel i towards a single location (at column i) in the output feature maps. We use routing weights $R_{(i,k),j}$ from timepoint $t_{i,k}$ to candidate positions j. As described for the previous mechanism, these weights should sum up to 1 across all candidates. Candidate positions are the sliding local blocks, unfolded from the input feature maps, that contain the timepoint $t_{i,k}$. In order to update the routing weights a prediction is calculated in a similar manner as in *routing through optimization* (i.e., through convolution of the routed input and kernels BU, followed by transposed convolution with weights TD). As opposed to that mechanism, routing weights are here optimized by calculating the dot product between actual input and prediction, then applying softmax in order to ensure the sum-to-1 condition. This process is repeated for N_r iterations.

   This approach is reminiscent of the self-attention mechanism [32] and inspired by theory on capsule networks [28, 29]. This variant is denoted AttConvNet-N_r.

Implementational details for training the routing weights R, kernels BU, and weights TD for each variant are described in Section 2.3.2.

2.3. Experimental Setup

2.3.1. Protocol and Statistical Testing

For each of the 9 subjects we analyzed, we train four models (each model specifies N_r, the number of inner iterations in the corresponding routing mechanism):

- OptimConvNet-0
- OptimConvNet-20
- AttConvNet-0
- AttConvNet-10

Note: the maximum number of iterations for each routing mechanism is a hyperparameter in our study (in preliminary experiments, these numbers achieved a more stable and rapid convergence); moreover, the first layer in models with zero iterations is equivalent in a regular convolutional layer.

For each model, we qualitatively analyze the learning process by looking at training and validation losses and accuracies across epochs.

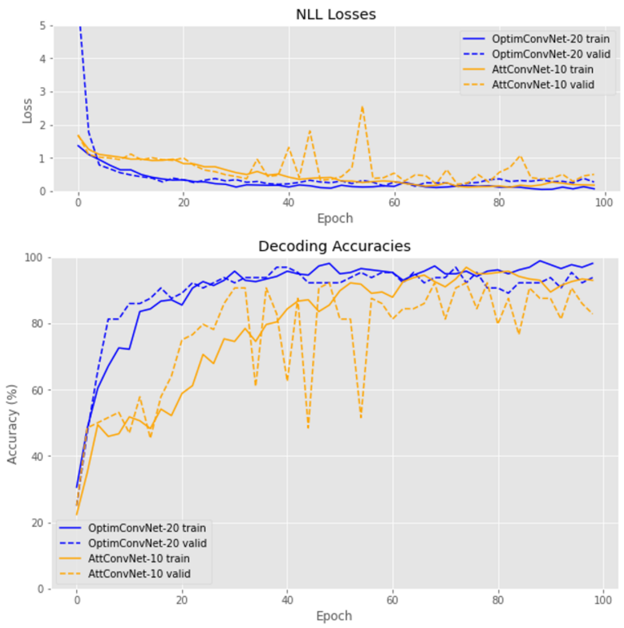

Figure 4: Learning process for OptimConvNet-20 and AttConvNet-10 for Subject 1. Upper image displays history of NLL losses throughout epochs. Lower image displays history of decoding accuracies (%). Blue lines belong to OptimConvNet-20 and orange lines belong to AttConvNet-10. Training stage is displayed in solid lines, while validation stage is in dashed lines.

We use the decoding accuracies for assessing the performance of our models. To show a complete picture of the decoding process, we also compare the confusion matrices across subject-specific models.

We use Wilcoxon signed rank tests [45] to check for statistically significant differences in the following contrasted conditions:

- OptimConvNet-0 and OptimConvNet-20
- AttConvNet-0 and AttConvNet-10

By doing so, we assess the impact in decoding accuracies of each routing mechanism, while controlling for implementational differences between models.

