EEG Signals Classification Using Neural Networks

Xinyi Gu, Yutan Gu, Di Jin

Abstract

This project investigates different deep learning models such as Fully Connected Neural Network, Convolutional Neural Network and Recurrent Neural Network performs on the classification of the motor imaginary signals from EEG data. From the project, CNN outperforms among all the the data, and achieves the accuracy as high as 60%.

1. Introduction

The main goal of this project is to classify the EEG data, with four outcome classes. Two main potential issues we may face of classifying EEG data are inadequate training data and its low signal-to-noise ratio. Considering these challenges, we first do some preprocessing for the raw EEG data, then design several neural network models to see how well they can perform in this task.

1.1. Preprocessing

We first exclude the three electrodes in the front face, which are the EOG data, since they are not closely related to the classification task. Thus, we have a total of 22 electrodes channels.

Notice that each trial of each subject contains 1000-time bins while the signals were sampled with 250Hz. Thus, each trial contains a signal within 4 seconds. Here we decide to use a window to extract some random subsample from one trial. For example, if we choose window_length=300 and window_number=30 as default, we will randomly choose 30 consecutive 300-time-bins windows from the raw trial data, augmenting the data by 30 times. The label of each subsample remains the same.

Moreover, considering the low SNR of the EEG data, we try to apply PCA to every trial to reduce the error caused by redundant information. This also helps to reduce the computation time since we reduce the dimension from 22 to 10. And considering that there are some NaN data points will cause the error when doing PCA, we also delete the trial which includes the NaN data points using try and expect expression.

In RNN and LSTM we also used wavelet analysis to preprocess our data and this new method has a better performance. We know that in motion analysis, the signal will be filtered by 8 - 30 Hz. In addition to reduce the noise in the signal, it is mainly because that this band includes the alpha and beta frequency band which have been shown to be most important in motion analysis. Here we analysis the signal by using wavelet analysis and filter the signal into 0-32 Hz which includes the alpha and beta bands we need. Hence the input is N(samples) * 254 * 10.

1.2. Fully Connected Neural Network

Fully Connected Neural Network is probably the simplest neural network we get. Since the dataset is not too large, we use a 3-layers FCNet, with batchnorm and dropout following the hidden layers. The 1st layer has 800 hidden units, and the second layer has 100 hidden units. The dropout ratio we use is 0.5 to reduce the overfit.

1.3. Convolutional Neural Network

Convolutional Neural Networks(CNN) have been proved to be a great architecture when dealing with different kinds of tasks, such as image classification. We now try to build a CNN model to classify the EEG data. For a EEG classification problem, in each trial we have a electrode*time_bins data. Thus, we can see the EEG data in each trial as a channel=1 image data.

We choose to use 2 Conv layers in our model to reduce the complexity and save running time. But we also add 2 FC layers to increase the performance of the model. Each Conv layer is followed by a batchnorm layer, a Relu activation function, a dropout layer, and a max-pool layer with filter 2*1. The first Conv layer has 24 filters with size 12*5, and the 2nd Conv layer has 48 filters with size 10*6. We choose not to use square filters because our data is rectangle. We also use 0.5 dropout here because our small dataset tends to have serious overfit problem. Moreover, we utilize xavier initialization for each Conv layer.

1.4. Recurrent Neural Network

RNN is designed to utilize the sequential information with a cyclic structure. However, traditional RNN will always faces the problems of exploding and vanishing gradient. And LSTM (long short-term memory) are designed to solve this problem.

We use RNN and LSTM to train and test EEG data. The data is splitted in to train, validation and test data, preprocessed by the same wavelet and PCA decomposition operations. Notice that here we will not use the previous data augmentation preprocess. Thus, the input size will be N * 254 * 10. We use orthogonal initialization and choose 1-layer network because it works the best. In order to avoid the model overfitting, we stop learning when the validation loss stop decreasing for 5 epodes and use current model to test the test data.

2. Results

We first optimize the classification performance of each subject individually, and compare each model. Moreover, we take a deeper look into the hyperparameter 'window_size' specifically of how it will affect the classification performance.

In training we use Adam as optimizer and cross-entropy loss as cost function. The train, validation and test split in each experiment is stated in the appendix. We use Pytorch as the deep learning frame to implement our models.

