Final Project

Classification of MEG Data

Alexander Reisach

31 May 2020

KEN4257: Deep Learning

Faculty of Science and Engineering
Dpt. of Data Science and Knowledge Engineering
Maastricht University
Netherlands

Contents

1 Introduction

This report forms part of the final project for the course KEN4257 Deep Learning at Maastricht University. In this piece we will apply deep learning methods to Magnetoencephalography (MEG) data. The analysis of MEG and electroencephalogram (EEG) data using deep learning methods is a highly active field of research and has led to great advancements at the intersection of Artificial Intelligence and neuroscience [1] [2] [3]. In a MEG, the magnetic field induced by the currents naturally occurring in the human brain are recorded for different brain areas. The resulting data sets form a time series of brain activity for different parts of the brain. The electrical current in the brain carries valuable information about the state and activity of the individual observed that can be used for diagnosis and other forms of classification. In this project, we aim to reconstruct the activity that was performed during the MEG recording using deep convolutional neural networks.

2 Data

The dataset used for this paper consists of two separate sections, one for intra-subject classification, and one for cross-subject classification. Both sections contain data obtained during MEG recordings of an individual that was performing one of the following tasks:

- Rest
- Motor Task
- Story / Mental Math Task
- Working Memory Task

All classes are equally distributed across our data set and the data is recorded at 2034 Hz. On a representational level, we are dealing with a multi-instance classification problem where there is a joint label for a sequence of discrete temporal observations, each of which consist of one data point per sensor.

2.1 Intra-Subject

The data for intra-subject classification contains 140 seconds of MEG recording for each of the tasks as training data and 35 seconds of MEG recording for each task as test data. All recordings are taken from the same subject.

2.2 Cross-Subject

The training data for cross-subject classification contains 140 seconds of MEG recordings for each of the tasks for two different subjects. The test data contains 70 seconds of MEG recordings for each task and are taken from a total of six other subjects.

3 Pre-Processing

3.1 Downsampling

The raw MEG data is sampled at a high frequency. For computational reasons we decide to downsample the data. Following common practice in the literature, we downsample to a frequency of approximately 200Hz. We use a Chebyshev filter during downsampling for anti-aliasing since the data is highly periodic.

3.2 Windowing

We only have data for a total of 9 subjects. In order to increase our data set we split each sequence into multiple windows and assign the bag label to each of them.

3.3 Normalization

Apart from the periodic components, many individual sensor series exhibit differences in long-term trends and levels. Following the long-standing assumption of stationarity for the human electroencephalogram [4], we choose to normalize the data. Rather than normalizing each sensor, we normalize on a per-window basis to preserve relative differences between the sensors.

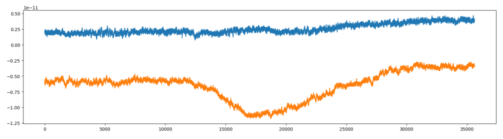


Figure 1: Example of different trends and levels across sensors

3.4 Memory Management

The density of data points makes handling the whole data set in memory a challenging task. We solve this problem by loading data sequentially, subsample before the windowing, and selecting only one percent of the resulting windows. At each pre-processing step, we only keep the processed version and free any memory allocated to raw data.

4 Model Choice

We are dealing with a multivariate time-series classification task. Classical modeling approaches for time series such autoregressive or recurrent neural network based methods would be suitable for this kind of data, however their state-based structure makes them less suitable for classification. Convolutional neural networks which have become the industry standard for representation learning and classification are seen as a more promising possibility. However, although our data is not structured in the sense of single interpretable values, we can assume Granger causality due to the strict time ordering of our observations. Furthermore, there is no inherent ordering to the sensor axis of the data. This means that typical methods for unstructured data such as two-dimensional convolutions may result in a great number of parameters that will remain unused at best and distort prediction at worst. For this reason we choose a neural network based on one dimensions convolutional layers. We will use these to find patterns within each sensor and use fully connected layers to combine the representations. Since we are dealing with a multi-class prediction problem, we apply a softmax to the class predictions and use cross-entropy loss for a loss function.

$$\sum_{c=1}^{M} = y_{i_c} \times log(p_{i_c})$$

where M is the number of classes, c is a binary indicator for belongingness to class c and p is a probability of observation i belonging to class c. To determine the accuracy of our models we choose the class with the highest probability after the softmax layer as classification for a given instance. For this model as for all models in the remainder of the report we use an *Adam* [5] optimizer.

