ECE247 Final Project - Classifications on EEG Signals

Nicolas Cheng
University of California
Los Angeles
nickycheng4@gmail.com

Ximeng Liu
University of California
Los Angeles
liux120@g.ucla.edu

Jingjie Wu
University of California
Los Angeles
wujingjie1996@gmail.com

Abstract

In this project, we worked on different deep-learning architectures for EEG signal classification. The EEG data are from 9 participants or subjects. There are 22 channels in the dataset, and each has a length of 1000 points. We aim to classify the data into 4 classes which corresponds to the human brain's reaction for the 4 different images.

In the project, we developed 3 different neural networks, namely CNN, RNN and CRNN. For each model, we used grid search to find the optimal hyperparameters and timing window for each model and validate them on the whole dataset as well as on each subject individually. To further improve the performance, we applied regularizations including parameter norms penalties, data augmentation and ensemble method. The results are given as follows. For the whole dataset, CNN achieves accuracy of 61% on validation and 63% on testing data. RNN has 60% on the validation and 55% on the test. CRNN reaches 71% on the validation and 66% on the test. Using ensemble method, CRNN finally achieves a testing accuracy of 71%.

1. Introduction

As the project focuses on the development of the deep learning algorithm, we will not dig into the EEG signals. Generally, we have developed 2 deep-learning models, which are CNN and CRNN[4] with different regularization, and finally we applied the ensemble method for the best model to obtain our final model with the highest accuracy. We will dive into these architectures in the following content section by section.

1.1. Regularization Strategies

We applied data augmentation in both models. This is because that during the development of our classifiers, we found that the training accuracy can always achieve more than 95%, but the validation accuracy is limited by 65%. We believe that this is due to the overfitting to the training data as we only have about 1700 samples for training (another 400 for validation). Thus, we divided the original data points into pieces with overlap to achieve data augmentation. For example, we could divide a original sample with 1000 points into three 800-point pieces with a step size of 10. Then, the pieces are corresponding to the index of 1-800, 11-810 and 21-820, and the points beyond 820 are discarded.

Besides, we utilized ensemble method for our final model. The reason behind it is that the variance of the training process is obvious even though the same hyperparameters are used. The validation accuracy varies with a range of about 5% and the testing accuracy diverses with a range of 10%. Thus, we introduced an ensemble method with a factor of 6 to reduce the variance and error on the testing accuracy.

Additionally, we trained and validated on different window sizes and picked the best one. To find optimal hyperparameters, we leveraged grid search for different settings including parameter norm penalties (None, L1, and L2) with different factors, optimizers (Adam, RMSprop, and Nesterov), learning rates, decay factors, batch sizes and epochs. In practise, we split the given train_val_data into train_data and val_data with a ratio of 4:1 for validation purposes.

1.2. Convolutional Neural Network

Although CNN is primarily developed for image classification, it is still worth trying in this signal classification problem. This is because the CNN utilizes spatial localized linear filters and this can help us extract features within each neighbourhood of sample points. The pooling units inside CNN also provide a solution to the sampling uncertainty and noise. Besides, our EGG signals have 22 channels and CNN provides filters to efficiently combine them together.

In the model, we treated the input data as a 2D array with a unit width, a window-size length, and 22 channels. For example, an input data is reshaped from 1000x22 into 1x1000x22. Since the volume of the training set is limited, making CNN too many layers may lead to the overfitting issue. As a result, we implemented 5 layers in the CNN model. It has 2 convolutional layers with a max-pooling

layer in between. They are followed by 2 fully-connected layers and a softmax function as our final activation function. Each layer contains a batch normalization filter and an activation function of ReLU. *See Appendix B.1.*

1.3. Recurrent Neural Network

As LSTM is good at extracting patterns in input feature space, where the input data spans over long sequences, it is supposed to perform well for the EEG data that spans on a long time duration.[2, 3] Therefore, in this section we chose RNN model as the other candidate.

The input of RNN is in the shape of 500x22 as the model performs the best at time duration of 500. The first layer is a fully connected layer with 16 units to combine the 22 channels together. After that, it gets permuted and input to another FC layer for data compression. Then, we have a LSTM layer as RNN core and followed by another two FC layers with dropout.An activation function of Softmax is used. *See Appendix B.2.*

1.4. Convolutional Recurrent Neural Network

As CNN and RNN are both used in the previous sections, and they each have advantage on feature extraction and input feature space pattern, we decide to further explore the combination of these two architectures, which is the CRNN.

