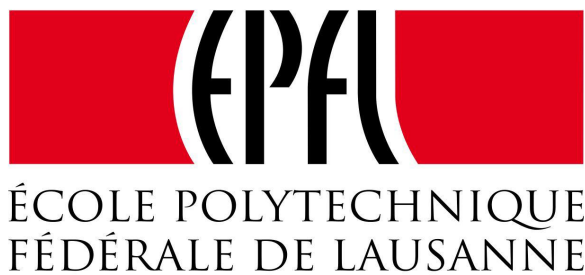


Deep Learning Project 1

Group 58: Xingce Bao, Xi Fan, Danjiao Ma

Date: May 17, 2018

Course: Deep Learning



Contents

List of Figures	1
List of Tables	1
1 Project Description	2
2 Attempts of Different Structures	2
2.1 Linear Predictor	2
2.2 Summary of Attempts of Different Structures	3
3 Final Structure	4
3.1 Pre-processing	4
3.1.1 Channel Selection	4
3.1.2 Time Window	5
3.1.3 Data Normalization	5
3.1.4 Other Failed Approaches	5
3.2 Neural Network Based on Convolution Layers	5
3.3 Other Parameters	7
3.3.1 Optimizer	7
3.3.2 Loss Function	7
3.3.3 Other Parameters	7
3.4 Results and Discussion	7
Bibliography	9

List of Figures

1	Loss and error rate of linear predictor	3
2	Scheme of electrode placement	4
3	Loss and error rate of final structure	8

List of Tables

1	Linear predictor layers structure	2
2	Comparison of different structures	4
3	Convolution layers structure	6

1 Project Description

This project is aiming to predict the left/right finger movement according to the EEG signals. The input data set of this project is a multichannel EEG sampled at 1 kHz for 0.5s, which shows the signal intensity of 28 different channels along the time domain. This project takes the $316 \times 28 \times 500$ data set as the training set and the $100 \times 28 \times 500$ as the testing set. The output of this project is an indicator which tells whether the corresponding movement is left or right, where 0 refers to left and 1 refers to right.

During designing and implementation, we have noticed that this data set has relatively inadequate number of samples to prevent over-fitting, which is the biggest problem to overcome in this project.

The final structure of contains the pre-processing of the EEG signal and the neural network. Pre-processing of this EEG time domain signal is conducted to improve the performance of the neural network, especially its stability and accuracy. The neural network, mostly based on convolution layers, is established and trained. The testing of this structure shows an average accuracy of 81%, with a standard deviation of around 0.044.

2 Attempts of Different Structures

2.1 Linear Predictor

First, a linear predictor is established and tested to observe the data set quality and character without any pre- or post-processing. The network structure is shown as below with an Adam optimizer.

Layer	Parameter
Linear	13*300, 13*40
ReLU	null
Linear	13*40, 80
ReLU	null
Linear	80, 2
Softmax	null

Table 1: Linear predictor layers structure

With tuning parameters we got the result of about 98% accuracy for the training set yet about 67% accuracy for the testing set.

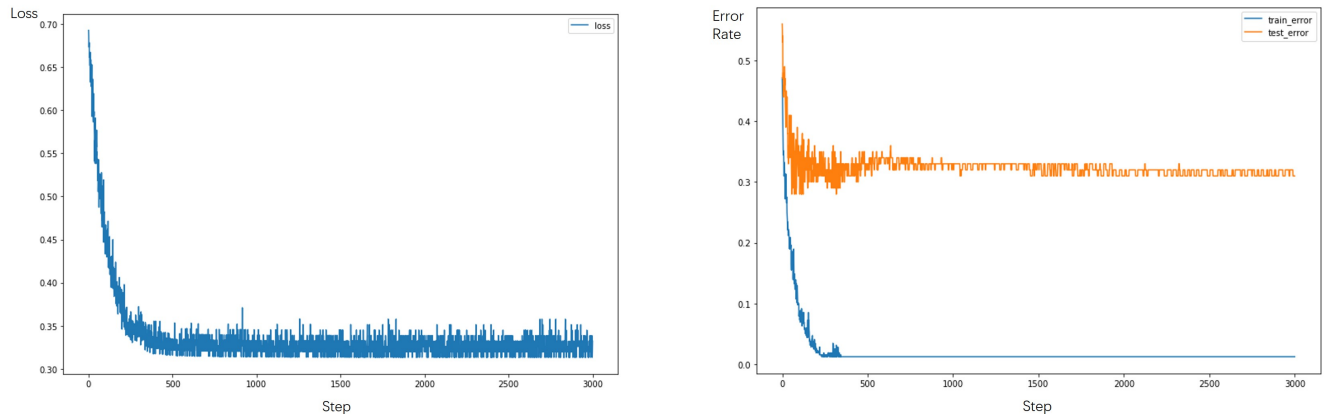


Figure 1: Loss and error rate of linear predictor

The main reason of this huge difference between the error rates of train and test is over-fitting. Since the data set does not have many samples, the capacity of the network can either be too small to reach low enough loss or too big that the additive complexity makes the network not focusing on the main feature of the dataset, which means the network is over-fitting.

