Deep Learning Mini-Project 1 Report
Prediction of Finger Movements from EEG Recordings

Group 78*
Caterina Bigoni, Luca Pegolotti and Nicolò Ripamonti

18 May 2018

1 Introduction

The goal of this project is to design and train a Deep Neural Network to analyse EEG data. In particular, we perform a two-class classification of EEG time signals collected from one patient to predict the direction of her upcoming finger movements (left vs. right). The dataset, part of the "BCI Competition II" [1], is composed of 416 recordings sampled with a 1000 Hz frequency for 28 channels (i.e., electrodes placed at different locations on the skull). The dataset is split in training and test datasets of 316 and 100 samples, respectively.

This report is organised as follows: we will briefly present and visualise the datasets in Section 2, where we introduce the suggested down-sampling, a possible data augmentation strategy and consider sample shuffling before splitting into train and test sets (si o no..?). Then, in Section 3, we will present five different models we tried and the corresponding results obtained on the test set. Of these, the one that performs best is selected for further tests in Section 4. In particular, we present a smoothing pre-processing strategy to reduce noise in the signals and an optimisation setup to select the hyper-parameters. Final conclusions and discussions are provided in Section 5.

2 Data Visualisation

Before rushing at adding extra layers or more neurons to increase the complexity of our model, we take a look at the given data. Figure 1 shows the average over all samples of the training and test datasets for three randomly chosen channels, i.e. channels 3, 10 and 25. The averages are shown separately for the two-classes, i.e. left and right, and for both the signal sampled at 1000 Hz (in red and blue) and its down-sampled signals (in black and magenta). We can observe that the down-sampled signals present a less noisy average with respect to the 1000 Hz signal. Moreover, we remark that the means over the test and train dataset are very different. This behaviour is observed, although with different intensity, for all 28 channels. One possible explanation of this discrepancy lies in the fact that the averages were not performed on equally numerous datasets: 316 samples and 100 samples in the training and test datasets, respectively. Indeed, we can confirm that the average obtained from a subset of the training dataset of 100 samples is quite different from the average over 316 samples shown in Figure 1. This behaviour can be justified if samples have a high variance and Figure 2,which shows the behaviour of three randomly chosen down-sampled samples for each class (i.e., left and right) for both the training and test dataset, confirmes it. Immediately, we can see that both the training and test selected samples are all very noisy and distant from their corresponding mean.

Even if we understand that the average is not a sufficient meaningful quantity to describe a complex dataset, the high variability of data makes us believe that this apparently simple two-class 1D classification problem, will instead be quite challenging. In order to reduce the discrepancies, we normalise

*As agreed with Dr. F. Fleuret, L. Pegolotti has collaborated with group 78 for Project 1 and with group 79, with M. Martin, for Project 2. C. Bigoni and N. Ripamonti have worked together on both projects.

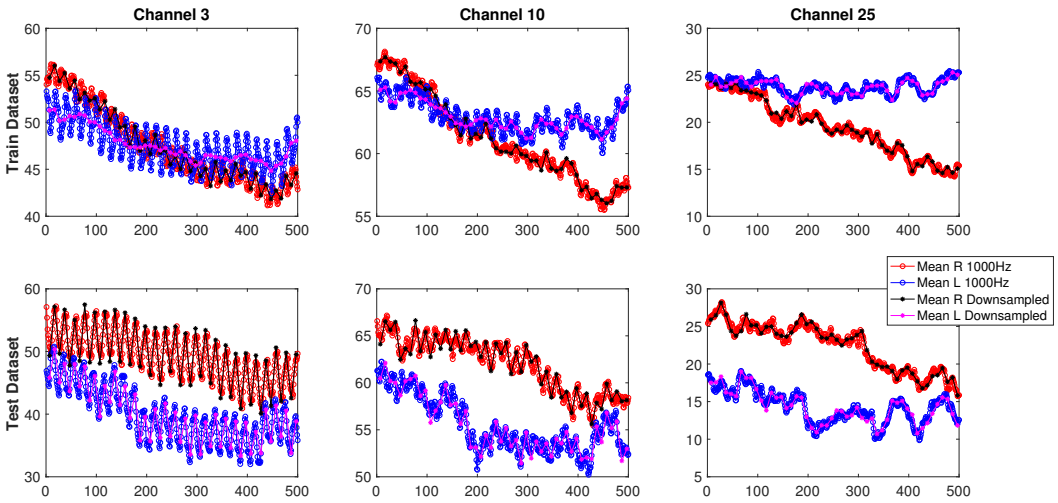


Figure 1: Comparison of training and test datasets averages per channel over 316 and 100 samples, respectively. Three channels (i.e., 3, 10 and 25) are randomly chosen, but the results generalise to all. Each plot shows the mean of the samples with output 0 (i.e., classified as "right") and those with output 1 (i.e., classified as "left"). The means of the signals sampled with 1000 Hz are plotted in red and blue, while those of the signals downsampled to 100 Hz ar plotted in black and magenta. For some channels (e.g., channel 3) the down-sampled averages present a lower variability than those for the signals sampled at 1000 Hz. In all cases, the averages of the training set are distinctively different from those of the test set.

the training dataset using its mean and standard deviation for all channels. The same values are used for the test dataset.

