Deep Learning - Project 1

NICCOLÒ SACCHI, SUCCA RICCARDO & MARCO ZOVERALLI
GROUP: RoadSegmentationFault
École polytechnique fédérale de Lausanne

May 11, 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 **??** proposes a first approach to the problem by using a series of simple classification models (baselines); Section IV goes to more complex solutions exploiting Fully Connected Neural Networks and Convolutional Neural Networks; Section V compares the results obtained with the most performing 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. 3 sessions of 6 minutes each 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. We also tested the 1000Hz records after we identified the best predictor model. An example of a record with the respective label is shown in figure **??**

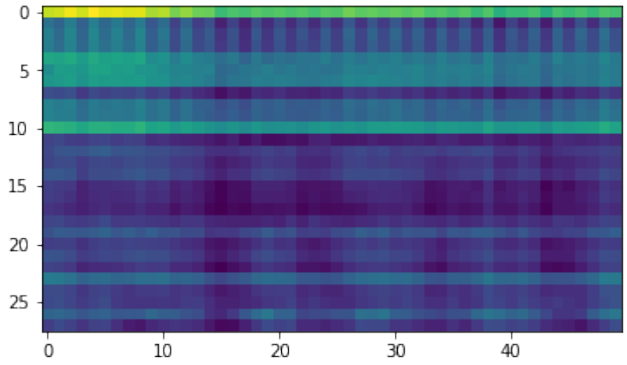

Figure 1: Sample from the dataset with label 1 (right movement)

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 figure **??**, 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)

The above table clearly shows that none among 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

Table I: Baselines results

Baseline	Test Accuracy
LogisticRegression	res1
Random Forest Classifier	res2
K-NN	res3

our own model and the methodology behind its construction.

IV. Deep Neural Networks

Our methodology was based, at first, on building models that were inspired to the main concepts that were introduced during the lectures. Then, we focused on modifying these models with the goal of getting better results.
One fundamental step of this approach consisted in adopting the cross-validation technique in order to extract the optimal parameters for each model. Given a certain parameter to optimize, 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 considered as training dataset. This procedure must be repeated for a set of possible values that this parameter can assume. Finally, the parameter values that lead to the best prediction scores can be considered to be the optimal ones. Two further considerations must be taken into account:

- There are a lot of parameters that influence the score. We considered some relevant ones: the number of layers, the regularization terms, the type of activation functions and optimizers, and the number of hidden units in the last layer. However, other relevant parameters are not considered during this optimization phase. For instance, we are not considering the number of filters, the filter dimensions, and the basic skeleton of the network is fixed.
- 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 this methodology, which consisted in a greed approach in which we selected the optimal parameters singularly. Although this does not provide an optimal solution, it still provides some improvement and, more importantly, it allows to keep the algorithm complexity at an acceptable level (linear).

A. CNN: 2D Convolutional Layers

This network is made of three convolutional layers and two fully connected layers:–da cambiare quando abbiamo i risultati

This was the first network that was used. It considers convolutional layers as 2D because at first it seemed natural to consider 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 in 28 signals (EEG channels) captured over different time frames. Since the position of the channels is 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 consider them all together by applying a 1D filter instead of a 2D.

B. 1D CNN + Batch Normalization

This network is made of five convolutional layers and two fully connected layers:

It can be seen that at each layer the batch normalization is applied. This technique provides significant improvements in terms of test accuracy (X vs Y). Batch normalization focuses on performing normalization at each layer, and one of the main benefits is the mitigation of the change of the input distribution that may occur across the network layers.
One remark that concerns all the networks that involve batch normalization: although several sources claim that batch normalization make the dropout advatanges negligible, we decided to try to tune that parameter during the cross-validation in any case.

C. 1D CNN + Batch Normalization + Residual

This network contains a set of convolutional layers, a set of linear layers, and a set of residual layers. Residual layers consist in having a block of convolutional layers (replicated a certain amount of times) whose output (at the end of the block) is influenced by the input of the block too. Our model accepts an arbitrary number of such blocks (each block has three convolutional layers), but by default there are two of them.

Table II: Network Architectures

Network	# Hidden Units	Optimizer	Activation Function	Regularization term	Dropout
CNN: 2D Convolutional Layers	res1	asd	lol	xd	wtf
1D CNN + Batch Normalization	res2	asd	lol	xd	wtf
1D CNN + Batch Normalization + Residual	res3	asd	lol	xd	wtf

V. Results

The traditional CNN was trained for 1000 epochs, each consisting of 25 batches of 4 images, on a NVIDIA GeForce GTX graphics card with 2 GB of VRAM, obtaining an F1 score of 0.833. While the U-Net was trained for 10000 iteration with a learning rate $\lambda = 10^{-3}$ which lead to an F1 score of 0.9385 on the test set submitted on Kaggle. To compare different techniques we used the Precision Recall method (PR) which focuses on the relevance of the prediction on the labels of a test set. To compute the PR values we split the training dataset into two subsets, one used for training and one used to compute the PR curve. The latter is computed by gradually increasing the threshold (used to establish, given its probability, if a pixel is a road) to compute different PR values. By doing so, we obtain the graph in Figure **??**.

We can observe three important things in this plot. Since the yellow curve is below every others, we can say that Unet is perfroming better than the traditional approach. The second observation is that the green curve is above the red one. Considering this we can conclude that adding more highway images and getting rid of the seaside images was a good thing. The third observation is about the blue curve. We can see that in some cases the blue curve is above the green one. In fact when we are not rotating the image the network learns better how to detect vertical and horizontal lines but the results are worse when predicting the others. That explains why the blue curve goes above the yellow one between 0.6 and 0.8. We concluded that the yellow curve corresponds to the best model and therefore the one we used for the Kaggle competition.

VI. Conclusion

Both models proved to be versatile and effective in achieving good predictions. However, our traditional CNN model failed in recognizing narrow roads, especially when they are covered by shadows, those types of roads are indeed almost inexistent in the training set while present in many images of the test set. Moreover, the train and test datasets were mainly restricted to cities. To obtain a model that would efficiently work with a wider range of input images such as images with snow, country images or on different meteorological conditions, we would require a bigger and diverse training set. In addition, without any preprocessing other than data augmentation, both models outperformed the baselines and confirmed their superiority in solving this kind of problem.