To mitigate this problem, more advanced network structure will be adopted as well as the pre-processing of the input.

2.2 Summary of Attempts of Different Structures

A summary of several structures that we have tried including the final structure are shown in the table below to compare their performances.

The 'Linear' structure refers to the linear predictor that we described in the beginning of Chapter 2, which only includes fully connected layers and activation layers. The best performance of this structure after parameter tuning is shown in the table 2.

To mitigate the problem of over-fitting of the 'Linear' structure, a structure that adds batch normalization layers and dropout layers to the previous 'Linear' structure is trained and tested (i.e. the 'Linear + BatchNorm & Dropout'). From the obtained error rate, it is obvious that the batch normalization and dropout helps the mitigation of over-fitting.

However, linear predictor cannot reach low enough error rate even with additional layers or optimized parameters, which leads to the adoption of convolution layers and pre-processing of the input dataset. The 'Convolution (no Batch Norm & Dropout)' is a structure that contains convolution layers and max-pooling layers to focus on the extraction of signal features. It performs as well as the 'Linear + BatchNorm & Dropout' however with relatively high instability of prediction performance. To improve the accuracy and stability of the structure, we will bring batch normalization and dropout in this structure to be our final structure, which will be introduced in Chapter 3.

Another structure that has one less convolution layer and one less max-pooling layer than the final structure, which refers to the 'Convolution (less layers)' shows a worse performance than the

final structure, which indicates that the depth of layers in the final structure that we optimized is a more reasonable option.

The final structure is described in Chapter 3.

Structure	Average Test Error Rate	Test Error Rate Std Dev
Linear	0.33	0.015
Linear + BatchNorm & Dropout	0.28	0.013
Convolution (no BatchNorm & Dropout)	0.29	0.111
Convolution (less layers)	0.246	0.106
Final Structure	0.19	0.044

Table 2: Comparison of different structures

3 Final Structure

3.1 Pre-processing

3.1.1 Channel Selection

From the description of the data set we can see that the 28 channels of signal correspond to 28 points on the scalp where EEG electrodes are placed. The scheme that is provided by the source of the data set is shown below[1-3], where we have the channels in the order of F3, F1, Fz, F2, F4, FC5, FC3, FC1, FCz, FC2, FC4, FC6, C5, C3, C1, Cz, C2, C4, C6, CP5, CP3, CP1, CPz, CP2, CP4, CP6, O1, O2.

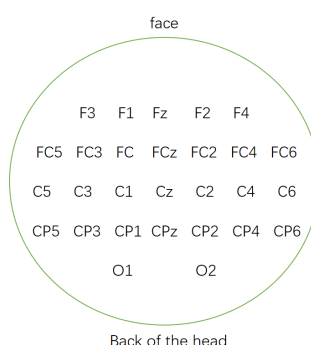


Figure 2: Scheme of electrode placement

According to EEG studies, finger movements have specific correspondence with only 8-10 of the 28 channels[4-5], which means that the 28 channels are excessive and bring more noise in the structure. Our approach is to reduce the number of channels as input by selecting some more relevant channels and abandoning the others. 13 channels are selected from the total 28 channels.

The number of 13 is chosen such that it won't take too much time to find a group of channels that contains most of the relevant channels and it won't bring too many irrelevant, i.e. noisy channels in the selected group.

The method is: Separate the 316 multichannel trials (i.e. the training set) into 216 sub-training trials and 100 validation trials. For each attempt, by using the linear predictor, we randomly select 13 channels out of 28 as the sub-training set and use the validation set to observe the accuracy of implementations of different selections. After several attempts, a collection of channels with index 16,20,24,17,9,18,23,10,4,5,25,7 and 3 is obtained since this selection outperformed all other attempts.

3.1.2 Time Window

According to the description, this dataset was recorded from computer keyboard that was pressed by the participants at either left part or right part. The 500 ms length of sampled signal corresponds to the brain activity ending 130 ms before a key-press[4,6,7].