2.3.2. Training and Hyperparameters

In order to perform within-subject classification, we train each model separately on the data pertaining to a single subject. Each model outputs a subject-specific conditional distribution over the four classes; hence we train the model to minimize the *negative log-likelihood* (NLL) loss. The minimization of this loss is equivalent to minimizing the cross entropy between the true distribution of classes and the one defined by our model.

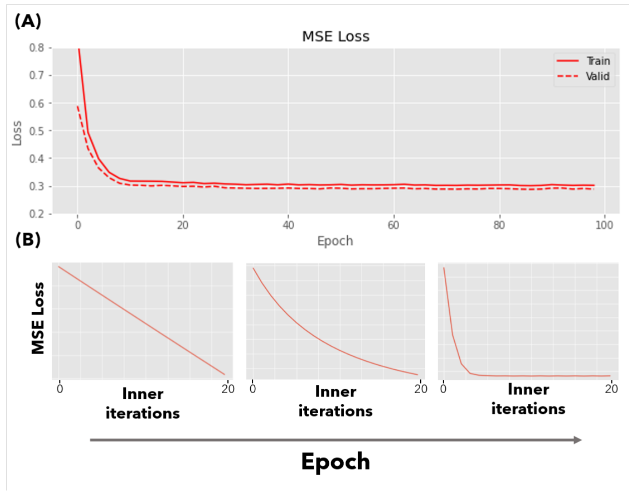

Figure 5: MSE loss history for Subject 1. (A) displays the MSE loss optimized at every epoch. (B) shows an schematic of the MSE loss history for the gradient descent process in *routing through optimization* throughout epochs. This process corresponds to the testing phase for Subject 1.

We use an Adam optimizer [37] with learning rate of 1e-3 to train the kernels BU in the first layer, as well as the parameters in the rest of the network, in order to minimize the NLL loss.

We keep the separation into training, validation, and test sets as used in [19]. Every model is trained for 200 epochs; we save the model weights for the epoch that performs best on the validation set and we use them to define our final model. All models use $D = 4$ spatial filters for each of the $F = 8$ temporal filters (see Section 2.2.3).

The remaining details on the training process differ per variant:

- *OptimConvNet*. We use a batch size of 16 trials. Moreover, we introduce an additional Adam optimizer with learning rate of 1e-1 to train the weights TD in the first layer, in order to minimize an MSE loss (this is possible because the network outputs an additional tensor containing the final prediction from the first layer). For the gradient descent process within the first layer, we implement gradient descent manually and use a learning rate lr_{in} of 1e+3 (we interpret this high value as a consequence of the normalization process in the layer).
- *AttConvNet*. We use a batch size of 8 trials. As opposed to the first routing mechanism, here we optimize all weights in the model (including weights TD) jointly with the same optimizer (described above).

