Mini-project I

Lucía Montero Sanchis, Milica Novaković, Ada Pozo Pérez,
Deep Learning 2018, EPFL Lausanne, Switzerland

***Abstract*—In this report we train a model based on LSTM and CNN for predicting the laterality of finger movements based on Electroencephalography (EEG) recordings. We evaluate the effect of preprocessing or not the data and compare the results obtained using a baseline model based on SVM, with accuracies up to 74%, to the ones obtained using LSTM, CNN and our final model that combines LSTM and CNN architectures. These last two models are capable of outperforming the baseline, reaching 80% accuracy, whereas the LSTM is not capable of modelling the data and obtains worse results. It is found that despite the noisy data found in EEG recordings, preprocessing is in general detrimental for models' performance and using raw input data should be preferred.**

I. Introduction

In this project we train a predictor of finger movements from Electroencephalography (EEG) recordings using the PyTorch framework. The objective is to implement a neural network that predicts the laterality of the finger movement, i.e. if it corresponds to a right or left movement, from the EEG recording.

The difficulty to solve this kind of problems based on neuroimaging comes in general from the lack of large scale datasets [1]. This limited number of samples difficults training deep models with millions of parameters.

We attempt to tackle this problem using different models. First, we use Support Vector Machines (SVM) [1] as a baseline, which helps us assess the accuracy of models based on Long Short-Term Memory Unit (LSTM) [1], [2] and Convolutional Neural Network (CNN) [1], [3]. Several experiments are conducted with these models, evaluating the results obtained under different conditions such as using the raw data or preprocessing it.

The rest of this paper is organized as follows: Section II describes the database and the preprocessing steps considered. The methods and architectures used are introduced in Section III, while the results are presented and discussed in Section IV. Lastly, conclusions are drawn in Section V.

II. Database

The database used is the dataset IV of the "BCI Competition II" [4]. The training set is composed from 316 samples, whereas the test set has 100 test samples. The signals were recorded at 1 kHz during 0.5 seconds with a band-pass filter between 0.05 and 200 Hz. Each of them is composed of 28 EEG channels.

In addition to the signals captured with 1 kHz sampling rate, this work also uses a version downsampled to 100 Hz.

A. Preprocessing

In general, EEG signals typically have low signal-to-noise ratio and suffer other interferences such as power line noise and artifacts from other muscles and eye movements [3], [2], [1]. To overcome these problems we have investigated both filtering the signal and reducing the number of electrodes before using it as input to the network.

1) Filtering and normalization:

- Notch filter: A notch filter was applied to the frequency range of 50 Hz with a quality factor of $Q = 30$ [2]. The motivation was to reduce the power line noise [2].
- High pass filter: The objective of this filter is to reduce artifacts that may have been produced by movements of the electrodes against the skin [2]. This is implemented using a IIR Butterworth filter of order five with cutoff frequency of 0.5 Hz [2].
- Standardization: After filtering, the EEG signals are normalized using standardization, that is, removing the mean and dividing by the standard deviation of each channel.

Since in general the tendency with deep learning is to implement as little preprocessing as possible [3], the models will be evaluated with and without this filtering and the resulting performances will be compared.

2) Reducing the number of electrodes: Having in mind that few brain areas are in charge of finger movements [5], we study whether the information from a couple of electrodes is enough to decode the finger movements. The motivation for this is that using only the electrodes that contain the most discriminant information about the movements could help reduce the complexity of the models.

We use Fisher distance to determine which electrodes are the most discriminant, keeping only the ones with values higher than a certain threshold. However, this simplification seems to be detrimental and reduces the performance in simple experiments. Hence, we consider all electrodes in all the experiments explained in Section IV.

III. Methods

We attempt to solve the task presented using Support Vector Machines (SVMs) and deep neural networks with models based on Long Short-Term Memory Units (LSTMs) and Convolutional Neural Networks (CNNs).

