Movement Classification based on Electroencephalography (EEG) Signals
Final Project Report for ECE239AS at UCLA, Winter 2019

Ruchen Zhen
UID: 205036408
rzhen@ucla.edu

Sihan Wang
UID: 705029981
sihanwang@g.ucla.edu

Jingjing Li
UID: 805033360
ljj950428@g.ucla.edu

Abstract

In this project, we implement multiple neural network architectures for movements and subjects classification based on Electroencephalography (EEG) signals, including convolutional neural network (CNN), recurrent neural network (RNN) and generative adversarial network (GAN). Meanwhile, we try different data augmentation methods to improve the decoding accuracy. Moreover, we apply signal processing methods to filter and subsample the original signal, and obtained further validation accuracy improvement. We obtained 66.5% best validation accuracy for CNN, as well as 65.3% for RNN.

1. Introduction

1.1. Convolutional Neural Network

The first architecture we tried is the convolutional neural network. We consider trails as images of size 22*1000 and as inputs to the neural network. We built a model similar to AlexNet and also reproduced the model by [1] as a starting point. Both of them consist of 4 convolutional layers, which also contains pooling, batch normalization and dropout. The summary of the model is shown in appendix. In order to optimize the model performance, we swept over some important parameters, eg. learning rate, dropout rate, etc.

1.2. Recurrent Neural Network

Choosing RNN is intuitive because the EEG is a time series dataset. Both LSTM and GRU can be used to build the model. We tried both of them on two different implementations.

The first approach is to think of 1000 time series samples as a sequence. Then each time step has a feature vector of 22 dimensions. However, the 1000 time series sequence is too long for vanilla RNN because of vanishing gradient problem, even using LSTM cell. To solve this problem, we connected the first convolutional layer before RNN structure to shrink the sequence lengths so that the network could be effectively trained. The second approach is to think of electrodes dimension as "time". Another thing to note is that we used bidirectional layers in RNN model because we think the data is meaningful both ways.

1.3. Recurrent Neural Network with Attention

We also implemented RNN with attention mechanism (RNNA). The attention mechanism has been proved to be very useful in RNN structures. More concretely, it allows the output layers to look at all the hidden state vectors and then choose the most relative ones to make the predictions.

1.4. Generative Adversarial Network

The use of generative adversarial network in this classification task is not obvious at first glance. Ideally, after training, the discriminator should be able to produce effective, compact feature representations for each input, which could be used for other classification tasks. Thus, we trained a deep convolutional GAN and use it to extract feature representations for our tasks.

1.5. Data augmentation

Since the original dataset is small in size, we think having more data to train is going to help increase the classification accuracy. One way we tried is to increase the number of training data by slicing each trail with some length less than the trail length, so that we can get N-m+1 number of training samples per trail. Here N is the length of a trail and m is the window size that we sliced into. The prediction we finally used is calculated by the mean of 11 trails that originated by the single trail. Another way to do the prediction is taking the most frequent prediction from all 11 trails. One thing to note is that the input size to the neural networks after data augmentation is changed, because for each trail we only have length of 750, while other hyperparameters remained the same.

1.6. Optimization: Filtering and Subsampling

For time series signals, we always consider filtering out the noise to make signal distinguishable. Besides, data augmentation is an important approach to improve the performance for neural network, and it can be achieved by subsampling the original signal.

Savitzky-Golay filter helps smooth the data as well as filter out high frequency noise. Considering the case when applying CNN to movement classification, we use convolutional filters to detect wave crests inside the signals, i.e. the light pixel of the reconstructing image. Smoothing the data will fade the noise pixel in the reconstructing image, which makes the signal pixel more distinguishable. Moreover, by filtering out high frequency noise, it is helpful to subsample the raw data using a lower sampling frequency. We perform detailed discussion in section 2.5 and 2.6.

2. Results and Discussions

To evaluate models, we use classification accuracy and F1-scores as metrics. More detailed results are shown in the table attached at the end. In the discussion below we will mainly focus on classification over events because the accuracy over subjects is very high and would not be helpful when comparing different models.