2.1. Optimization using Different Models

2.1.1 Fully Connected Neural Network

For FCNet, we train the model for 30 epochs, and the validation accuracy and test accuracy is shown below:

file	1	2	3	4	5	6	7	8	9
val (%)	32	30	44	32	28	34	34	34	42
Test (%)	30	18	38	32	28	44	34	42	28

The result of FCNet is not very stable. Validation accuracy for different files can varies largely from 28% to 44%. For the test accuracy, the best test accuracy is A06T, which is 44%, and the worst test accuracy is A02T, which is only 18%.

2.1.2 Convolutional Neural Network

For CNN, we train the model for 50 epochs, and the summary of validation and test accuracy is shown below:

file	1	2	3	4	5	6	7	8	9
val (%)	68	30	50	22	32	38	38	46	58
Test (%)	50	30	38	24	26	32	38	40	46

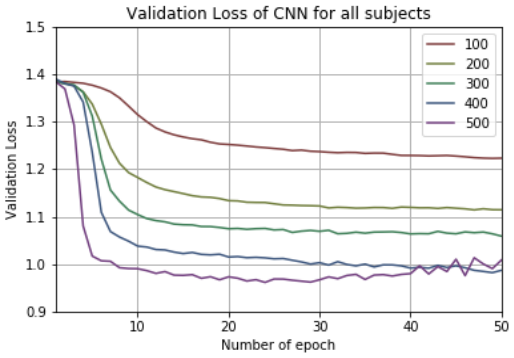
Similar to the result of FCNet, Validation accuracy and test accuracy for CNN model for different data files also varies largely. A01T fits the model best, and has the best validation accuracy 68%, and best test accuracy 50%. The model performance of A04T is the worst. The worst validation accuracy is only 22%, and test accuracy is only 24%.

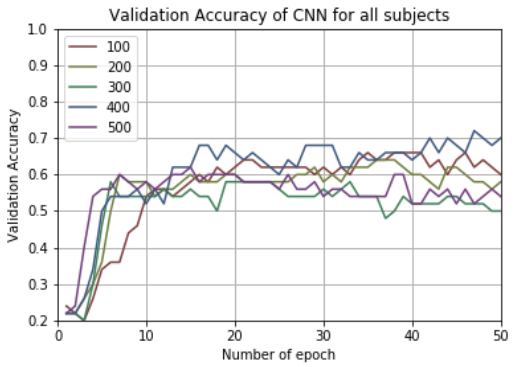
2.1.3 Recurrent Neural Network

file	RNN Val	RNN Test	LSTM Val	LSTM Test
A01T	40.0%	28.0%	20.0%	32.0%
A02T	31.6%	37.5%	31.6%	27.1%
A03T	20.0%	38.0%	20.0%	36.0%
A04T	40.0%	26.0%	25.0%	26.0%
A05T	20.0%	30.6%	20.0%	32.6%
A06T	50.0%	22.4%	20.0%	26.5%
A07T	50.0%	32.0%	30.0%	20.0%
A08T	26.3%	31.3%	31.4%	35.1%
A09T	44.4%	25.5%	27.8%	25.5%

2.2. Optimization on the window sizes

In the data preprocessing, we use random windows to augment the dataset. In this section, we tried different window sizes to analyze which window size yields the best performance. We choose to use CNN model and data file A01T since this combination gives the best performance in the previous section. We tried 5 window sizes from 100 to 500, and other hyperparameter remains unchanged. The validation loss and accuracy, and the test accuracy are shown below.





size	100	200	300	400	500
test acc	48%	50%	54%	50%	48%

From the plot, the change of window sizes has significant influence on the validation loss, but less influence on the validation accuracy as well as the test

accuracy. The validation accuracy of all five subjects are around 60%. The test accuracy is around 50%.

3. Discussion

3.1. Insights From Preprocessing

For the data augmentation, we tried 5 different window sizes from 100 to 500. We find the window size has relative large influence on the loss, but less influence on the accuracy. From the plots, validation loss for window 400 and 500 decrease rapidly in the first several epochs but have turbulence later. Validation loss for window 100 and 200 drops too slow. Validation loss for window 300 drops relatively quick and fluent. Moreover, the test accuracy for window 300 achieves 54%, which is the best. Therefore, we decide to choose 300 as our window size.