A. Support Vector Machines (SVM)

SVMs try to find the hyperplane that best separates two classes by leaving the maximum margin from both of them. The margin is defined as the distance from the nearest point of each class to the hyperplane. In other words, the margin

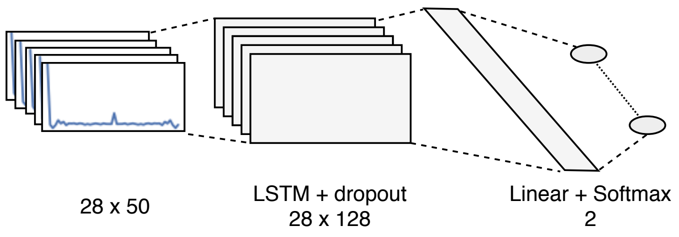


Fig. 1: Our architecture for the LSTM model. It has first a LSTM layer with 128 hidden units, followed by a dropout layer with 0.5 drop probability. The final layer has two output units and uses Softmax.

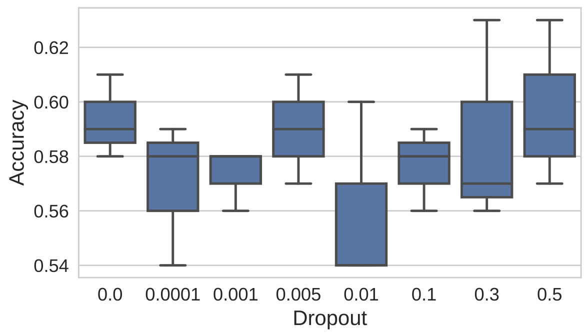


Fig. 2: Validation accuracy with different dropout values using 3-fold cross validation for the LSTM model.

is the minimum distance between the hyperplane and each of the classes.

In this work we will use a linear kernel and a non-linear Radial Basis Function (RBF). The first one only has a regularization parameter C, which tends to reduce overfitting the smaller it is, whereas RBF also has the width of the Gaussian kernel γ as parameter. A smaller γ implies that the kernel varies more abruptly the kernel and thus the SVM will be more prone to overfit.

B. Long Short-Term Memory Units (LSTM)

Recurrent Neural Networks (RNN) are neural networks with feedback loops within layers that are able to model and predict long-term patterns in an input signal [6]. However, they suffer from the vanishing gradient problem, which limits the information learn in each timestep from its context. As a consequence, LSTM networks were introduced with gated units that control how information from the previous timesteps and how much information from the new sample should be combined [7]. The use of a RNN in this work, and concretely a LSTM, is motivated by the fact that EEG signals are highly-dynamic signals that change over time [2]. Thus, it is expected that a LSTM can capture and model such variations over time.

The architecture used for this model is based on the one presented in [2], evaluating the performance of models with up to three hidden layers and with 32 to 256 hidden units using 3-fold cross validation. This cross-validation was performed on the raw input data downsampled and without preprocessing. The best-performing architecture was found to be using only 1 layer with 128 hidden units. A higher number of layers or units tended to result in overfitting, while a smaller number underfitted. This is consistent to what could be expected due to the small number of samples. The prediction of the LSTM is produced after processing the whole signal. In addition, to reduce overfitting this layer is followed by a dropout layer. Finally, the output layer uses a softmax activation function with two output units. The final architecture can be found in Figure 1.

The value for the dropout probability between the LSTM and the fully connected is tuned using 3-fold cross-validation on the downsampled input data without any preprocessing. Figure 2 shows the results obtained. It can be observed that there is a great variability within the same dropout value. The dropout value is fixed to 0.5, that is one out of two hidden units is dropped.

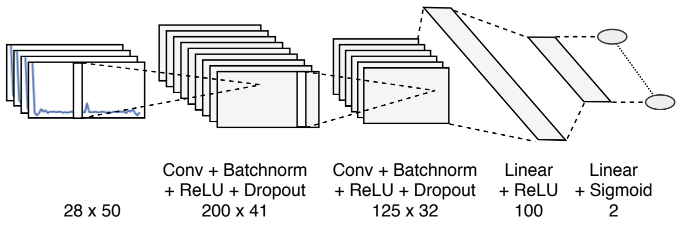


Fig. 3: Our architecture for the CNN model. The kernel size for both convolution layers is 10. The final layer has two output units and uses Sigmoid.

