Deep Learning - Project 1

Niccolò Sacchi, Succa Riccardo & Marco Zoveralli
École polytechnique fédérale de Lausanne

May 12, 2018

***Abstract*—The goal of this project is to train and test a predictor of finger movements (left or right hand) from Electroencephalography (EEG) recordings. We decided to construct several solutions, starting from simple baseline models and then escalating to complex deep neural networks.**

I. Introduction

This project consists in a two-class classification problem to predict the laterality of incoming finger movements 130ms before key-press. The electical pulse that generates the finger movement starts a fraction of second before the actual movement, giving us the possibiliy to predict which hand the subject is going to move through an EEG analisys.

The following sections give an overview of the techniques that were tried in order to construct a predictor. Section II breafly describes the dataset of EEG recondings and the experiment that was perfomed to obtain them; section III proposes a first approach to the problem by using a series of simple classification models (baselines); section IV goes to more complex solutions exploiting Convolutional Neural Networks and Residual Neural Networks; section V compares the results obtained with the different models.

II. Dataset

The dataset was provided by Fraunhofer-FIRST, Intelligent Data Analysis Group (Klaus-Robert Mller), and Freie Universitt Berlin, Department of Neurology, Neurophysics Group. It was first given in the "BCI competition II" (May 2003) with a sample of 416 epochs, divided in 316 train records (labeled) and 100 test records (unlabeled). The experiment perfomed to construct the dataset consisted in examining a normal subject sitting on a chair in a relaxed position, while he was typing on a computer keyboard in a self-chosen order. Three sessions of six minutes were taken, all conducted in the same day with a small break inbetween and with an average typing speed of 1 key/second. The EEG records were made by using 28 electodes to monitor the brain electrical activity. They measured a time slot of 500ms, ending 130ms before the keypress, and sampled at 100Hz and 1000Hz. In our project we used the records sampled at 100Hz, imported as a 28 (elecrode's measurements) x 50 (time frames) matrices. An example of a record with the respective label is shown in figure 1.

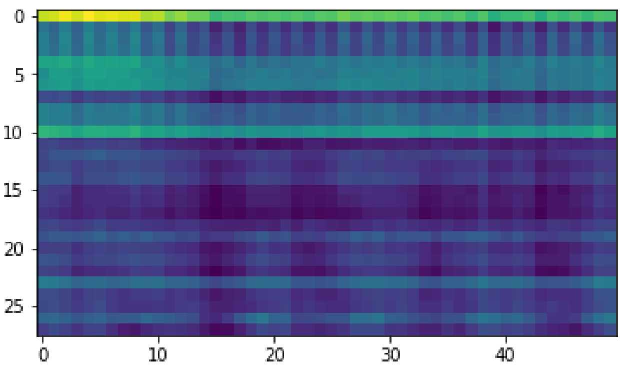

Figure 1: Sample from the dataset with label 1 (right movement)

III. Baseline Models

All the baseline models were tested with cross-validation and we iterated over a range of values for some parameters in order to find the best score. We implemented the following models (from sklearn library):

- Logistic Regression, tested with a grid search on the regularization parameter. Surprisingly, this model has good performance in our classification problem as shown in table I, despite the fact that it is much more simple than other models we tried.
- Random Forest Classifier, with iteration over the max depth of the trees.
- K-Nearest Neighbors, with multiple values for K. Moreover, before trainig this model, we normalized the input and applied a PCA to reduce the dimensions (we kept the 95% of the signal energy).

Baseline	Test Accuracy
LogisticRegression	0.7
Random Forest Classifier	0.58
K-NN	0.54

Table I: Baselines results

Table I clearly shows that none of the well-known classifiers is really able to provide a proper solution to this problem. In the next section we show the improvements that were obtained through the adoption of our own models and the methodology behind their construction.