We have also introduced PCA operation to reduce the dimension to remove redundant information and reduce computation time. After trying several times, it seems that 10 is ideal as the performance using smaller components are extremely bad due to lack of too much information.

3.2. Insights From CNN

As for the architecture of the ConvNet, firstly use two Conv layers as we only have limited training data. Too many parameters will result in overfitting problem. We did try to use three Conv layers but found it hard to adjust the architecture or tune the hyperparameters to gain a good performance.

In each ConvNet, we first use 10 and 20 filters in each of the two layers respectively. However, we found it even difficult to improve the training accuracy, which might be a sign that the model is too simple and tends to underfit. We also tried 48 and 96 filters respectively but found the training process running slowly on our CPU-only laptops. We thus tried to use 24 and 48 filters, which gave a satisfied result. We also use dropout layer to solve the overfit problem and find a relatively good ratio 0.5, while we also tried 0.2 and 0.3 with worse results.

The introduction of batchnorm layer did improve the performance as it normalizes the inputs of each layer and solve the potential internal covariate shift problem. We also tried Xavier initialization for each Conv layer but got almost no improvement after doing so.

In the training step, we use Adam to update the weights. Compared to SGD, which is highly sensitive to a good initial learning rate, Adam converges sooner and can usually find a good local minimal as it automatically adjusts the learning rate of each weight.

3.3. Insights From RNN

Compared to CNN, the performance of RNN and LSTM seems bad. We first use the same random preprocess methods as we did in CNN for RNN and LSTM, but it doesn't work well. We thought it is probably because the EEG signal has a small SNR and contains too much noise. And the noise will become larger when the network continues to remember the signal from previous time and it results in the noise accumulating over time. As we know that 8-30 Hz is an important band in motion analysis which contains the alpha and beta bands. Here we use wavelet to do signal filter. We applied a 3-layer wavelet on the signal and get the 0-32 Hz bands from EEG signal. And this time, the RNN and LSTM work better than before. The result has been shown in Table.

From the results we can find that most of the files have a test accuracy around 30%, while the difference between validation is large. In our training process, we find that both RNN and LSTM experience an rapid overfitting. The train loss is keep decreasing during the learning while the validation loss will increase after some epoch of decreasing. The reason we think may result in the overfitting is the small size of the data. We have added dropout and regularization into networks but it still have a good performance as CNN do.

3.4. Model Comparison

FCNet model has the simplest architecture, thus it has the fastest speed when training. However, the performance of FCNet is around 30%, which is quite low. Therefore, FCNet may not be a good model to classify EEG signals.

CNN model has more complex architecture than FCNet. From tuning parameters, it yields the best performance among three models. However, due to the complexity of model, CNN needs much more time than FCNet, causing troubles for CPU-only laptops. But since it gives the best performance, it is still a good model to train EEG signals.

Compared to CNN, the validation and test performance of RNN and LSTM is not well. RNN and LSTM faces the problem of overfitting here which indicate that those two models may not suit for those EEG signals.

3.5. Insights From Training Across All Subjects

In previous sections, we train and test each individual subject. We also tried aggregating all subjects together to train and test. We used our best model CNN for this task. However, using the same CNN architecture as before, no matter how hard we try to tune the training hyperparameters, we still even couldn't get an acceptable training accuracy(>30%), and the training loss remains high(Figure 1 in the appendix). We think that this is probably because of in the previous individual task, due to the limitation of our laptop(no GPU available), we designed a relatively simple ConvNet with small number of filters in each Conv Layer. With training across all subjects, we have much more different variety of data that our simple model may underfit with this larger dataset.

References

[1] Schirrmeister, Robin Tibor, et al. "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).

[2] Min, Seonwoo, Byunghan Lee, and Sungroh Yoon. "Deep learning in bioinformatics." Briefings in bioinformatics 18.5 (2017): 851-869.

[3] Tang, Zhichuan, Chao Li, and Shouqian Sun. "Single-trial EEG classification of motor imagery using deep convolutional neural networks." Optik-International Journal for Light and Electron Optics 130 (2017): 11-18.