C. Convolutional Neural Networks (CNN)

The main component of CNNs is the convolutional layer, where the learning of features is attained by convolving a set of filters over the input. Architectures are typically composed of convolutional layers, ReLU or ELU activations, batch normalization, pooling, dropout and fully-connected layers [2].

CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing [8]. However, in this work we compare the performance achieved with and without filtering. The architecture used is based on the deep CNN presented in [3], reducing the number of hidden layers to prevent overfitting and using ReLU activation layers instead of ELU as shown in Figure 3. After evaluating the performance using a 1D and a 2D convolutional layer for the first layer – in the first case convoluting only over time, in the second case over time and channels – we decided to use only 1D convolution over time [1]. The number of channels produced by the convolution was adjusted using random search and cross-validation.

After choosing the number of channels we reduced overfitting by adding dropout layers and L2 penalty in the Adam optimizer. The values for both parameters were adjusted as shown in Figure 4, which presents the results obtained after running 3-fold cross-validation for experiments each with a different value of regularization and dropout probability. The

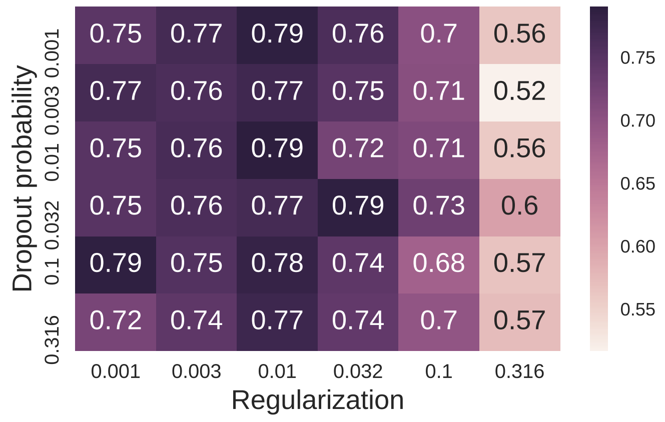


Fig. 4: Validation accuracy for CNN with different dropout probabilities and L2 regularization, considering in each experiment the epoch where the highest average accuracy is obtained.

numerical values in the figure have been computed by averaging the learning curves over the three folds, and choosing the epoch with the highest mean in each experiment. From these results we decided to use 0.0316 for both regularization and dropout probability, since it is the set of parameters for which we have a lowest train accuracy among the ones with highest validation accuracy – and is therefore less likely to overfit.

IV. Results and discussion

In this Section we present the results obtained for each of the chosen architectures. All neural network models have been trained using Adam with Cross-Entropy as loss function. The minibatch size used is 25 samples. The trained models are:

1) Support Vector Machine (SVM).
2) Long Short-Term Memory Units (LSTM).
3) Convolutional Neural Network (CNN).
4) LSTM and CNN combined model.

A. Baseline - Support Vector Machines (SVM)

We first obtain a baseline for our problem using Support Vector Machines. We evaluate the performance of a system using both a linear and a RBF kernel. As explained in Section III-A, both of them have a regularization parameter C, while the RBF has in addition γ. We tune both parameters using 3-fold cross validation on the downsampled training set, both for the linear and the RBF kernel with and without the preprocessing described in Section II-A.

The results from this classifier, as well as the values for γ and C obtained can be found in Table I. It can be observed that the best result is obtained with a linear kernel with 73% accuracy. The performance degrades if the input is preprocessed for both types of kernel. In addition, we show the results using the input samples without downsampling. The accuracy doesn't improve significantly and moreover, it decays in most cases, specially with the RBF kernel. In addition, it should be mentioned that the best results are obtained with the downsampled version of the dataset.

TABLE I: Parameters and accuracy using SVM. 100 Hz refers to the downsampled version of the dataset and 1 kHz to version without downsampling.