The input is treated as 2D array with a width of 22, a length of the window size, and a channel number of 1. The CRNN model has three to four convolutional layers depending on the training time (epoch number). If the training data size is large enough, a fourth layer with 128 1x10 filters will be added. Max-pooling layerd are inserted in between. After theym, there are three bidirectional long short term memory (LSTM) layers. After the LSTM layers, a dropout layer is added before the final fully connected layer. *See Appendix B.3.*

2. Results

We developed the models following the same procedure. For each model, we developed the optimal architecture with the data from all 9 subjects and then evaluated it with individual subjects in addition to the whole dataset. In the practise, we firstly developed a baseline after a few manual choices of hyperparameters. Then, we tested the model with different window size with and without data augmentation. If there is an inconsistency between the two settings, we would prefer the one with data augmentation since we would use it for the final model.

After the optimal size is determined, we applied grid search as described in the previous section. Then, with the optimal window size and hyperparameters, we trained the model with a heavier augmented dataset (e.g. we used 3x augmentation in the grid search but would use 10x or 20x in this section) and evaluated the model with the testing data from all 9 subjects.

Once all the three architectures were evaluated, we picked the model with the highest validation accuracy and applied ensemble method on it. With the top 2 settings, we trained 3 copies of each model and evaluated them with the testing data. We recorded and averaged the scores from the 6 models and used the mean values to determine the prediction result. Finally, we computed the final testing accuracy from the result.

In addition to evaluating models with the whole dataset, we evaluated them with data from individual subjects as well. Furthermore, we re-trained the models for each individual subject, and then evaluated and compared the individual testing accuracy.

2.1. CNN

The baseline CNN model has no regularization and the optimizer is the default Nesterov. It is trained with a batch size of 50 and 10 epochs. The validation accuracy is 44.7% and testing accuracy is 47.4%. Once we applied augmentation of 5x, the validation accuracy is 60.5% and testing accuracy is 59.4%.Determining the optimal window size in Table.1, we found the optimal size for original data is 700, but the one for augmented data is 900. Since we are going to use data augmentation, we picked 900 as our window size. In the grid search, we got the highest validation accuracy of 60.6% with the optimal hyperparameters and regularizations.

In the final model of CNN, we did data augmentation of 20x with the optimal hyperparameters discovered ahead. For the optimal model, the validation accuracy is 60.6% and the testing accuracy is 62.5%. As we tested the general model on individual subjects, the highest accuracy is 74% and the lowest accuracy is 48% . Then, we re-trained the model with data from individuals only, and the accuracy is a range from 44% to 81%. The details are given in the table in the Appendix section.

2.2. RNN

The baseline RNN architecture uses epoch number 20, batch size 64 with no regularizer. It achieves a testing accuracy of 31.6%. With augmented data of 4x, testing accuracy gets improved to 46.3%. When optimizing for the best time window size, as given in Table.2, for dataset without augmentation 400 is the best time window, and for dataset with augmentation best time window is 500. Therefore, we chose 500 as the best time window for further grid search, and we got the best validation accuracy of 60.1% and test accuracy of 54.8%.

As for individual subjects, we first trained a model based on the whole dataset and tested it on each individual subject. The highest prediction accuracy is subject 4 with

Table 1: CNN Acc. vs. Window Size

Win Size	Acc w/o Aug.	Acc w/ Aug.
100	0.42	0.30
200	0.44	0.33
300	0.48	0.35
400	0.49	0.45
500	0.49	0.44
600	0.47	0.45
700	0.50	0.49
800	0.48	0.54
900	0.46	0.56

Table 2: RNN Acc. vs. Window Size

Win Size	Acc w/o Aug.	Acc w/ Aug.
100	0.37	0.41
200	0.38	0.47
300	0.32	0.51
400	0.39	0.50
500	0.36	0.54
600	0.38	0.53
700	0.38	0.50
800	0.34	0.51
900	0.30	0.48

Table 3: CRNN Acc. vs. Window Size

Win Size	Acc w/o Aug.	Acc w/ Aug.
100	0.46	0.41
200	0.50	0.53
300	0.52	0.55
400	0.64	0.66
500	0.61	0.64
600	0.62	0.66
700	0.62	0.69
800	0.64	0.66
900	0.64	0.67

70%, while the lowest is subject 1's 46%. After that, we trained different models based on each individual's dataset and tested them on their own test sets. In this way, the highest accuracy is subject 8's 78% and the lowest is subject 1 and 2's 30%.