Based on the above, we found that the later part of the 500ms has more relativity with the key-press action, which means that the time window that is set from the 200th to 500th point of the sample signal can improve the coherence between the input and the target.

3.1.3 Data Normalization

As different EEG channels have different ranges of data values, data normalization is conducted to have the same range of values for each of the inputs to the ANN model to prevent the network to be saturated therefore make sure we have stable convergence of weights and biases.

3.1.4 Other Failed Approaches

In order to improve the performance of the ANN, we have also tried several other methods of pre-processing including Fourier transform (as shown below), filtering (since the EEG signal seems to be more dominant at the 0-10Hz and 20-30Hz frequency range[6,7]), wavelet and so on. However these methods prove to be more or less ineffective to help with the improvement and therefore not adopted at last.

3.2 Neural Network Based on Convolution Layers

This project is about extract specific features from the complicated data set which makes convolution layers very suitable since a convolution layer is good at reduce the complexity of the input, extracting features from noisy data set and mitigate the problem of over-fitting.

The structure of the ANN based on convolution layers is shown below.

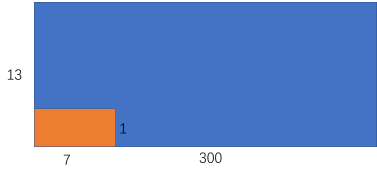
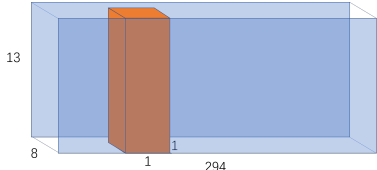
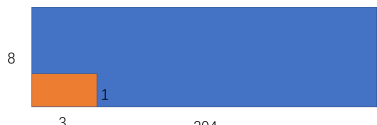
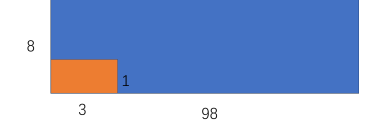
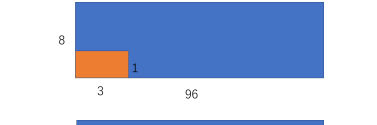
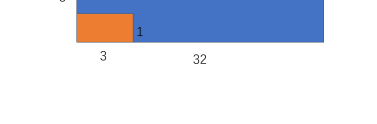
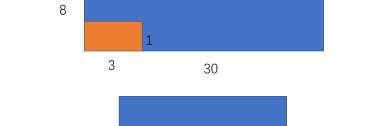
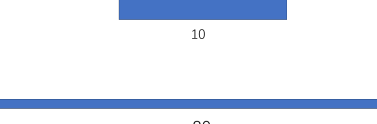


Layer	
Permute((0,3,1,2))	
Conv2d(1,8,(1,7))	
BatchNorm2d(8), Permute((0,2,1,3))	
Conv2d(C,1,(1,1))	
BatchNorm2d(1), Dropout(0.3), ELU()	
MaxPool2d((1,3))	
Conv2d(1,1,(1,3))	
BatchNorm2d(1), Dropout(0.4), ELU	
MaxPool2d((1,3))	
Conv2d(1,1,(1,3))	
BatchNorm2d(1), Dropout(0.5), ELU	
MaxPool2d((1,3))	
Flatten	
BatchNorm1d(80)	
Linear(80, 2)	
Softmax	2

Table 3: Convolution layers structure

The adoption of convolution layers is based on the fact that the data set is composed of time

domain signal that is suitable for convolution. The signal is very noisy therefore by using convolution layers we shall use relatively small amount of parameters in the layers to mitigate over-fitting.

Other necessary layers include batch-normalization (to be able to have higher learning rates and to reduce over-fitting with its regularization effects without losing information), dropout (to reduce over-fitting since the NN is complicated for the small number of training samples), max-pooling (to extract most dominant features), ELU (exponential linear unit that we used as an activation function to improve the robustness against the noise of the input), linear layer, and softmax.

3.3 Other Parameters

3.3.1 Optimizer

This project chose Adam optimizer. Adam Optimizer is an advanced method to reach minimum through gradient descent. It is based on the combination of Adaptive Gradient Algorithm and Root Mean Square Propagation, which makes the Adam very effective computing and suitable for a data set that is very noisy like the data set of this project.

Other optimizers that we have tried include SGD and RMS propagation, which didn't perform as well as Adam since they tend to get trapped in the local minimum.

3.3.2 Loss Function

The loss function of this project is set to be the Cross Entropy loss, which is suitable for classification especially for a network with targets as $[0,1]$ and $[1,0]$.