5 Classification Results

5.1 Shallow Model

Our initial baseline model is structured as as follows:

```
simpleCNN(
  (cnn1): Conv1d(248, 16, kernel_size
      =(5,), stride=(2,))
  (fc1): Linear(in_features=7968,
      out_features=4, bias=True)
)
```

5.2 Intra-Subject

For intra-subject classification we choose 80% of the training data as cross validation set. We find that even our simple model finds a perfect fit for all, training cross-validation and test set almost instantly as can be seen in 4. Train, validation and test accuracy are all at 100%.

5.3 Cross-Subject

Unlike intra-subject classification, cross-subject classification requires external validity of the hypotheses fit by our classifier. For this reason, we decide against the usual 80-20 train and cross validation split, which would by necessity lead to intra-subject cross-validation. Rather, we decide to use the data provided as part of the intra-subject classification task for cross validation and train on 100% of the cross-subject training data. We use the same preprocessing pipeline as for 5.2, however this results in a greater overall training sample of 2209 windows of length 1000 due to the fact that now we have data for two subjects in our training set.

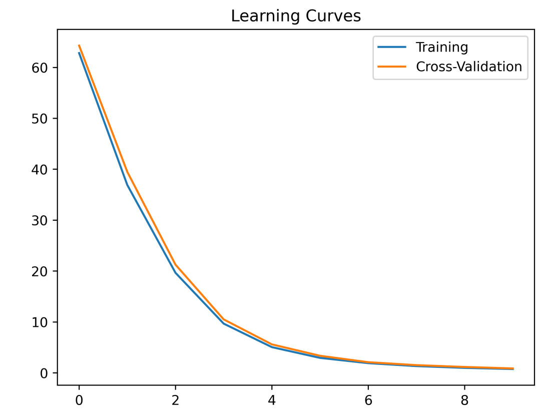


Figure 2: Learning curves of shallow baseline model on intra-subject classification

Our baseline model shows to quickly overfit on the training set as can be seen in 3. Nonetheless, the resulting accuracies as seen in 1 are far better than the 25% we would expect for random classification and should be good enough to correctly classify a longer sequence consisting of multiple windows. The learning curve in 3 shows the pattern initial reduction on both train and validation, until the model increasingly picks up patterns that are idiosyncratic to the training set and validation loss increases.

accuracy	subject
0.913295	test1
0.469667	test2
0.793319	test3

Table 1: Performance of our baseline model on cross-subject using a learning rate of 1e-3

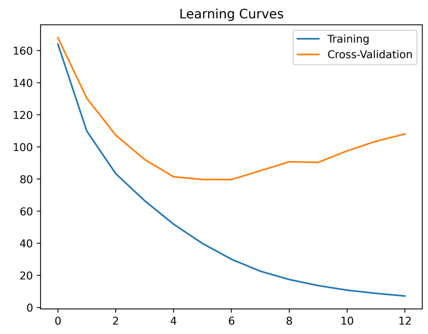


Figure 3: Learning curves and for our baseline model using a learning rate of 1e-3

5.4 Test Data Sets

Interestingly, it seems that in particular test set 2 is notoriously hard to fit. With no model do we achieve more than 65% accuracy and often this accuracy seems to come at the expense of accuracy on test 1 and 3. In particular, it seems that the tasks *task_story_math* and *task_working_memory* seem to be mislabeled frequently. The fact that this observation holds across all models and hyperparameters could point at inconsistencies in the data.