[4] Fedjaev, Juri. "Decoding EEG Brain Signals using Recurrent Neural Networks."

[5] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. 2015.

Preprocess detail:

1. Data augmentation: Create several windows with window_size=300 and window_number=30 as default. Randomly choose 30 consecutive 300-time-bin-window from each raw trial data.
   Raw data: 288 x 1000 x 22 -> Augmented data: 8640 x 300 x 22
2. PCA to electrodes: Set n_components=10. Apply PCA to every trial on the electrodes dimension, reducing the dimension of 22 to 10.
   Augmented data: 8640 x 300 x 22 -> After PCA: 8640 x 300 x 10
3. (For FCNet & CNN)Randomly split 50 test data and 50 validation data from the whole raw 288 data in each trial, along with the data augmentation and PCA.
   Train: 5640 x 300 x 10, Validation: 1500 x 300 x 10, Test: 1500 x 300 x 10
4. (For RNN & LSTM)Wavelet Analysis: Randomly split 50 test data and 20 validation data from the whole raw 288 data in each trial. Use 3 level Wavelet, along with PCA operation and without data augmentation.
   Train: 218 x 254 x 10, Validation: 20 x 254 x 10, Test: 50 x 254 x 10

Model detail:

Conv Net:

Layer	Detail	Input	Output
1	24 x Conv1(12x5)	1x300x10	24x289x6
	Batchnorm	24x289x6	24x289x6
	Maxpool(2x1)	24x289x6	24x144x6
	Relu	24x144x6	24x144x6
	Dropout(0.5)	24x144x6	24x144x6
2	48 x Conv2(10x6)	24x144x6	48x135x1
	Batchnorm	48x135x1	48x135x1
	Maxpool(2x1)	48x135x1	48x67x1
	Relu	48x67x1	48x67x1
	Dropout(0.5)	48x67x1	48x67x1
3	Flatten	48x67x1	1x3216
	FC(500)	1x3216	1x500
	Relu	1x500	1x500
	Dropout(0.5)	1x500	1x500
4	FC(4)	1x500	1x4

FCNet:

Layer	Detail	Input	Output
0	Flatten	300x10	1x3000
1	hidden(800)	1x3000	1x8000
	Batchnorm	1x8000	1x8000
	Relu	1x8000	1x8000
	Dropout(0.5)	1x8000	1x8000
2	hidden(100)	1x8000	1x1000
	Batchnorm	1x1000	1x1000
	Relu	1x1000	1x1000
	Dropout(0.5)	1x1000	1x1000
3	hidden(4)	1x1000	1x4

LSTM(hidden units = 128):

Layer	Detail	Input	Output
1	Input Layer	254x217x10	254x20x10
	Hidden Layer	254x20x10	254x128x10
	Dropout	254x128x10	177x128x10
	Output Layer	177x128x10	177x20x10

RNN(hidden units = 128):

Layer	Detail	Input	Output
1	Input Layer	254x217x10	254x20x10
	Hidden Layer	254x20x10	254x128x10
	Dropout	254x128x10	177x128x10
	Output Layer	177x128x10	177x20x10

Training Detail:

1. (For FCNet & CNN)All the raw trials are extended to 30 sub trials after preprocess. Thus, for a single original trial, we will get 30 predictions. Here, we introduce a 'vote' process, choosing the most frequent predicted label in the 30 predictions as the final prediction of this trial.
2. (For RNN & LSTM)We use orthogonal initialization here. In order to avoid overfitting, the model will stop training when the validation loss keep increasing for 5 epoch and use the model at that time to test our test data.
3. The related hyper parameters and optimizers are shown below:

	Learning rate	Num epoch	Optimizer	Batch size
FCNet	1e-3	30	Adam(weight_decay=2e-2)	120
CNN	1e-4	50	Adam(weight_decay=2e-2)	120
RNN	1e-4	40	Adam(weight_decay=2e-2)	20
LSTM	1e-4	40	Adam(weight_decay=2e-2)	20