2.1. Convolutional Neural Network

During evaluation we found that the model can be easily overfitting with high training accuracy (close to 1) and low validation accuracy (less than 40%). So we increased dropout rate (0.25). Average pooling has a slightly better performance than max pooling because there might be information loss in max pooling layer. This eventually gave us test accuracy of 65%, which is the better than many other models. Surprisingly, increasing the number of training data by slicing trails does not increase the test accuracy in our experiments. Since the input size is changed, we have to change the model accordingly. This might be the reason why we did not achieve better accuracy. Note that the accuracy is comparable, so if we have more time to tune parameters, we might be able to get better results.

2.2. Recurrent Neural Network

The RNN model achieved comparable results to the CNN model. We would expect RNN to do better because EEG is a time sequence dataset. However, the performance is actually a lot worse without the preprocessing convolutional layer. We believe this is due to the fact that 1000 samples are too long for LSTM to memorize. In fact, with vanishing gradient, this would cause the model learning nothing. The preprocessing convolutional layer shrinks the input to a reasonable size of 100*64, so that the LSTM layer could work properly.

2.3. Recurrent Neural Network with Attention

With the same preprocessing convolutional layer and bidirectional RNN structure, the model with attention mechanism seems to perform worse (accuracy of 47%). We think there are three main potential reasons for it. One is that the additional model capacity leads to overfitting problem. The second is that the attention is used only at the last step as output. Generally, it would be used in a decoder part of a sequence-to-sequence model through multiple output time steps. In other words, in our implementation, the use of attention mechanism is constrained. The third reason may be the input type. In our tasks, the inputs are so noisy and oscillating. It's hard to learn which part is more important than others.

2.4. Generative Adversarial Network

The classifier using feature representations by GAN achieves way lower accuracy than other models (only 35%), which is predictable. The training process of GAN is unsupervised and unstable, especially when generator and discriminator are implemented as deep neuron networks. So, the trained discriminator is not that powerful and the feature representations lose some information on the original inputs.

2.5. Optimization: Filtering the Raw EGG Signal

The following two subsections discuss the optimization in a signal processing perspective. In this subsection, we apply the Savitzky-Golay filter to the time series EEG signal, while in next subsection, we subsample the signal based on Nyquist Theorem to perform data augmentation.

By plotting the EEG signals in both time and frequency domains, we observe significant noise within the signal. As can be seen in Figure 1, some EEG signals are not largely affected by noise, such as the EEG signal from second electrode in 1st trial. However, other signals are overwhelmed by noise, which makes it difficult to recognize both in time domain and frequency domain. This may due to the environment difference when recording the signals in different trials, and this will significantly influence the classification. Smoothing the signal and filtering out the noises seems meaningful.

The two parameters of Savitzky-Golay filter matters the smoothing. With the increase of window size, zig-zagging in larger region will be ignored, which is similar to reduce the passband of low pass filter. Increasing polynomial order enables the filter to fit complex signal, but most of the case is noise. Our experiment shows that, when applying Savitzky-Golay filter to the raw data, both training and testing data, validation accuracy increases. Among them, the filter with window size equal 13, polynomial order equals 2, achieves the highest validation accuracy. The validation accuracy curve will be shown in the next subsection.

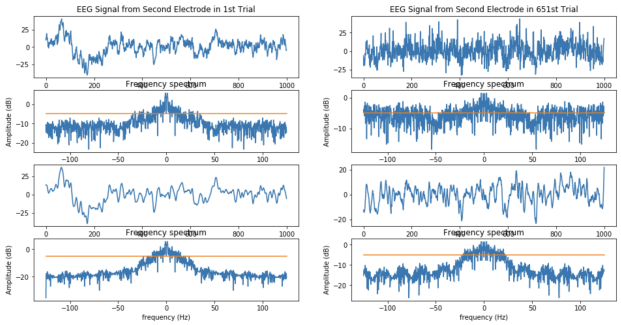

Figure 1. Time and frequency domain plots for 2 EEG signal examples. Left: EEG signal from second electrode, 1st trial. Right: EEG signal from second electrode, 651st trial. In top-down direction, time and frequency spectrum plots before applying Savitzky-Golay filter, time and frequency spectrum plots after applying Savitzky-Golay filter. Yellow horizontal line shows the -5dB amplitude threshold.