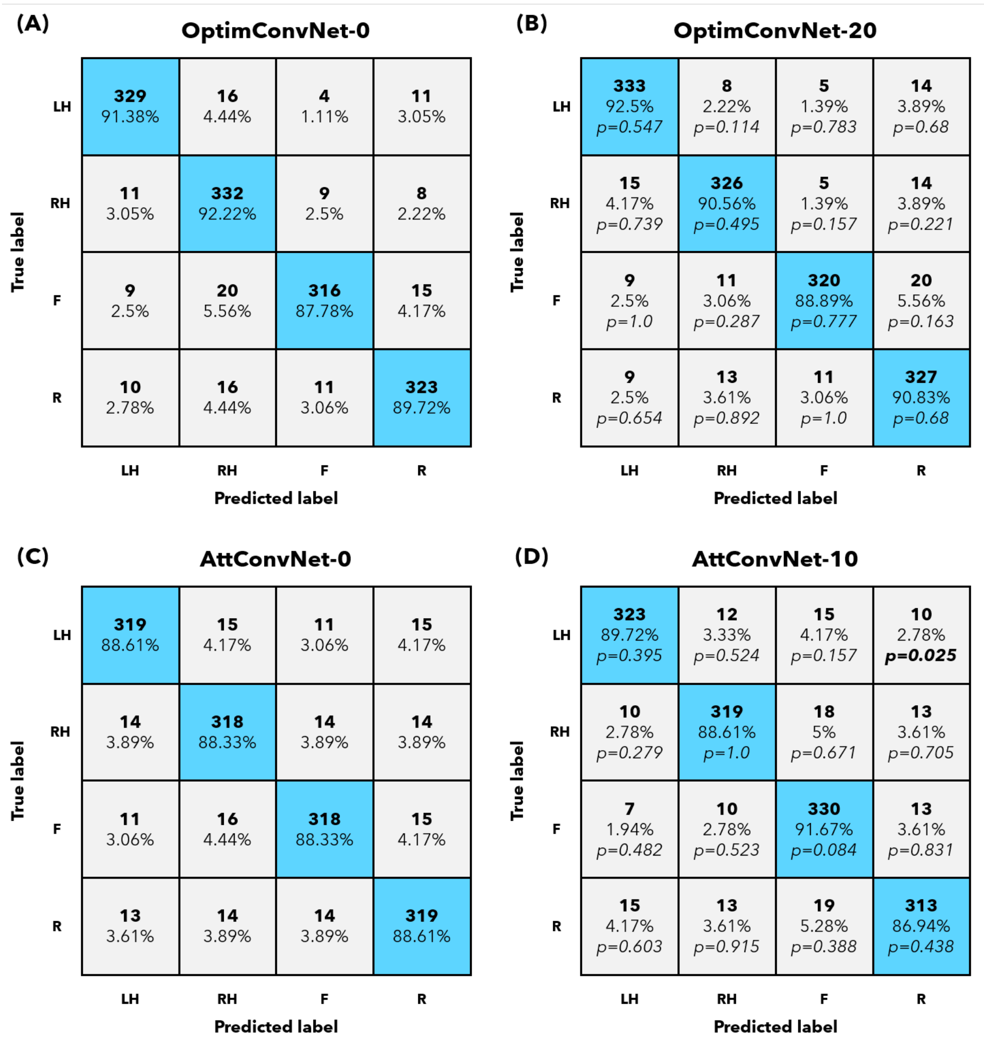

Figure 6: Confusion matrices for OptimConvNet-0 (A), OptimConvNet-20 (B), AttConvNet-0 (C), AttConvNet-10 (D). Matrices show at entry (i,j) the sum across subjects of predictions for class j when the true label is class i. Each entry also contains the proportion with respect to the total number of trials of the same class. (B) contains the p-values comparing entries between OptimConvNet-20 and OptimConvNet-0 (computed through Wilcoxon signed-rank test). (D) contains the p-values comparing entries between AttConvNet-10 and AttConvNet-0. Bold p-values indicate statistical significance ($p < 0.05$).

2.3.3. Implementational Details

The software and hardware used in our study are as follows:

- Programming language: *Python 3.8.*
- Deep learning framework: *PyTorch.*
- Preprocessing library: *braindecode* and *MNE.*
- Hardware: We trained our models in Google Colab, using an *NVIDIA Tesla T4 GPU.*

3. RESULTS

3.1. Learning Process

In order to verify that the models effectively learn to classify input EEG trials, we look at their learning process. Figure 4 displays the history of NLL losses throughout training epochs for OptimConvNet-20 and AttConvNet-10 trained on data from Subject 1. It also shows the history of decoding accuracies per model.