		Parameters	**Accuracy (%)**	
			100 Hz	**1 kHz**
Linear	**Without preprocessing**	$C = 0.0001$	73	74
	With preprocessing	$C = 1$	66	60
RBF	**Without preprocessing**	$C = 10000$ $\gamma = 1e-05$	63	49
	With preprocessing	$C = 10000$ $\gamma = 1e-04$	65	57

B. Long Short-Term Memory Units (LSTM)

The LSTM is trained with Adam as optimizer. The learning rate is set to 0.001 and the decay rates of the first and second moments are 0.9 and 0.999 respectively, as recommended in [9]. Different values for these parameters as well as for the L2 penalty were evaluated with no improvement in the performance.

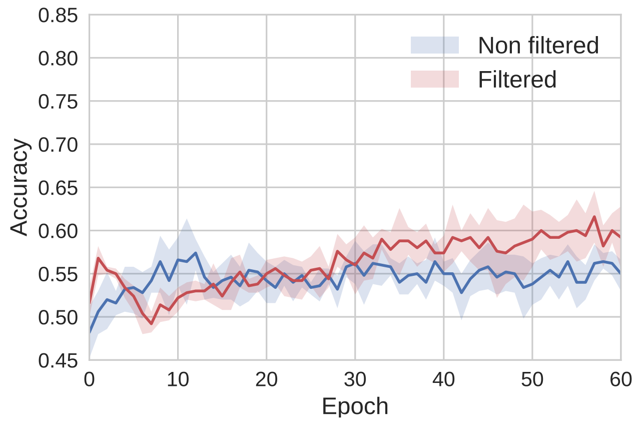
We study the performance of this model with the architecture presented in Section III-B using both the data downsampled to 100 Hz and with 1 kHz sampling rate and with and without filtering, so we can compare the effect these parameters have. Figure 5a shows the results with 100 Hz while Figure 5b shows the results with 1 kHz after training for 60 epochs. In both cases, the models are trained five times and the mean result per epoch and 95% confidence interval presented. With 100 Hz the best accuracy, around 60% accuracy, is obtained with the filtered data. On the other hand, the performance decreases with 1 kHz as sampling rate, reaching only 55% accuracy with both filtered and non-filtered data. This is consistent with what one could expect from LSTMs, since they usually benefit from shorter time sequences [2]. Hence, with the higher sampling rate the model tends to learn less, especially with filtering, where the model performance decreases as the number of epochs grow. A possible cause is that filtering removes information from the data, making it easier to overfit.

In comparison to our baseline, this model achieves worse accuracies and it's clearly outperformed.

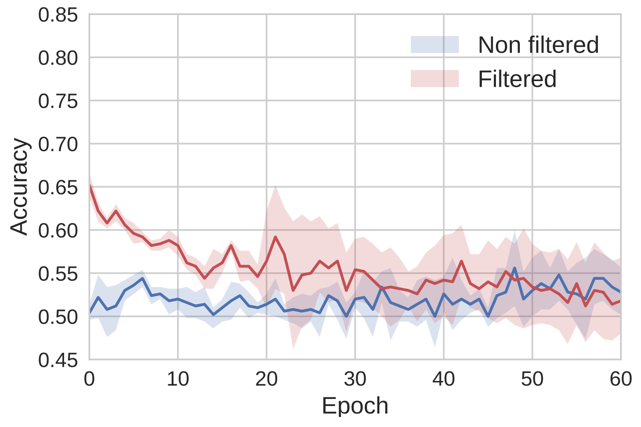
C. Convolutional Neural Network (CNN)

We study the performance of the CNN model with the architecture presented in Section III-C. Similarly to the analysis carried out in Section IV-B, we train with Adam using the same parameters as before for the data with 100 Hz and 1 kHz sampling rates and with and without filtering. Figure 6a shows the results with 100 Hz while Figure 6b shows the results with 1 kHz after training for 200 epochs.