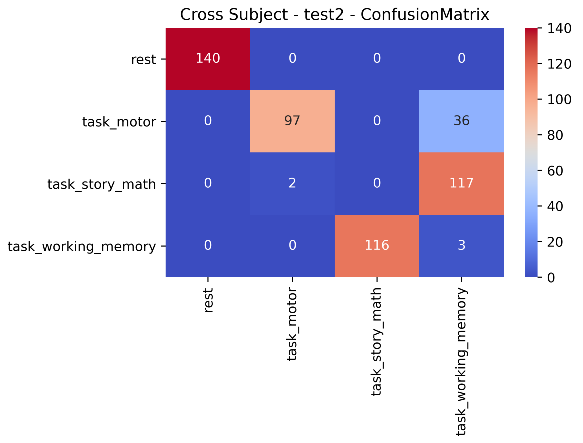


Figure 4: Class confusion matrix on test2

5.5 Extended Model

We also develop a more expressive model so we are better able to analyse the impact of batch normalization and Dropout. We only present the results of this model on the more difficult cross-classification task.

```
deepCNN(
  (cnn1): Conv1d(248, 128, kernel_size
      =(5,), stride=(2,))
  (cnn2): Conv1d(128, 64, kernel_size
      =(5,), stride=(2,))
  (cnn3): Conv1d(64, 32, kernel_size
      =(5,), stride=(2,))
  (fc1): Linear(in_features=3904,
      out_features=100, bias=True)
  (fc2): Linear(in_features=100,
      out_features=4, bias=True)
)
```

For this model we obtain the benchmarks in 2 on cross-classification. We observe that the extended baseline model achieves its optimal fit slower (20 epochs) and does not yet outperform our shallow model in 1.

accuracy	subject
0.905588	test1
0.534247	test2
0.772443	test3

Table 2: Performance of extended baseline model.

6 Hyper Parameters and Model Architecture

6.1 Baseline Model

Our model architecture contains as hyperparameters the number of layers, type and properties of the layers. We try to choose model hyperparameters on the basis of domain knowledge, literature review and empirical evaluation, in this order. As introduced in 4, we know from the nature of our data set that a combination of convolutions, fully connected layers and softmax seems suitable. As for the number of layers, we start with the most simple combination possible, namely one of each. In general, we start at a simple model and increase depth and expressivity as required. As for the specification for the convolutional layer, we want the model to be able to find a suitable representation of the oscillating recording of neural patterns. We do so by choosing a kernel size of 5 at a stride size of 2, which means that our kernels overlap with four other adjacent kernels by at least one observation, which helps detect changes. At the same time, we need to shrink our representation so we don't overparameterize our network when we feed it to the subsequent fully connected layer. We do so by restricting the number of filters to 16, which shrinks the dimensions from 248×1000 to 16×498.

6.2 Model Training

During model training we apply batch gradient descent on a batch size of 16. For each epoch, we monitor loss and accuracy for training and test set. We apply early stopping once the model does not reach a new best loss on the validation data set for ten consecutive epochs. In this case we use the model parameters which achieved the best loss on the validation set.

6.3 Extended Model

To measure the effect of dropout and batch normalization we choose a more complex model as introduced in 5.5. For this purpose we use the same design pattern as before but increase the depth and width of the network [6]. We place the batch normalization after the non-linearity as proposed by [7] and place dropout after batch normalization. We use stochastic dropout layers between fully connected layers and also between convolutional layers as investigated by [8], but do not apply them at the intersection of convolution and fully connected layers as we find this severely hurts performance. We choose lower dropout rates for the parts of our networks that are less likely to be overparameterized (visible layer and convolutional layer) and higher rates between fully connected layers.

7 Batch Normalization and Dropout

We study the effect of batch normalization and dropout primarily on the cross-subject classification task. This seems like the more fruitful area of study after even the simplest model could fit the intra-subject task perfectly.

7.1 Batch Normalization

To study the effect of batch normalization, we apply batch normalization after applying the non-linearity to each convolutional layer and between the fully connected layers. In figure 5 we can see that batch normalization results in a model that is able to reduce validation loss much further then our extended model baseline in figure 2. We hypothesize that this is the case as internal covariate shifts are prevented, the model is effectively equipped with an "equalizing" component that allows it to find multiple different hypotheses that all fit the training set well and result in marginal gains on the validation set.