Both models show convergence to a minimum loss and a maximum accuracy during training. OptimConvNet-20 achieves a maximum training accuracy of 98.82%, while AttConvNet-10 achieves 96.86%. Validation loss and accuracy for OptimConvNet-20 closely follow the training values, thus showing the results of regularization in the network. During validation, OptimConvNet-20 achieves a maximum accuracy of 96.87%. For AttConvNet-10, we observe an irregular trend, both for NLL losses and decoding accuracy. Despite this, both values increase throughout the

Subject	OptimConvNet-0	OptimConvNet-20	AttConvNet-0	AttConvNet-10
1	88.125	91.25	85	87.5
2	88.75	92.5	87.5	87.5
3	92.5	91.875	87.5	86.875
4	90.625	92.5	90	90.625
5	85	92.5	89.375	90.625
6	92.5	88.125	88.125	91.25
7	91.25	86.875	89.375	87.5
8	92.5	91.25	92.5	93.125
9	91.25	89.375	86.875	88.125
Summary	90.28 ± 2.4	90.69 ± 1.96	88.47 ± 2.02	89.24 ± 2.08
p-value	0.859		0.159	

Table 1: Accuracies (%) per subject achieved by each model. For each model, a summary is provided as (mean±std) and p-values are reported(comparing OptimConvNet-0 vs. OptimConvNet-20 and AttConvNet-0 vs. AttConvNet-10).

epochs. AttConvNet-10 achieves a maximum validation accuracy of 95.31%.

In addition, we look at the MSE loss history for OptimConvNet-20 in Figure 5. The MSE loss optimized at every epoch (see Section 2.3.2) rapidly converges to a minimum (in about 10 epochs), where it stays until the end of training. When we look at the history of MSE losses optimized within the first layer of OptimConvNet-20 during a forward pass of test data, it can be observed that the learning curve does not converge in the initial epochs. However, in the final epoch, we can observe a trend similar to that observed in the history of MSE loss optimized per epoch.

3.2. Comparison

Figure 6 shows the confusion matrices for each of our models. At each position (i,j), we show the total number of test trials predicted as class j when the true label is class i. This number is computed as the sum across the 9 subjects we analyzed. Each entry also contains the percentage with respect to the total number of trials of the same class.

The matrix corresponding to OptimConvNet-20 shows, for each entry, the p-value obtained from a Wilcoxon signed-rank test performed between the entries for OptimConvNet-20 and OptimConvNet-0. That is, we compared two samples of entries, each containing 9 elements. The same information is displayed for the comparison between AttConvNet-0 and AttConvNet-10.

We found only one statistically significant difference between the entries ($p < 0.05$). Specifically, the entry (Left Hand, Rest), with $p = 0.025$. However, this entry corresponds to an incorrect classification.

Table 1 displays the accuracies obtained by each model for each of the 9 subjects analyzed. OptimConvNet-20 shows an accuracy across subjects of $90.69 \pm 1.96\%$ (mean±std), while AttConvNet-10 achieved $89.24 \pm 2.08\%$. However, no statistical significance was found for the difference in accuracies in any of the two routing mechanisms ($p = 0.859$ for routing through optimization and $p = 0.159$ for routing through attention).

4. DISCUSSION

The relevance of new ideas in deep learning research for the task of brain-signal decoding is yet to be fully explored. In this study, we introduced one of these ideas: routing mechanisms for convolutional neural networks. We designed two mechanisms, namely *routing through optimization* and *routing through attention*, with the purpose of decoding EEG trials from a motor execution task.

We presented a base CNN architecture, inspired by the trial log-variance calculation in FBCSP, that extracted temporal and spatial filters from its raw input. Two variants were derived from this architecture: OptimConvNet-20 implemented routing through optimization using 20 inner iterations; and AttConvNet-10 performed routing through attention using 10 inner iterations.

When looking at the learning process for each of these models, it is evident that the models converge to an optimal state, achieving at least 96% training accuracy and 95% validation accuracy. From these observations, we can conclude that the routing mechanisms are implemented effectively and they do what was intended for each. However, they do not say anything about the impact each routing mechanism has on EEG decoding accuracies.

In order to assess this impact, we compared each model with a baseline version performing zero inner iterations. We tested the statistical significance of the difference between entries in the confusion matrix for each routing mechanism and the decoding accuracies (both across 9 subjects). Our results show no statistically significant impact of the routing mechanisms described in this paper for the decoding task we tackled.