In all cases the accuracy is between 70% and 80% for the non-filtered data, and around 60% for the filtered one. The reason for this is that the CNN is able to learn the features from the training data better than the LSTM. By filtering the input signals we are removing information, resulting in a worse

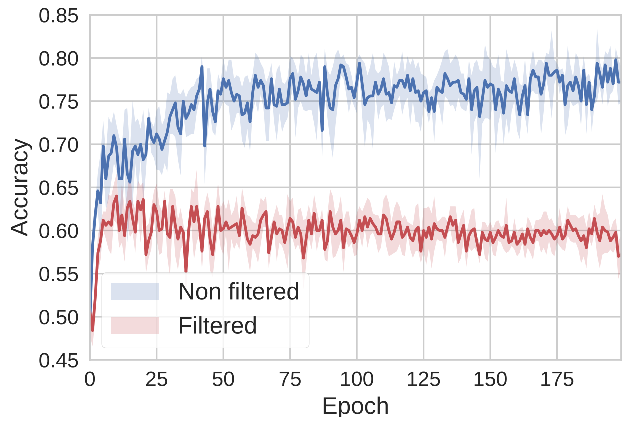


(a) Accuracy with 100 Hz data.

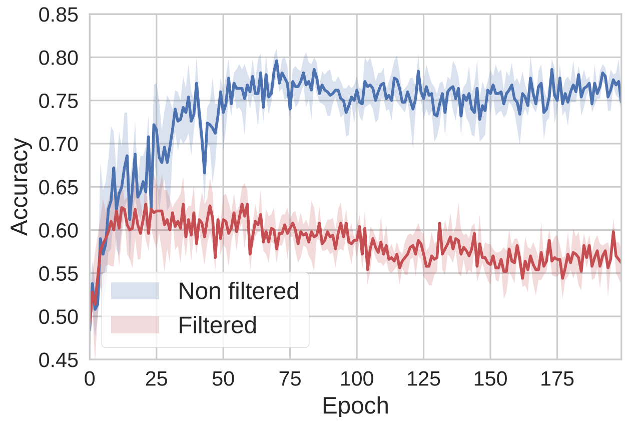


(b) Accuracy with 1 kHz data.

Fig. 5: Accuracy of the LSTM model for the downsampled and full versions of the dataset training the model five times. In deeper blue and red are the mean values, while in lighter colors are the 95% confidence intervals.



(a) Accuracy with 100 Hz data.



(b) Accuracy with 1 kHz data.

Fig. 6: Accuracy of the CNN model for the downsampled and full versions of the dataset training the model five times. In deeper blue and red are the mean values, while in lighter colors are the 95% confidence intervals.

performance of the CNN. We can also note that for 1 kHz data, filtering causes the model to overfit – as seen from the accuracy decrease around epoch 100 for the filtered curve in Figure 6b.

When comparing the accuracies obtained for the non filtered, downsampled and non-downsampled data, we can observe that it is considerably similar for both. It is worth noting that the parameters chosen for the model had been optimized for the downsampled training dataset, so the fact that the accuracy is similar shows the robustness of the CNN model.

This model achieves better accuracies than both the LSTM model and the baseline.

D. LSTM and CNN combined model

In addition to the previous experiments, we combine the CNN and LSTM models. Both models are trained on parallel with the architecture described on Sections III-B and III-C, removing the final fully connected of each model. Instead, the results from both models are concatenated and a final fully connected layer with Sigmoid as activation function is added to obtain the output.

The resulting model is trained with 0.032 L2-regularization using non-filtered data with 100 Hz and 1 Khz sampling rate. Since it has been found that the filtering is, in general, detrimental to the performance, we do not report the accuracies with this preprocessing.

The accuracy obtained can be found in Figure 7. It can be observed that the result is almost the same with the downsampled and full versions of the dataset. Both obtain accuracies close to 80%, with the downsampled version performing slightly better. These values are similar to the ones reached using only CNN and no clear improvements can be observed.