Best Validation Accuracy	CNN	RNN
Control Group	60.7%	57.8%
Filtering	63.7%	59.1%
Filtering + Subsampling	66.5%	65.3%

Table 1. Optimization for CNN and RNN: Best Validation Accuracy

2.6. Optimization: Subsampling the Raw Data

Applying Savitzky-Golay filter also enables us to make data augmentation. The original data is sampled with 250Hz sampling frequency. Based on Nyquist Theorem, we can sample a signal without aliasing if and only if the sampling frequency is larger than twice of the max bandpass frequency, and thus four times the baseband frequency.

Our idea is that, we can set a threshold, i.e. -5dB or -8dB. If the amplitude of a specific frequency is lower than the threshold, we treat it as noise. We want to figure of the frequency band of true EEG signal and thus decide the subsampling frequency. According to Figure 1, we can not subsample at all for signals like the right one in Figure 1, given that amplitudes exceed the threshold in the entire spectrum.

We use Savitzky-Golay filter, with best parameters we found, to filter out high frequency noise. We then check the maximal absolute frequency exceed the thresholds. The maximal absolute baseband frequency exceeds -5dB and -8dB threshold are 29.75dB and 68.75dB, respectively.

The optimization performance are shown in Figure 2, Figure 3, and Table 1. According to Figure 2 and Figure 3, filtering and subsampling lead to fast convergence and way-better validation accuracy. Table 1 shows the best validation accuracy for all cases. Applying filtering and subsampling improves the best validation accuracy by 9.5% for CNN, and 12.9% for RNN.

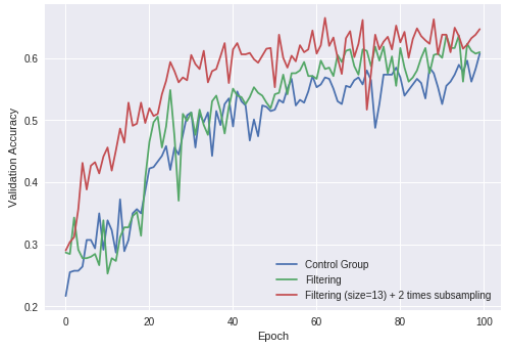

Figure 2. Optimization for CNN: validation accuracy curve for control group (applying neither filtering nor subsampling), applying filtering only and applying both filtering and subsampling.

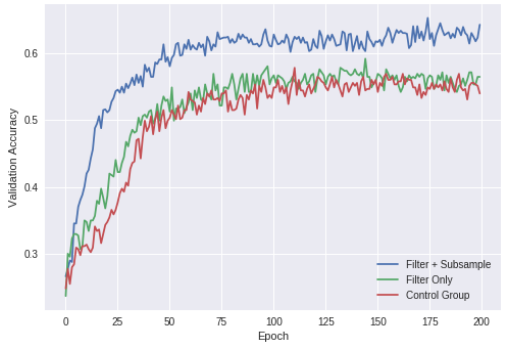

Figure 3. Optimization for RNN: validation accuracy curve for control group, applying filtering only and applying both filtering and subsampling.

2.7. Subject Classification vs Event Classification

The time needed for best accuracy is significantly longer for subject classification than for event classification. This is reasonable because there might be some unique signals that would show up in the last half second, not relative to the event, but are useful for determining subjects. The accuracy for subject classification is much higher than event classification. From this we can see that the signal is more unique for different subjects than for different events.

3. Conclusion

In this project, we perform movement and subject classification data based on EEG signals. We perform several architectures such as CNN, RNN, RNN with Attention and GAN. The validation accuracy are not remarkable. Two main reasons are limitation of dataset size as well as noise inside the EEG signal. Therefore, we performed data augmentation by subsampling the original signal, as well as filtering to EEG signals. The performance improved a lot, and we finally obtained 66.5% best validation accuracy for CNN, as well as 65.3% for RNN.