It is worth mentioning that a potential benefit of routing mechanisms *in general* for EEG decoding must not be discarded. Rather, further work could look into different mechanisms guided by domain-knowledge. Potential tweaks to the mechanisms here presented include a different choice of loss function for *routing through optimization* or a different way of quantifying agreement in *routing through attention*. Moreover, it can be beneficial to inspect the temporal and spatial filters learned by the network as described in [20, 19].

5. CONCLUSION

In conclusion, this bachelor thesis introduced the idea of routing mechanisms in convolutional neural networks for EEG decoding. We presented two mechanisms: routing through optimization and routing through attention. By using a CNN architecture specifically designed to handle EEG data, we derived two variants: OptimConvNet and AttConvNet, each implementing a different routing mechanism within the first layer. We analyzed their learning process and assessed the impact on decoding performance of each routing mechanism in a motor execution task. Our experimental results show that both routing mechanisms function effectively, with each model reaching at least 89% decoding accuracy. However, no statistically significant impact of the proposed routing mechanisms on EEG decoding accuracies was found. Future studies must investigate different routing mechanism designs that are guided by domain-knowledge of brain signals and inspect the effect of routing on feature representations.

References

[1] K. R. Müller *et al.*, “Machine learning techniques for brain-computer interfaces,” 2004. Biomed. Tech 49(1), pp. 11-22.

[2] K. R. Müller *et al.*, “Machine learning for real-time single-trial EEG-analysis: from brain–computer interfacing to mental state monitoring,” 2008. Journal of neuroscience methods 167(1), pp. 82-90.

[3] B. Blankertz *et al.*, “The Berlin brain-computer interface: Machine learning based detection of user specific brain states,” 2006. J. UCS 12(6), pp. 581-607.

[4] S. Silvoni *et al.*, “Brain-computer interface in stroke: a review of progress,” 2011. Clinical EEG and neuroscience 42(4), pp. 245-252.

[5] T. Chen *et al.*, “An EEG-based brain-computer interface for automatic sleep stage classification,” 2018. IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 1988-1991.

[6] D. J. McFarland and J. R. Wolpaw, “Brain-computer interface operation of robotic and prosthetic devices,” 2008. Computer 41(10), pp. 52-56.

[7] A. Bashashati *et al.*, “A survey of signal processing algorithms in brain–computer interfaces based on electrical brain signals,” 2007. J. Neural Eng. 4 R32.

[8] D. J. McFarland *et al.*, ““BCI meeting 2005-workshop on BCI signal processing: feature extraction and translation,” 2006. IEEE Transactions on Neural Systems and Rehabilitation Engineering 14(2), pp. 135-138.

[9] F. Lotte *et al.*, “A review of classification algorithms for EEG-based brain–computer interfaces,” 2007. J. Neural Eng. 4(2) R1.

[10] S. Gordon *et al.*, “Real world BCI: Cross-domain learning and practical applications,” 2017. Proceedings of the 2017 ACM Workshop on An Application-oriented Approach to BCI out of the Laboratory, pp. 25-28.

[11] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” 2015. Nature 521, pp. 436-444.

[12] A. Voulodimos *et al.*, “Deep learning for computer vision: A brief review,” 2018. Computational intelligence and neuroscience.

[13] Y. Guo *et al.*, “Deep learning for visual understanding: A review,” 2016. Neurocomputing 187, pp. 27-48.

[14] S. Wu *et al.*, “Deep learning in clinical natural language processing: a methodical review,” 2020. Journal of the American Medical Informatics Association 27(3), pp. 457-470.

[15] S. A. Fahad and A. E. Yahya, "Inflectional review of deep learning on natural language processing," 2018. International Conference on Smart Computing and Electronic Enterprise (ICSCEE), pp. 1-4.