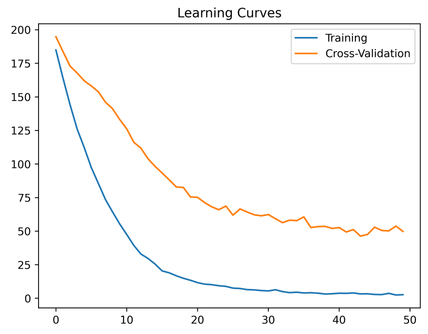


Figure 5: Learning curves of extended model using batch normalization

However, perhaps because the training set only consists of two different subjects, accuracies obtained by this model are not generally better than those without batch normalization. It seems that, although batch normalization allows us to train our model much further, these gains do not translate into superior classification accuracies. At this point it is important to note, that a lower loss does not necessarily result in better accuracies, since the gain could also consist of our model becoming more and more sure about the samples it gets right. It seems that batch normalization, by preventing the internal covariate shift, allows us to train a "better" model, but not one that is necessarily better for our purpose.

accuracy	subject
0.934489	test1
0.273973	test2
0.670146	test3

Table 3: Caption

7.2 Dropout

We use a dropout ratio of 0.2 for the visible layer, 0.2 between the convolutional layers and 0.5 between fully connected layers. In 6 we observe that our model performance on the validation set, even when it does no longer gain accuracy, never actually drops substantially unlike the previous models. This indicates that the hypotheses fit by the model with dropout are more generalizable than those of previous models. The results in table 4 confirm this theory. This model, while not competitive overall, has the most even distribution of accuracies across the test sets. Remarkably, it achieves by far the highest classification accuracy on test2. We conclude that the dropout as implemented for our model reduces overall performance somewhat but is highly effective in improving the gerneralization of the hypotheses fit. Note that we turn off dropout at evaluation time, which results in a huge increase in accuracy further corroborating this theory.

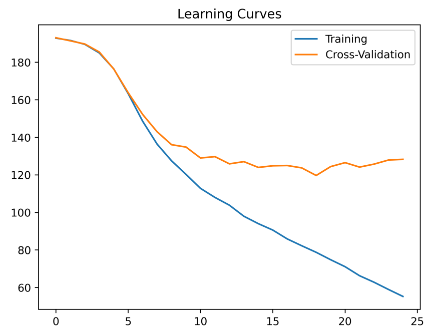


Figure 6: Learning curves of extended model using dropout

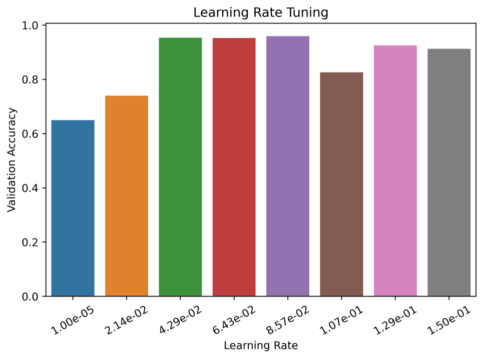
accuracy	subject
0.666667	test1
0.606654	test2
0.647182	test3

Table 4: Performance of our extended model with dropout.

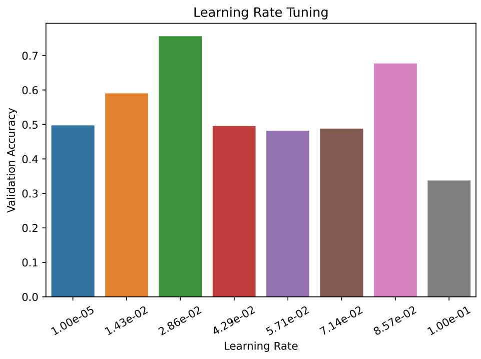
8 Learning Rate and Weight Initialization