2.3. CRNN

The baseline of CRNN uses arbitrary hyperparameters of: epoch number of 40, batch size of 40, Adam as optimizer, and no regularizer. For this combination, we obtained testing accuracy of 60.3% and a validation accuracy of 57.6%. With augmentation of 3x, the validation accuracy is 61.0% and the testing accuracy is 61.1%. To determine the best time window size, we scanned through time duration from 100 to 1000. From the Table.3, we found that with data augmentation, 700 is the optimal time window followed by 900, which provides validation accuracies of 68.7% and 67.0%. Here we ignored the result without data augmentation as it provides less accuracy.

After that, we did a grid search on both of these two time windows and it showed that there are two sets of hyperparameters, one with L1 Norm and the other with L2 Norm, having similar performance of a validation accuracy of 71.2%.

As the CRNN model with data augmentation provides the best validation accuracy among the three neural networks, we further improved the model by applying ensemble method. We trained two models with the two optimal sets of hyperparameters with 10x augmentation and got a ensemble testing accuracy of 65.9%. Then, we trained 3 copies for each setting, ensembled the 6 models, and obtained a testing accuracy of 70.8%. When we tested on each subject individually, the testing accuracy ranges from 56% to 80% with the general model and it ranges from 50% to 85% with re-trained models on each subject.

3. Discussion

The Improvement of switching from CNN and RNN to CRNN is obvious, where the difference is about 10%. This is because the convolutional layers in CRNN extract the local features from the multi-channel signals, and also the LSTM layers improve the performance on time series based data [1]. Thus, we concluded that CRNN is very suitable for EGG signals as it can catch both the short-term and long-term features.

It is a common understanding that neural networks are data eager and they require high volume of data to train a reliable model. In the project, however, we only have about 2000 samples and 1/5 of which needs to be seperated for validation, so it is very hard to train a reliable model with the original data. Due to this consideration, we applied data augmentation to assemble a larger training dataset to fully train the model. According to the result, the improvement is significant as it improves about 10% for CNN and RNN and 5% for CRNN. Besides, it also helps reduce the model's variance. Without augmentation, the model has very different performance every time we re-trained it, but augmentation reduces this diversity in the training.

Another noticeable fact is that the ensemble method significantlyy improves the performance. As we trained 6 CRNN models with optimal hyperparameters, accuracies of them ranges from 64% to 69%. However, ensemble method takes the average score of predictions from the 6 models and provides a better testing accuracy of 70.8%. Thus, ensemble method is an easy but useful regularization to improve accuracy while reducing the model variance as well.

In addition, we compared the performance of the model trained by all subjects with the one trained by each subject individually. An interesting fact is that in general, the individual models perform a higher accuracy than the general models but the diversity of accuracy on subjects is higher. We believe this is due to the personal diversity and the issue of overfitting. As the data is more consistent on one subject compared to crossing subjects, the individual model is more suitable for itself. However, since the quantity of training data for one subject is quite small, the model is overfitting to the training data and downgrades the generalization.

References

[1] J. T. Connor, R. D. Martin, and L. E. Atlas. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2):240–254, 1994.

[2] N. F. Güler, E. D. Übeyli, and I. Güler. Recurrent neural networks employing lyapunov exponents for eeg signals classification. *Expert systems with applications*, 29(3):506–514, 2005.

[3] X. Li, D. Song, P. Zhang, G. Yu, Y. Hou, and B. Hu. Emotion recognition from multi-channel eeg data through convolutional recurrent neural network. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 352–359. IEEE, 2016.

[4] D. Zhang, L. Yao, X. Zhang, S. Wang, W. Chen, R. Boots, and B. Benatallah. Cascade and parallel convolutional recurrent neural networks on eeg-based intention recognition for brain computer interface. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

A. Performance Summary

Here we summarize all the models and strategies we tried. The first part is the model trained, validated and tested with the whole data set from 9 subjects. We can see the validation and testing accuracy improved as more strategies are applied. Finally, we applied ensemble method to the best CRNN model and reached the highest testing accuracy of 71%.