IV. Deep Neural Networks

Our methodology was based, at first, on building networks that were inspired to the main concepts introduced during the lectures. Then, we focused on modifying these models with the goal of improving the results.
One fundamental aspect of our approach is to perform a grid search over a set of parameters of the network and using a cross-validation for each combination. The cross-validation technique consists, as first step, in splitting the available train dataset in k equal parts. Then, k-1 of these sets are used to train the model and 1 is used as a validation dataset. This process is repeated k times: at each iteration, a different part is considered as validation dataset and the remaining k-1 parts are taken as training dataset. This procedure must be repeated for a set of possible values that this parameter can assume. Finally, the parameter value that lead to the best prediction score can be considered to be the optimal one.
Two further issues must be taken into account. First, there are a lot of parameters that influence the score. We took into account the most relevant: the number of layers, the regularization terms, the type of activation functions (i.e. ReLU, Tanh, ELU) and optimizers (i.e. Adam, Adamax, Adadelta), and the number of hidden units in the last fully connected layer. However, other relevant parameters are not considered during this optimization phase. For instance, we are not iterating through the number of filters, the filter dimensions, the learning rate and we fixed the basic skeleton of the networks.
Second, getting the optimal combination of parameters implies enumerating all the possible combinations of all the parameters. This results in exponential complexity and, after some tries, we observed that this is unfeasible for our available computational resources. Therefore, we adopted an approximation of the grid search method, which consisted in a greedy approach where we selected the optimal parameters singularly. Although this does not provide an optimal solution, it allows to keep the algorithm complexity at an acceptable level (linear). With this approach, we have to carefully choose the parameter's order. First, we search over the parameters that mostly influence the model strucure, in particular the number of layers, the number of hidden units and the activation function. Then, the grid search moves to the optimizer's parameters (optimization function and weight decay). Finally, we optimize the dropout parameter. After all the parameters are defined, we decided to perform again a search over the number of layers, because its best value may change after the other parameters are set.
The table II summarizes the grid search results for all the networks.

A. CNN: 2D Convolutional Layers

The first model that we tried is a 2D Convolutional Neural Network. After a grid search on the parameters, we obtained the structure in figure 2: it consists in 5 convolutional layers, ReLU as activation function and Adadelta as optimizer.

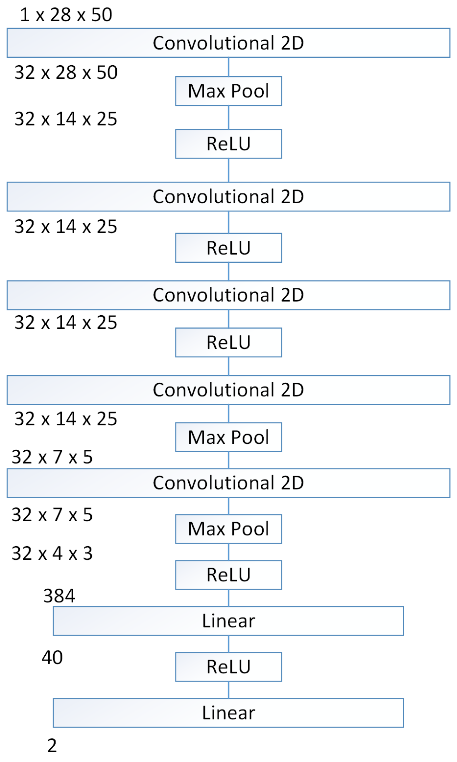

Figure 2: 2D Convolutional Neural Network

This network uses 2D convolutional layers, because at first it seemed natural to treat the provided dataset as a set of images. Therefore, the network was trying to find patterns between close pixels, by applying 2D filters over the input. This approach makes sense if there is a spatial relationship among the features. In our case, as mentioned in II, the input sample consists of 28 signals (EEG channels) captured over different time frames. Since the positions of the channels are not known, we cannot assume any spatial relationship between them. Therefore, applying a 2D filter over this dataset might try to find patterns between channels that are not spatially correlated. Consequently, we decided to process them all together by applying a 1D filter instead of a 2D.

B. CNN: 1D Convolutional Layers

For the 1D CNN, we started with a simple structure and then we increased the complexity by adding more modules.
In the first CNN, we only included dropout layers. Then, we used Batch Normalization: this technique provides significant improvements in terms of test accuracy. It focuses on performing normalization at each layer, and

Network	# Additional Layers	Hidden Units	Optimizer	Activation Function	Regularization term	Dropout
2D CNN	3	40	Adadelta	ReLU	0.0	0.0
1D CNN	4	40	Adamax	ELU	0.0025	0.1
1D CNN + Batch Norm	5	160	Adamax	ELU	0.0025	0.3
1D CNN + Batch Norm + Dilation	1	40	Adamax	ELU	0.135	0.1
Residual 1D CNN	6	120	Adamax	Tanh	0.007	0.2

Table II: Network Architectures

one of the main benefits is the mitigation of the change of the input distribution that may occur across the network layers. Finally, we added a dilation in the convolutional layers, hoping that increasing the receptive view of the filters could bring an improvement.
Among these three variants, the best results were obtained by the 1D CNN with Batch Normalization (see Section V for the details). The structure is shown in figure 3.