Result

File	FCNet	CNN	RNN	LSTM
A01T	30%	50%	40%	20.0%
A02T	18%	30%	32%	31%
A03T	38%	28%	20%	20%
A04T	32%	24%	40%	25%
A05T	28%	26%	20%	20%
A06T	44%	32%	50%	20%
A07T	34%	38%	50%	30%
A08T	42%	40%	26%	31%
A09T	28%	46%	44%	28%

Table 1. Validation Accuracy for All Models

File	FCNet	CNN	RNN	LSTM
A01T	32%	68%	28%	32%
A02T	30%	30%	38%	27%
A03T	44%	50%	38%	36%
A04T	32%	22%	26%	26%
A05T	28%	32%	31%	33%
A06T	34%	38%	22%	27%
A07T	34%	38%	32%	20%
A08T	34%	46%	31%	35%
A09T	42%	58%	26%	26%

Table 2. Test Accuracy for All Models

Window Size	100	200	300	400	500
Test Acc	48%	50%	54%	50%	48%

Table 3. Test Accuracy for different window sizes on subject A01

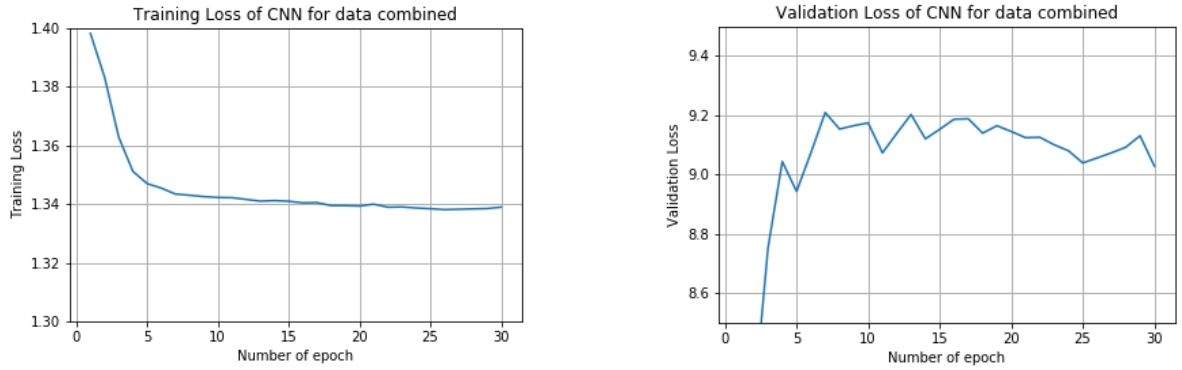


Figure 1. Train and Validation loss of CNN for Aggregated Subjects

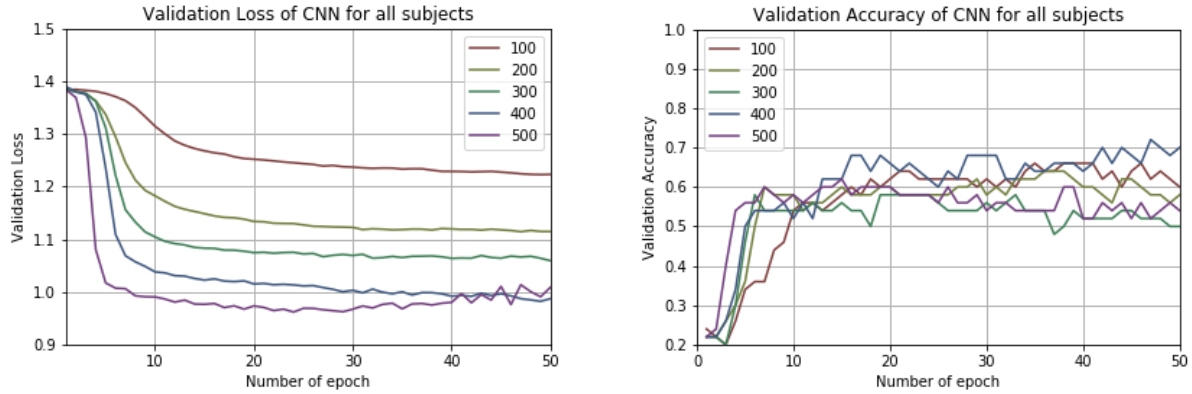


Figure 2. Validation Loss and Accuracy of CNN for All subjects