8.1 Learning Rate

For this analysis we are using the model introduced in 5.1. All learning rates mentioned are passed to an Adam optimizer. On the intra-subject classification task, the model proves quite insensitive to the learning rate. In contrast, for the cross-subject classification task, we find that for low learning rates $< 1e-6$, our models tend to learn well but are too slow to be feasible. For learning rates in excess of $1e-2$, our models learn initially, but cross validation loss diverges quickly and the accuracies obtained are poor in comparison. We therefore conclude from the figures obtained in figure 7 that the range $[1e-3, 1e-2]$ most likely contains the optimal zone for the learning rates of our shallow model on cross-subject classification. For our deeper model from 5.5 the computational cost of a full grid search is prohibitively expensive, however we find that a learning rate of $1e-6$ works well empirically.



(a) Intra-subject classification

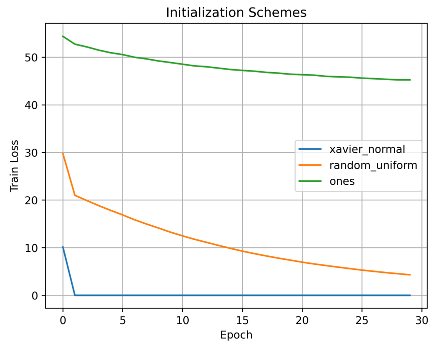


(b) Cross-subject classification

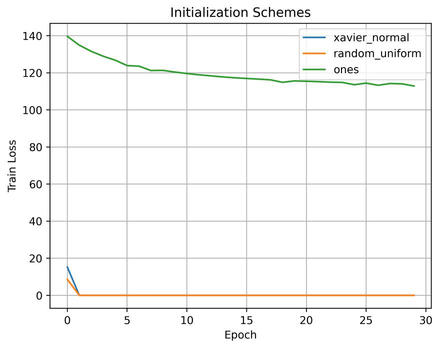
Figure 7: Effect of different learning rates on both classification tasks

8.2 Weight Initialization

We perform an analysis of different weight initialization schemes on our shallow model from section 5.1. In particular, we use the Xavier normal distribution initialization described in [9], a random uniform distribution in the range $-0.01 to 0.01$ and a naive initialization of setting each weight to one. In figure 8 we can see that for both, intra-subject and cross subject classification, Xavier uniform performs best, followed by random uniform. The initialization of ones is not only slow to learn, the plot makes it seem unlikely that a network initialized in this way would have the capacity to catch up with the other networks shown, even after extensive training. This observation is in line with the lottery ticket hypothesis as introduced in [10], which states that a network requires a certain set of suitably initialized neurons that will do the majority of the learning. If now, as in the case of "ones" in figure 8 we set all weights to one, there will be no chance for any lucky combination of random node weights that could be reinforced by backpropagation.



(a) Intra-subject classification



(b) Cross-subject classification

Figure 8: Effect of different initialisation schemes on both classification tasks

9 Alternative Approaches

We do find a substantial difference between training and test accuracies for all models trained on cross-subject classification. Since we observe a similar gap between training and cross-validation set, it remains possible that there are some relationships in our data that are helpful for generalization that our current models have not yet taken into account. In the previous sections we apply one-dimensional convolutions that are suitable for finding patterns in the time series created by the individual MEG sensors. However, we rely on fully connected layers, to combine the series from the different sensors. This approach does not yet take into account the possibility of patterns across sensors in the same way it does for patterns within one sensor. One way of approaching this, would be to keep using one-dimensional convolutions but flipping the channel and observation axes during a certain part of a forward pass of the network. This allows taking spatial periodicity into account while keeping the number of model parameters low. Another more common alternative is using two-dimensional convolutional layers on the grid of *n_channels* $\times$ *n_observation*. This results in a larger number of parameters but allows the network to fit a representation that takes both temporal and spatial components into account in the same step. Both methods introduce the ordering of channels as an additional parameter of variability. However, this aspect is alleviated once multiple stacked convolutions increase the receptive field.