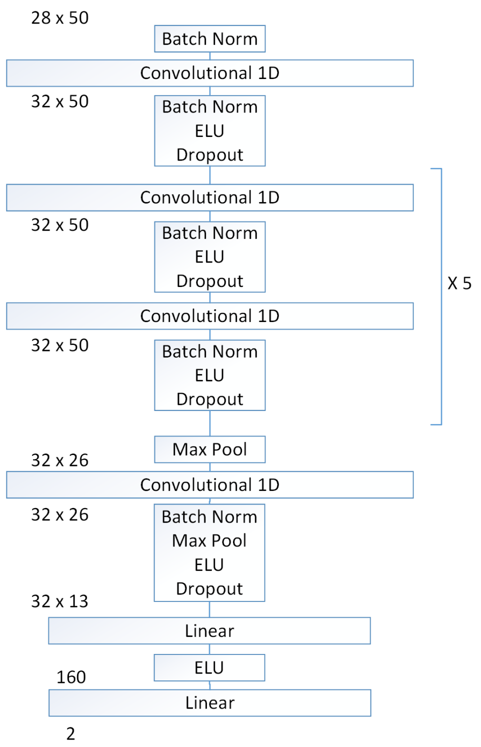

Figure 3: 1D Convolutional Neural Network with Batch Normalization

One remark that concerns all the networks that involve batch normalization: although some sources[1] claim that batch normalization makes the dropout advatanges negligible, we decided to try to tune both parameters during the grid search in any case.

[1]Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift - Sergey Ioffe, Christian Szegedy

C. Residual 1D CNN

This model consists in a series of aggregated residual blocks. Each residual block is made of a predefined number of convolutional layers, but the peculiarity of the block is that the input is also propagated directly to the output. Then, a group of residual blocks (two by default) are put together to construct an aggregated block. The grid search for the model iterated on the number of aggregated blocks. Figure 4 shows the final structure.

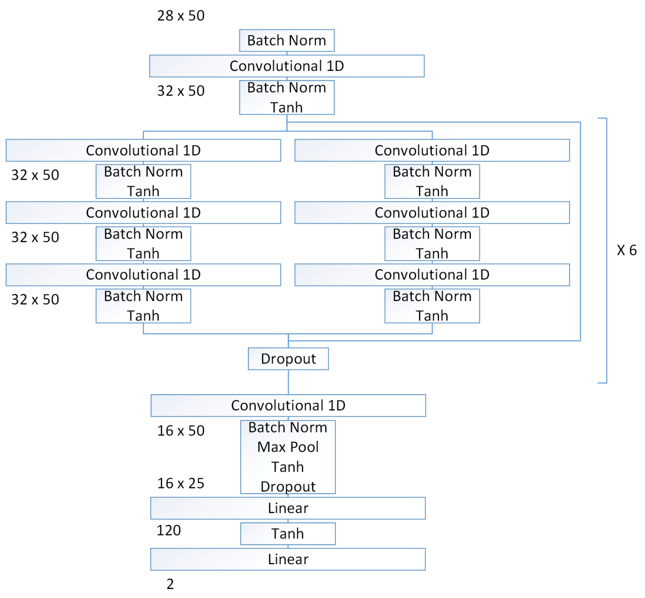

Figure 4: 1D Residual Neural Network

Even if residual CNN are mainly used in the field of computer vision, we decided to try this type of network with our dataset. As we will show in section V, although the complexity of the model is much higher than the previous 1D and 2D CNN, the results did not improve in equal measure.

V. Results

We now present the results of the networks that we described in the previous section. After each model was defined with a grid search over the parameters, we tested them with a cross-validation on the train set and with a

final evaluation over the test set.
Figure 7 shows the results of the cross-validation.