	Accuracy	Best Seq Position	F1-Score			
			769	770	771	772
CNN	66.5%	820	60.4%	57.8%	57.5%	67.5%
RNN	65.3%	890	56.5%	58.6%	51.5%	64.3%
RNNA	47.4%	700	44.8%	43.8%	44.0%	49.6%
GAN	35.2%	920	24.9%	17.6%	36.2%	35.2%

Table 2. Best validation accuracy for all models.

	Accuracy	Best Seq Position	F1-Score								
			1	2	3	4	5	6	7	8	9
CNN	96.8%	910	92.9%	99.0%	98.0%	96.2%	94.0%	92.3%	100%	96.8%	98.9%
RNN	98.9%	920	98.3%	94.2%	95.8%	98.0%	95.6%	99.0%	100%	96.1%	98.9%
RNNA	95.5%	990	91.3%	94.1%	95.8%	99.0%	97.8%	92.9%	99.0%	91.1%	98.9%
GAN	55.5%	890	40.8%	57.1%	79.6%	40.0%	44.7%	39.0%	37.9%	70.9%	79.6%

Table 3. Best training accuracy for all models.

References

[1] Robin Tibor Schirrmeister: Deep learning with convolutional neural networks for brain mapping and encoding of movement-related information from the human EEG. June 11,2018

CNN Architecture

Layer	Output Shape	Description
Input	22 * 1000 * 1	
Conv	22 * 496 * 64	64 filters of size 1*10 with stride (1,2)
Conv	1 * 496 * 64	64 filters of size 22*1 with stride (1,1)
Avg_Pool	1 * 165 * 64	1 * 3 filter with stride (1,3)
Dropout	1 * 165 * 64	Dropout rate 0.25
Permute	64 * 165 * 1	Change axes
Conv	1 * 156 * 128	128 filters of size 64 * 10 with stride (1,1)
Permute	128 * 156 * 1	Change axes
Norm	128 * 156 * 1	Batch Normalization
Avg_Pool	128 * 52 * 1	1 * 3 filter with stride (1,3)
Dropout	128 * 52 * 1	Dropout rate 0.25
Conv	1 * 43 * 256	256 filters of size 128 * 10 with stride (1,1)
Permute	256 * 43 * 1	Change axes
Avg_Pool	256 * 14 * 1	1 * 3 filter with stride (1,3)
Dropout	256 * 14 * 1	Dropout rate 0.25
Conv	1 * 5 * 512	512 filters of size 256 * 10 with stride (1,1)
Permute	512 * 5 * 1	Change axes
Norm	512 * 5 * 1	Batch Normalization
Avg_Pool	512 * 2 * 1	1 * 3 filter with stride (1,3)
Dropout	512 * 2 * 1	Dropout rate 0.25
FC	1024	Fully connected layer with 1024 units
Out	4/9	Softmax
Optimizer: Adam		

GAN Architecture

Generator		
Layer	Output Shape	Description
Input	11 * 8 * 256	
Conv_Transpose	22 * 40 * 128	128 filters of size 3 * 5 with strides (2,5)
Norm	22 * 40 * 128	Batch Normalization
Conv_Transpose	22 * 200 * 64	64 filters of size 3 * 5 with strides (1,5)
Norm	22 * 200 * 64	Batch Normalization
Conv_Transpose	22 * 1000 *1	1 filter of size 3 * 5 with strides (1,5)
Out	22 * 1000 * 1	tanh
Optimizer: Adam		
Discriminator: Same as CNN		

RNN Architecture

Layer	Output Shape	Description
Input	22 * 1000 * 1	
Conv	22 * 200 * 64	64 filters of size 1 * 5 with strides (1,5)
Conv	1 * 200 * 64	64 filters of size 22 * 1 with strides (1,1)
Avg_Pool	1 * 100 * 64	1 * 2 filter with stride (1,2)
Dropout	1 * 100 * 64	Dropout rate 0.25
Permute	100 * 64 * 1	Change axes
Squeeze	100 * 64	Delete axes
Bidirectional LSTM	512	Dropout rate and recurrent dropout rate 0.25 256 neurons
Out	4/9	softmax
Optimizer: Adam		