Finally, driven by the results already published for this dataset, known as dataset VI in the BCI competition [1], and supported by other studies (see e.g. [4]), we propose a preprocessing strategy to reduce both the generalisation error and the size of the model needed to fit the training set. Indeed, by reducing the noise in the data, i.e., by increasing the signal-to-noise ration, we aim at reducing the amount of variation that the model has to take into account [3] and thus . A *filtering preprocessing* technique is therefore applied to our final model, as explained in Section 4. In particular, we use the 1D Savitzky-Golay filter [2] to smooth the data without greatly distorting the signal. By a convolution, the filter fits successive subsets of adjacent data points in a low-degree polynomial using the linear least squares method. In our tests, we use subsets of 7 points to be fitted in a polynomial of order 3. Figure **??** shows ....TODO

3 Models we considered

In this Section, we describe some of the models we tested on the EEG signals and summarize their performance in terms of errors on train and test datasets. Each of these models is tested against the standard down-sampled dataset and an *augmented dataset*. The latter is obtained by preprocesing the dataset to reduce the generalisation error [3]. In particular, we exploit the 1000 Hz dataset to extract not one, but 10 signals with frequency 100 Hz. In this way, our training dataset has the following dimensions: 3160 samples, 28 channels, 50 time-steps. We considered the possibility of extending the data-augmentation strategy to the test dataset to further increase the dimension of our training dataset by including some samples from the test dataset after shuffling. However, we soon realised that this would imply solving an oversimplified problem being the 10 down-sampled signals all very alike. We therefore apply the data-augmentation strategy only to the training dataset.

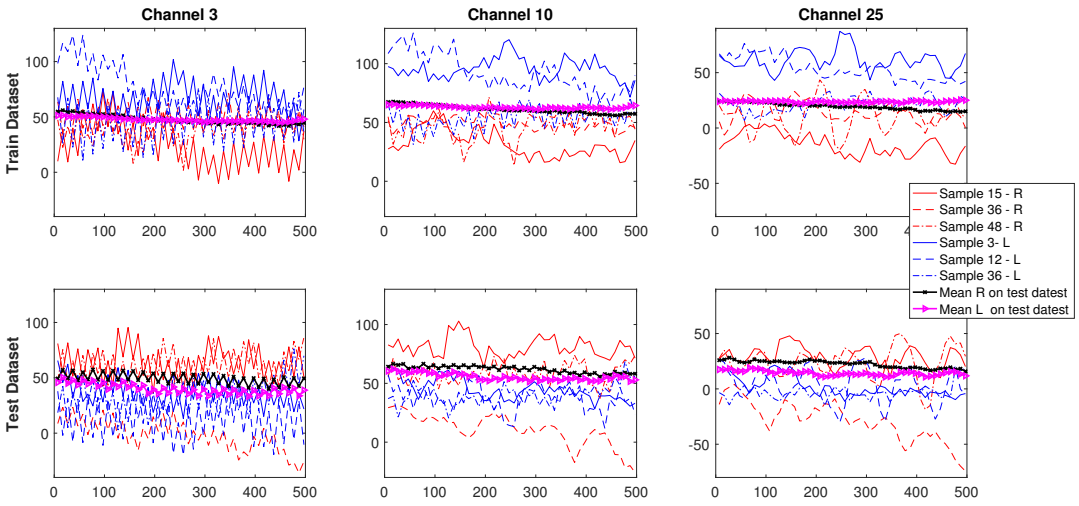
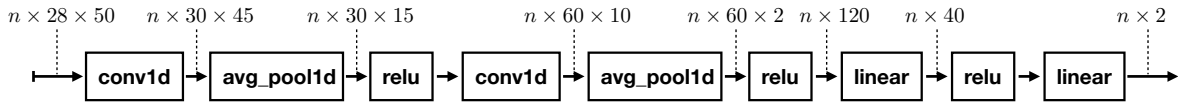


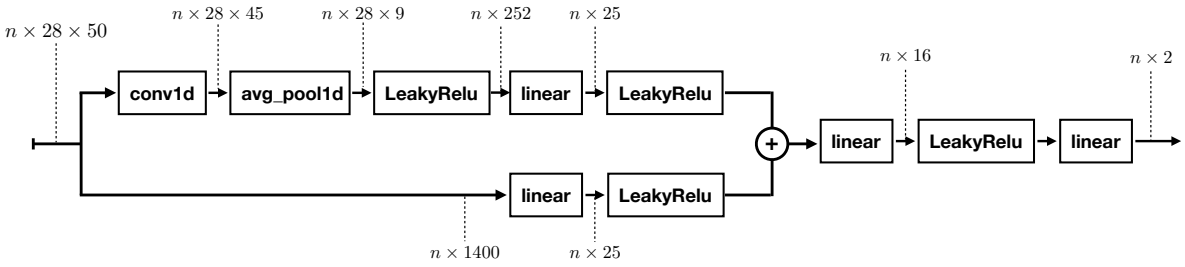
Figure 2: Plot of few randomly chosen samples from training and test datasets for channels 3, 10 and 25. Samples classified as 0 (i.e., "right") are plotted in red, while samples classified as 1 (i.e., "left") are plotted in blue. The down-sampled averages corresponding to the training and test dataset are also shown in black. These plots confirm that the data have a high variability.