9.1 Conv2D

For a CNN using 2D-convolutions, we use a total of three stacked layers, each of which is passed to a RELU non-linearity. The resulting representation is passed to two fully connected layers. Class predictions are determined by a softmax activation on top of the raw predictions.

```
CNN2D(
  (cnn1): Conv2d(1, 32, kernel_size
      =(7, 7), stride=(3, 3))
  (cnn2): Conv2d(32, 16, kernel_size
      =(7, 7), stride=(3, 3))
  (cnn3): Conv2d(16, 8, kernel_size
      =(7, 7), stride=(3, 3))
  (fc1): Linear(in_features=1960,
      out_features=100, bias=True)
  (fc2): Linear(in_features=100,
      out_features=4, bias=True)
)
```

We find that the convolutional neural network constructed in this way is easily able to fit the training data set. In fact, it seems that it can find a number of different representations that allow it to fit the training data set perfectly, so it can keep optimizing those for cross-validation. This results in the learning curve in 9 where the difference in learning and cross-validation loss is slowly increasing while both are steadily falling. From some point onward however, the model's cross-validation loss is starting to catch up with its test loss with both leveling out close to zero.

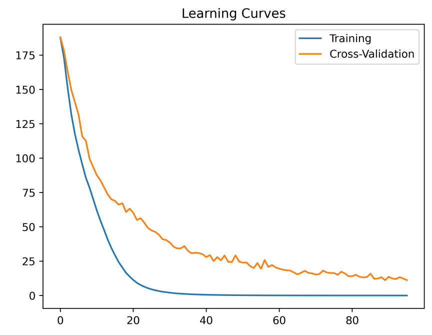


Figure 9: Conv2D learning curves

10 Conclusion

We find that a deep learning approach for classifications of MEG data is feasible. A simple network of 1D convolutions and fully connected layers can already classify intra-subject MEG windows with 100% accuracy. Even on the larger cross-subject classification task, this small model performs similarly to a more expressive scaled-up version. We find that for the scaled-up version dropout increases the external validity of our model predictions but does not improve overall accuracy. Depending the task specification the resulting balanced prediction accuracies may still be a desirable property. Batch normalization allows our model to find a fit with a lower loss, but this comes at the expense of accuracy on some test data sets. It is likely that both of these approaches would benefit from more and more diverse data. We further find that there is a Goldilocks zone of learning rates that are feasible for our model. Our model is able to learn from different random weight initializations as long as there is some chance for subsets of weights to have suitable initial combinations. Finally, we apply a network based on 2D convolutional layers to take into account more aspects of our data. It seems that 2D convolutions might scale up best to more extensive data sets. While their performance on test data does not exceed that of simpler models by a large margin, the model is clearly able to fit the training data set in a way that does not jeopardize performance on the cross-validation data set. Combining this approach with batch normalization and/or dropout should lead to a model that could make good use of a larger data set.

References

[1] P. Garg, E. Davenport, G. Murugesan, B. Wagner, C. Whitlow, J. Maldjian, and A. Montillo, "Automatic 1d convolutional neural network-based detection of artifacts in meg acquired without electrooculography or electrocardiography," in *2017 International Workshop on Pattern Recognition in Neuroimaging (PRNI)*, pp. 1–4, IEEE, 2017.

[2] I. Zubarev, R. Zetter, H.-L. Halme, and L. Parkkonen, "Adaptive neural network classifier for decoding meg signals," *Neuroimage*, vol. 197, pp. 425–434, 2019.

[3] A. Hasasneh, N. Kampel, P. Sripad, N. J. Shah, and J. Dammers, "Deep learning approach for automatic classification of ocular and cardiac artifacts in meg data," *Journal of Engineering*, vol. 2018, 2018.

[4] B. A. Cohen and A. Sances, "Stationarity of the human electroencephalogram," *Medical and Biological Engineering and Computing*, vol. 15, no. 5, pp. 513–518, 1977.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[6] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[7] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[8] S. Park and N. Kwak, "Analysis on the dropout effect in convolutional neural networks," in *Asian conference on computer vision*, pp. 189–204, Springer, 2016.

[9] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[10] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," *arXiv preprint arXiv:1803.03635*, 2018.