3.3.3 Other Parameters

The tuning of the parameters for the whole structure is a major time-intensive work. The tuned parameters include the learning rate (0.005), the batch size (158), the number of epoch (1500), as well as the parameters for the layers in the network.

One example of this kind of tuning is the parameter of the layer 'BatchNorm2d(8)' where we have 8 as the parameter which indicates the C from an expected input of size (N,C,H,W). This parameter is tuned to be 8 since larger value will result in over-fitting and smaller value will result in bad performance of training in terms of reducing the loss.

3.4 Results and Discussion

The final results of the implementation of this final structure of the project is shown below. Generally we achieved an average accuracy of 81% with a best performance of 86% accuracy. The

structure performance is stable with a standard deviation of 0.044 indicating the high reliability of this structure.

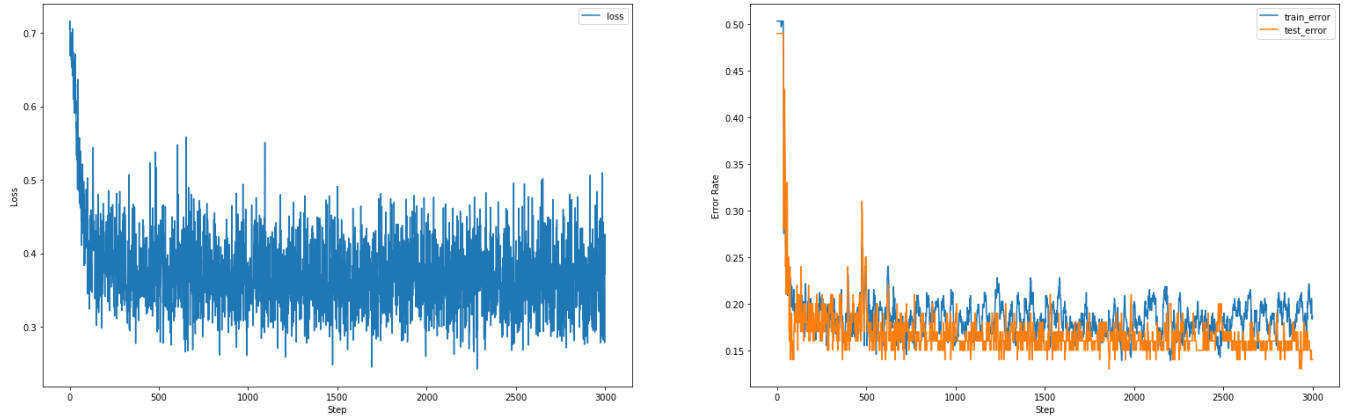


Figure 3: Loss and error rate of final structure

The results shows that the combination of pre-processing and convolution layers has significant effect on mitigating over-fitting in this case, since they help reducing the complexity of the input and finding the main features of the samples within small amount of input trials.

Bibliography

- [1] X. An, D. Kuang, X. Guo, Y. Zhao, and L. He, “A Deep Learning Method for Classification of EEG Data Based on Motor Imagery,” in *Intelligent Computing in Bioinformatics*, vol. 8590, D.-S. Huang, K. Han, and M. Gromiha, Eds. Cham: Springer International Publishing, 2014, pp. 203–210.
- [2] Y. Wang, Z. Zhang, Y. Li, X. Gao, S. Gao, and F. Yang, “BCI competition 2003-data set IV: An algorithm based on CSSD and FDA for classifying single-trial EEG,” *IEEE Trans. Biomed. Eng.*, vol. 51, no. 6, pp. 1081–1086, Jun. 2004.
- [3] “BCI Competition II.” [Online]. Available: <http://www.bbc.de/competition/ii/>. [Accessed: 25-Apr-2018].
- [4] “berlin data set for bci competition 2003.” [Online]. Available: http://www.bbc.de/competition/ii/berlin_desc.html. [Accessed: 20-Apr-2018].
- [5] E. Parvinnia, M. Sabeti, M. Zolghadri Jahromi, and R. Boostani, “Classification of EEG Signals using adaptive weighted distance nearest neighbor algorithm,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 26, no. 1, pp. 1–6, Jan. 2014.
- [6] B. Blankertz, G. Curio, and K.-R. Müller, “Classifying Single Trial EEG: Towards Brain Computer Interfacing,” in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. MIT Press, 2002, pp. 157–164.
- [7] R. T. Schirrmeister et al., “Deep learning with convolutional neural networks for EEG decoding and visualization,” *Hum. Brain Mapp.*, vol. 38, no. 11, pp. 5391–5420, Nov. 2017.