Classification of EEG Dataset with Multiple Neural Networks

Weinan Song
UCLA
CA 90095 US
wsong@ucla.edu

Ying Wang
UCLA
CA 90095 US
yingw4@uw.edu

Tianxue Chen
UCLA
CA 90095 US
tianxuechen@ucla.edu

Abstract

As the increasing popularity and decreasing cost of EEG based devices, automated analysis of EEG data is currently in the stage of seeking higher accuracy predicting user's brain activities. This leads to one of the sub problem that classify brain activities based on EEG raw data. In this paper, we would discuss the methods of solving the problem. We have applied multi-method in order to classify the given 4 labels, given the results of at least above 55% test accuracy on all networks we present in this paper, and the highest test accuracy is 68.5% using stack CNN with RNN.

1. Introduction

Electroencephalography (EEG) is a noninvasive method to measure human brain singles, which embedded rich information of human thoughts. And the object of this work is to classify human being's imagination activities based on EEG raw data.

Deep learning is an artificial approach of training the input data with parameters to lead a better classification or prediction on new data. Current popular networks are CNN and RNN under various conditions. As CNN using connectivity pattern between its neurons and RNN using time-series information, we proposed both networks and combine these two networks to do the classification and compared the performance.

Convolution neural networks(CNN) has shown great performance in classifying images since [2]. Many problem in computer vision can be efficiently solved based on CNN. In this paper, we treat the sequence data as images and change the problem into a basic image classification problem.

Recurrent neural networks(RNN) is an efficient algorithm which can be used in dealing with sequence data. The algorithm has proved it performance in processing natural language processing(NLP) like [4]. In this paper, we also change this problem into a NLP problem by taking each 22x1 vector as a word vector and the whole sequence as a word.

This document enclosed the network architecture comparing for the EEG classification problem mentioned above. The paper will be presented in the following arrangement. Section 2 presents the loss, train and validation accuracy as a function of time and the test accuracy for four model. Section 3 shows some insight discussion based on why we choose the model and how to lead the result.

2. Results

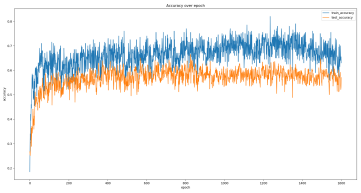
According to the characteristics of EEG dataset, we attempted different models to challenge to gain a better classifier, as shown in the Appendix of Method: one-dimensional deep CNN, two-dimensional CNN, stack CNN before RNN, and stack CNN after RNN. As for comparing the performance of various network, the following section present the test results across 9 dataset. For detailed accuracy on each subject, please refer to the Appendix of Table of Performance.

2.1. Data preprocessing

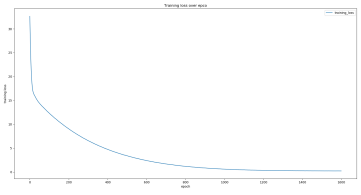
The EEG data was collected by 25 electrodes, where 22 EEG channels and 3 EOG channels from 9 subjects, and for each dataset there are 288 trails of four different motor imaginations. As the first step of preprocessing, the trail with Nan data was deleted and the EOG channels are extracted out. As for splitting training data and testing data, we just simply concatenated 50 trails from each subject as testing data and splitted the remaining trails as training data and validation data with a proportion 0.8.

2.2. 1D deep CNN

Fig 1a and Fig 1b show the training performance of one-dimension, five-layer CNN. The best validation accuracy of this model is 66.4% with training accuracy around 82%. The model could lead the validation accuracy around 60% within 200 epochs. This could result in 57.78% test accuracy across 9 dataset.



(a) Train accuracy

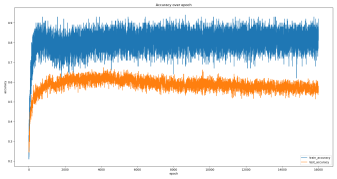


(b) Train loss

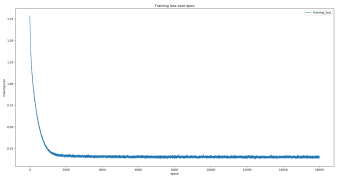
Figure 1: Training performance of 1D CNN

2.3. 2D CNN

Fig 2a and Fig 2b show the training performance of 2-dimension CNN. The best validation accuracy of this model is 67.5% with training accuracy around 90%. The model will give validation accuracy around 60% within 200 epochs, and the test accuracy is 55.11% across 9 dataset.