[16] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," 2017. Communications of the ACM 60, pp. 84–90.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. Proceedings of the International Conference on Learning Representations (ICLR).

[18] K. He *et al.*, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," 2015. Proceedings of the IEEE international conference on computer vision, pp. 1026-1034.

[19] R. Schirrmeister *et al.*, "Deep learning with convolutional neural networks for EEG decoding and visualization," 2017. Human Brain Mapping 38, pp. 5391-5420.

[20] V. Lawhern *et al.*, "EEGNet: A compact convolutional neural network for EEG-based brain–computer interfaces," 2018. J. Neural Eng. 15 056013.

[21] M. Riyad, M. Khalil, and A. Adib, "MI-EEGNET: A novel convolutional neural network for motor imagery classification," 2021. Journal of Neuroscience Methods 353, 109037.

[22] H. Yang, J. Han, and K. Min, "A multi-column CNN model for emotion recognition from EEG signals," 2019. Sensors 19(21) p. 4736.

[23] M. Zhou *et al.*, "Epileptic seizure detection based on EEG signals and CNN," 2018. Frontiers in neuroinformatics 12, p. 95.

[24] X. Wei *et al.*, "Automatic seizure detection using three-dimensional CNN based on multi-channel EEG," 2018. BMC medical informatics and decision making 18(5), pp. 71-80.

[25] K. Ang *et al.*, "Filter bank common spatial pattern (FBCSP) in brain-computer interface," 2008. IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 2390-2397.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015. Proceedings of the 32nd International Conference on Machine Learning, PMLR 37:448-456.

[27] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015. Neurocomputing 187, pp. 27-48.

[28] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," 2017. arXiv preprint arXiv:1710.09829.

[29] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," 2018. International conference on learning representations.

[30] Y. Sun, S. N. Ravi, and V. Singh, "Adaptive activation thresholding: Dynamic routing type behavior for interpretability in convolutional neural networks," 2019. Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4938-4947.

[31] K. Sun *et al.*, "DRCNN: Dynamic routing convolutional neural network for multi-view 3D object recognition," 2020. IEEE Transactions on Image Processing 30, pp. 868-877.

[32] A. Vaswani *et al.*, "Attention is all you need," 2017. arXiv preprint arXiv:1706.03762.

[33] J. Gu *et al.*, "Recent advances in convolutional neural networks," 2018. Pattern Recognition 77, pp. 354-377.

[34] Y. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," 2010. Proceedings of the 27th international conference on machine learning (ICML-10), pp. 111–118.

[35] M. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," 2014. Proceedings of the European Conference on Computer Vision (ECCV), pp. 818–833.

[36] Y. LeCun *et al.*, "Efficient backprop," 2012. Neural Networks: Tricks of the Trade - Second Edition, pp. 9–48.

[37] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. 3rd International Conference for Learning Representations.

[38] R. Wijnhoven and P. de With, "Fast training of object detection using stochastic gradient descent," 2010. International Conference on Pattern Recognition (ICPR), pp. 424–427.

[39] J. Schmidhuber, "Deep learning in neural networks: An overview," 2015. Neural Networks 61, pp. 85-117.

[40] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," 2017. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1800-1807.

[41] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting," 2014. Journal of Machine Learning Research 15, pp. 1929-1958.

[42] P. Sotnikov, "Selection of optimal frequency bands of the electroencephalogram signal in brain-computer interface," 2015. Science and Education of the Bauman MSTU.

[43] M. Pregenzer and G. Pfurtscheller, "Frequency component selection for an eeg-based brain to computer interface," 1999. IEEE Trans Rehabil Eng. 7(4), 413-9.

[44] A. Selvı *et al.*, "Selection of eligible channel and frequency components for visual evoked EEG applications," 2018. 26th Signal Processing and Communications Applications Conference (SIU), pp. 1-4.

[45] F. Wilcoxon, "Individual comparisons by ranking methods," 1945. Biometrics Bull 1:80–83.