For completeness, Figure 8 shows the training and test accuracy curves for the signal downsampled to 100 Hz. We can see that the model does not seem to learn much after 125

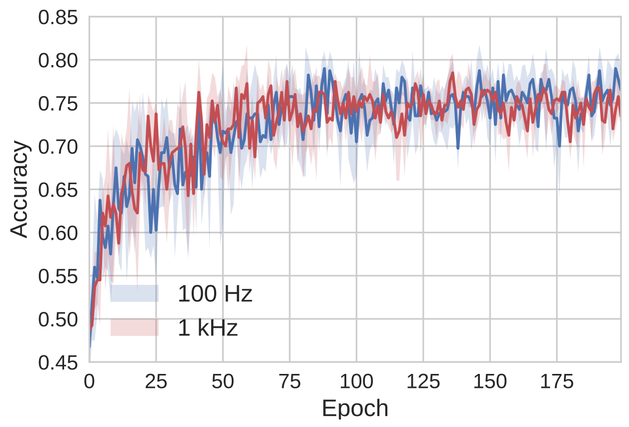


Fig. 7: Accuracy of the LSTM and CNN combined model for the downsampled and full versions of the dataset training the model five times. In deeper blue and red are the mean values, while in lighter colors are the 95% confidence intervals.

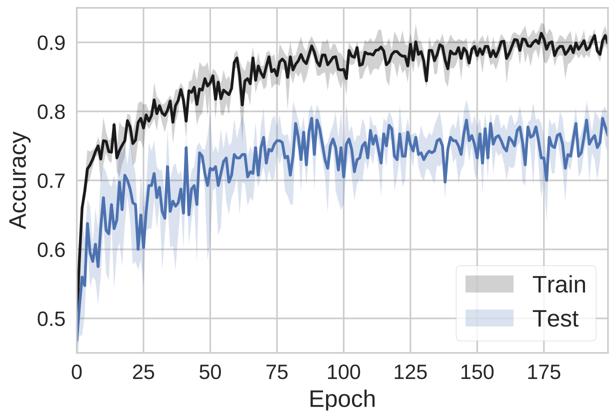


Fig. 8: Train and test accuracy of the LSTM and CNN combined model for the downsampled input signals after training the model five times. In deeper black and blue are the mean values for train and test respectively, while in lighter colors are the 95% confidence intervals.

epochs, which prevents it from overfitting but may indicate that the learning capacity of the model could be improved.

V. Conclusions

In this project we proposed different solutions for predicting the laterality of finger movements. Our final model was a combination of CNN and LSTM predictors, which results in a model that achieves between 70% and 80% accuracy. These values are similar to what we achieved using the model based on CNN, with the advantage that the combined model was more robust and performed similarly for signals both with 100 Hz and 1 kHz sampling frequencies. We could also verify that there was not a clear advantage of using a higher sampling frequency, which increments the complexity. Nevertheless, the proposed models generally perform better when using input signals downsampled to 100 Hz.

Regarding preprocessing, we initially considered filtering and normalizing the input signals as well as reducing the initial number of electrodes. Neither of these techniques resulted in better results, since reducing the amount of information in the signals reduced the accuracy of the models.

Future work could include using the CNN to obtain a series of features that could be used as inputs to a LSTM. In addition, the parameters of our models should be tuned trying more combinations, which would probably increase the accuracy – particularly for the combined model.

References

[1] P. Bashivan, I. Rish, M. Yeasin, and N. Codella, “Learning representations from EEG with deep recurrent-convolutional neural networks,” *CoRR*, vol. abs/1511.06448, 2015.

[2] J. Fedjaev, “Decoding eeg brain signals using recurrent neural networks,” Master’s thesis, Technische Universitat Munchen, 2017.

[3] 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,” *CoRR*, vol. abs/1703.05051, 2017.

[4] “BCI Competition II,” http://www.bbci.de/competition/ii/, accessed: 2018-05-14.

[5] J.-A. Rathelot, R. Dum, and P. Strick, “Posterior parietal cortex contains a command apparatus for hand movements,” *Proceedings of the National Academy of Sciences of the United States of America*, 2017.

[6] J. L. Elman, “Finding structure in time,” *COGNITIVE SCIENCE*, vol. 14, no. 2, pp. 179–211, 1990.

[7] R. Jozefowicz, W. Zaremba, and I. Sutskever, “An empirical exploration of recurrent network architectures,” in *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML’15, 2015, pp. 2342–2350.

[8] “LeNet-5 convolutional neural networks, Lecun, Yann,” http://yann.lecun.com/exdb/lenet/, accessed: 2018-05-14.

[9] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2014.