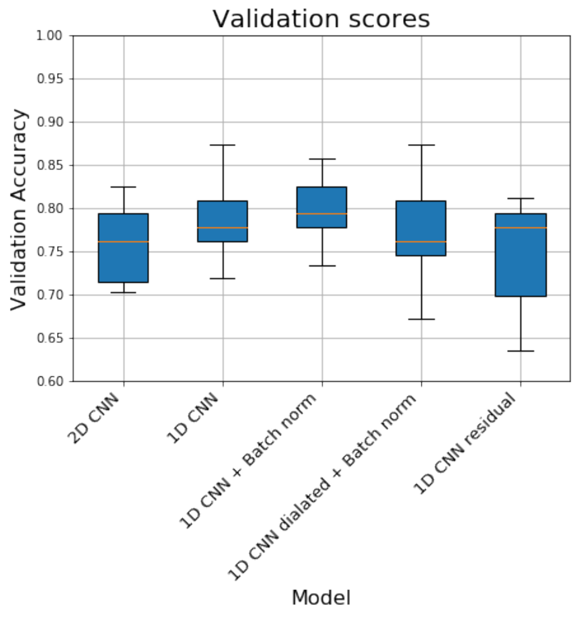

Figure 5: Result of the Cross Validation on the train set

On the test set, the results are the following:

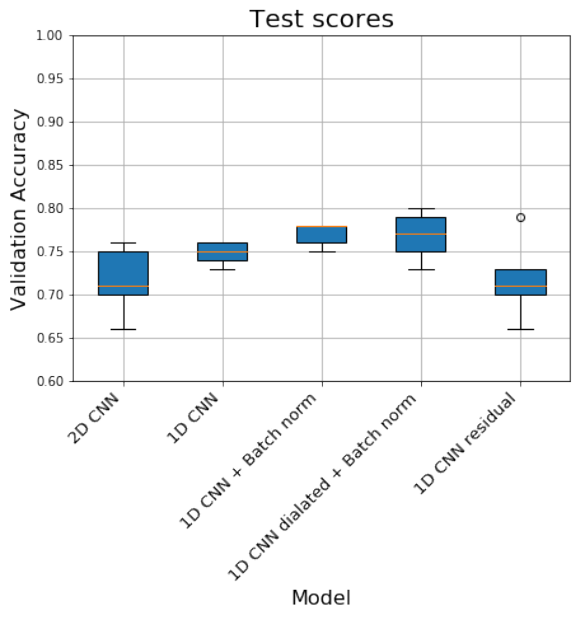

Figure 6: Result of several run on the test set

Network	Cross Validation Accuracy	Test Accuracy
2D CNN	0.76 ± 0.046	0.716 ± 0.036
1D CNN	0.79 ± 0.051	0.748 ± 0.012
1D CNN + Batch Norm	0.80 ± 0.041	0.770 ± 0.013
1D CNN + Dial + Batch Norm	0.77 ± 0.067	0.768 ± 0.026
1D CNN Residual	0.74 ± 0.067	0.718 ± 0.043

Table III: Accuracy Table

We noticed that the results on the test set are quite unstable. This is due to the fact that the cardinality of the test set is low (only 100 samples) and a misprediction on a sample can substantially vary the score. To overcome this problem, we decided to run the test several times and we reported the means and standard deviations of the results. In order to avoid overfitting, we had to choose an optimal number of epochs, i.e. the number for which the validation loss during the cross-validation is minimized. Then this number of epochs is used for the test.
From the boxplots above, the 1D Convolutional with Batch Normalization seems to be the best one, with an accuracy mean slightly better than the others. For this model we show how the loss decreases with the three different optimizers that we used:

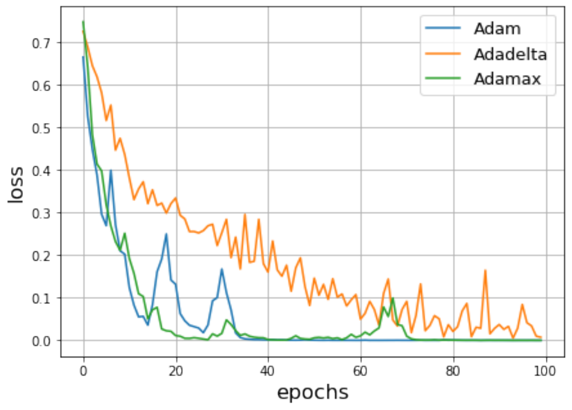

Figure 7: Loss function for different optimizers

The grid search chose the best one, Adamax in this case.

VI. Conclusion

The results that we obtained are partially satisfying. On one hand, the test accuracy is not excellent. The reasons can be found in the limited size of the train and test set and in the fact that no preprocessing was performed. Understanding the nature of the input signal was beyond the goal of this project, but an accurate analysis on the EEG dataset could have led to better results.
Still, the methodology that we followed allowed us to improve the results at each step. The limitation of the computational power forced us to apply an approximation

of the grid search technique. It can be argued that an extensive search on the parameters could have given an higher accuracy for all the models.