The second and the third parts show the testing accuracy on the individual subjects. The second section uses the optimal models trained with the data from all 9 subjects and tests on individual subjects. While, the third section retrains the optimal models with the individual data and tests themselves.

A.1. General model accuracy on all subjects

Strategy	CNN	RNN	CRNN
Baseline (val)	0.45	0.36	0.60
Baseline (test)	0.47	0.32	0.58
Data Augmentation (val)	0.60	0.46	0.61
Data Augmentation (test)	0.59	0.46	0.61
Grid Search (val)	0.61	0.60	0.71
Grid Search (test)	0.63	0.55	0.66
Ensemble (test)	N/A	N/A	**0.71**

A.2. General model accuracy on single subject

Subject	CNN	RNN	CRNN
1	0.48	0.52	0.64
2	0.54	0.46	0.56
3	0.74	0.54	0.72
4	0.74	0.58	0.80
5	0.56	0.70	0.59
6	0.58	0.51	0.67
7	0.66	0.52	0.72
8	0.70	0.48	0.72
9	0.54	0.64	0.72

A.3. Individual model accuracy on single subject

Subject	CNN	RNN	CRNN
1	0.66	0.54	0.78
2	0.48	0.30	0.5
3	0.50	0.30	0.82
4	0.44	0.36	0.62
5	0.66	0.66	0.74
6	0.49	0.43	0.55
7	0.72	0.64	0.8
8	0.72	0.44	0.76
9	0.81	0.79	0.85

B. Architecture Summary

B.1. CNN Architecture

CNN	
Input: [1x900x22] with 20x data augmentation	
Layers	
[1x893x32] Conv Layer	32 1x8 Filters ReLU & Batch Norm
[1x223x32] MaxPool	1x4 stride 1
[1x216x64] Conv Layer	32 1x8 Filters ReLU & Batch Norm
[13824] Permute and Flatten	-
[64] FC Layer	13824x64 units & L1 ReLU & Batch Norm
[4] FC Layer	64x4 units & L1
Activation Function	SoftMax
Training Details	
Optimizer	Nesterov lr=0.01 & decay=1e-6
Batch Size	40
Epochs	20

B.2. RNN Architecture

RNN	
Input: [500x22] with 8x data augmentation	
Layers	
[500x16] FC Layer	22x16 Units & L1 Batch Norm & ReLU
[16x500] Permute	-
[16x256] FC Layer	500x256 & L1 Batch Norm & ReLU
[16x128] Bidirectional LSTM	64 Units
[2048] Flatten	-
[64] FC Layer	2048x64 units & L1 Batch Norm & ReLU
[64] DropOut	-
[32] FC Layer	64x32 units & L1 Batch Norm & ReLU
[32] DropOut	-
[4] FC Layer	32x4 units & L1
Activation Function	SoftMax
Training Details	
Optimizer	RMSprop lr=0.001 & decay=0.01
Batch Size	48
Epochs	20

B.3. CRNN Architecture

CRNN	
Input: [22x900x1] with 10x data augmentation	
Layers	
[22x891x16] Conv Layer	16 1x10 Filters & L1L2 Batch Norm & ELU
[2x891x32] Conv Layer	32 21x1 Filters & L1L2 Batch Norm & ELU
[2x222x32] MaxPool	1x4 Stride 1
[2x213x64] Conv Layer	64 1x10 Filters & L1L2 Batch Norm & ELU
[2x53x64] Max Pooling	1x4 Stride 1
[2x44x128] Conv Layer	128 1x10 Filters & L1L2 Batch Norm & ELU
[2x11x128] MaxPool	1x4 Stride 1
[2x11x128] DropOut	-
[11x256] Permute and Flatten	-
[11x256] Bidirectional LSTM	128 Units
[11x128] Bidirectional LSTM	64 Units
[64] Bidirectional LSTM	32 Units
[64] DropOut	-
[4] FC Layer	64x4 units & L1L2
Activation Function	SoftMax
Training Details	
Optimizer	Nesterov lr=0.001 & decay=0.001
Batch Size	48
Epochs	40
Ensemble	3 models with L1 of 0.001 plus 3 models with L2 of 0.01