In the following, we provide a small description of each model we considered. The diagrams representing the graph structures show the dimensions of the input and output tensor of each module composing the networks except for the non-linear nodes, which do not modify the dimensionality of the inputs; n is the number of samples of the input.

- Linear perceptron (M1): the simplest model one could devise for a classification problem is the linear perceptron: given an input tensor x, the output is computed by the model as $y_{ij} = \sum_k x_{ik} A_{jk} + b_j$, where A is the matrix of weights, and b is the bias vector.
- Simple convolutional linear network (M2):



- Multichannel deep convolutional neural network + linear perceptron (M3):



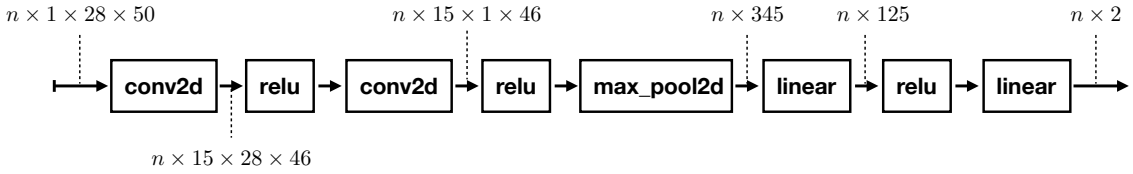
The convolutional layer in this model is such that that different masks are applied to each channel; at the implementation level, this is achieved by setting the `groups` parameter equal to

	non-augmented dataset				augmented dataset			
Trial	M1	M2	M3	M4	M1	M2	M3	M4
1	27	34	32	32	23	37	36	35
2	27	29	33	25	24	32	36	37
3	29	28	30	24	24	33	35	34
4	30	23	29	29	24	34	37	34
5	26	28	30	33	26	30	34	33
6	27	29	32	26	26	33	40	33
7	26	29	33	29	25	31	34	32
8	27	27	29	21	24	32	44	29
9	29	31	30	29	24	32	39	37
10	26	32	31	22	24	32	36	35
Best	26	23	28	21	23	28	34	29
Average	27.4	29.0	30.9	27.0	24.4	32.6	37.1	32.9

Table 1: Test errors for the linear perceptron (M1), the simple convolutional neural network (M2), the multichannel deep convolutional neural network combined with the linear perceptron (M3) and the shallow convolutional neural network (M4).

28 in the constructor of the model. Convolutions of this kind are employed to study multichannel uni-dimensional time-series also in [5].

- Shallow convolutional neural network (M4):



In this network, the first and the second convolutions operate in the dimension of the time samples and the channels respectively. We took inspiration from [4] for the design of this model.

The models are tested on the standard downsampled and augmented datasets to evaluate their performance. We use the stochastic gradient descent for the optimization step, with bash size equal to 100 or to 1000 when the train set is composed by 316 or 3160 samples respectively. The learning rates are 10^{-1} for the linear perceptron and 10^{-3} for the other three models.

Table 3 shows the performance of the four models in terms of error on the test dataset after 1000 iterations of the stochastic gradient descent. Surprisingly, the linear perceptron (M1) performs better than (M2) and (M3), leading to a smaller average error over all the runs and also to a smaller variance of the result. In the non-agumented dataset, however, the model that ensures in our experience a smaller minimum error and a smaller error on average is the shallow convolutional neural network.

Augmenting the dataset leads to a slightly smaller error for M1, but to a considerable loss of accuracy for the other three models. It is likely that, with a different choice of parameters (size of convolutional and pool nodes, kernels ...), the advantage of using larger datasets could be exploited also for M2, M3 and M4.

Since the shallow convolutional network (M4) showed the most promising reults in these test runs, in the following Section we decide to focus on optimizing this particular network. It is important to notice that our findings are greatly determined by the choices of parameters for each of the networks we considered, and it is possible that other combinations for M2 and M3 could lead to smaller errors.

4 Model that works best

it doesnt overfit + graph

– Optimization with Hyperparameters

5 Conclusion

References

[1] BCI Competition II. `http://www.bbci.de/competition/ii/`.

[2] Savitzky-Golay Filter. `https://en.wikipedia.org/wiki/Savitzky-Golay_filter`.

[3] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[4] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter, K. Eggensperger, M. Tangermann, F. Hutter, W. Burgard, and T. Ball. Deep learning with convolutional neural networks for brain mapping and decoding of movement-related information from the human eeg. *arXiv preprint arXiv:1703.05051*, 2017.

[5] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao. Time series classification using multi-channels deep convolutional neural networks. In *International Conference on Web-Age Information Management*, pages 298–310. Springer, 2014.