(a) Train accuracy

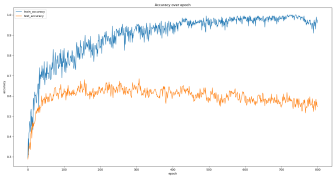


(b) Train loss

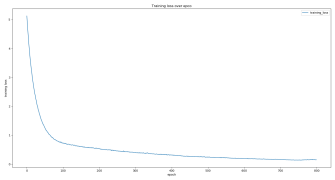
Figure 2: Training performance of 2D CNN

2.4. Conv-RNN

The first RNN structure that we test is a simple one layer LSTM network. After 1000 epochs, the training accuracy keep near 50%. Inspired by the performance of CNN and RNN, we combine these two structures together to form a Conv-RNN architecture. Fig 3a and Fig 3b shows the training accuracy and training loss of this architecture. The best validation accuracy of this model is 68.5% with training accuracy around 99%. The model will give validation accuracy around 60% within 50 epochs, and finally get 68.54% test accuracy across 9 dataset.



(a) Train accuracy



(b) Train loss

Figure 3: Training performance of Conv-RNN

2.5. RCNN

After applying convolution layer before RNN, we designed a model which takes the output of RNN into a convolution layer. Fig 4a and Fig 4b show the training performance of RCNN. The model we designed has the best validation accuracy of 62.7% with training accuracy around 80%. The model will give validation accuracy around 60% within 200 epochs, and will get 62.68% test accuracy across 9 dataset.

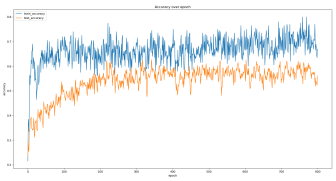
2.6. Result on single dataset

For single dataset, we train a single dataset on 1D deep CNN. Fig 5a and Fig 5b show the training performance on single dataset. The best validation accuracy of this model is 65% with training accuracy around 90%. After trained this model we test this model on testdata from other subjects. The results is given in Table 6.

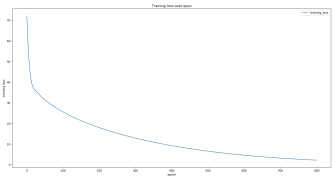
3. Discussion

3.1. Comparison between 1D deep CNN and 2D CNN

Since the dataset records the signal from different position, single channel produce a one dimension data. Thus, if we assume every channel is independent, we can use 1D deep CNN to do classification.

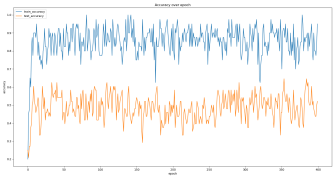


(a) Train accuracy

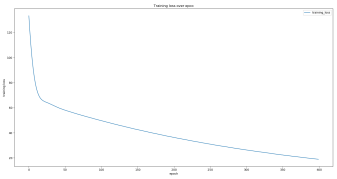


(b) Train loss

Figure 4: Training performance of RCNN



(a) Train accuracy



(b) Train loss

Figure 5: Training performance on single dataset

Further more, we consider that different electrode position may have relationship with others. Thus for 2D CNN, we take the input signal as a 22×1000 image and design a CNN to do typical image classification. We use 1D and 2D convolution kernel to do convolution, followed by batch normalization and drop-out layer, the latter of which is used to reduce over-fitting. In order to better express the feature of the input image, we first do a 1D convolution on the time sequence, that is after convolution, the point in the output indicate the overall feature in 0.004 seconds. After that, we apply two 2D convolution layers on the output to combine features between channels and time sequence.

An interesting result is that the experiment shows that 2D CNN has slightly better results than 1D deep CNN, which may indicate the independence of 22 channel.

3.2. Comparison between CNN and RNN

We want to adapt RNN for several reasons. First of all, unlike CNN, which is a type of feed-forward neural network, RNN uses sequential information, with the output being depended on the previous computations. More specifically, the network processes a sequence of vectors x by updating recurrent hidden units at every time step. Secondly, the EEG dataset is recorded in a continuously timeslot. Given the property of this dataset, we apply RNN to get better performance. However, simply using RNN did not give us very better results. Other people also give the same conclusion[1].

Considering CNN could help to extract features of input, we first tried to add multiple Conv1d layers before LSTM layers so that we can use CNN to pre-process the input data. We then designed a model which use CNN after LSTM layers. The input of the CNN layer is the 1000 hidden state matrices produced by RNN. The CNN then can use this information to classify multiple actions. Given the results state above, the test accuracy of CRNN is 6% higher than RCNN. We then analysis that the reason why CRNN is better than RCNN. Conv1d layer can exact local information from neighboring time points, which is the step to learn temporal dependencies. In addition, the Conv1d layer could also decrease the time sequence, which will improve computational efficiency. Similar structure also has been proved to be efficient in [3].

3.3. Accuracy based on different dataset

From Table 6 we can tell that training a model on single dataset and test it on other subject may not give us good results. Since different subjects indicate different people, we may tell that people have unique way of thinking. In that case, using one dataset to train a model may not be able to capture general features in all subjects.

4. Conclusion and Further work

Classifying brain activities on EEG signals is often the first step of brain learning. We compare four models based on CNN and RNN to do a four-label classification. According our experiment, the best model is CNN stacking before RNN, which could lead test accuracy 68.54%. For now, we did not do any preprocess to the data, except removing Nan. We have tried used Fourier Transform to preprocess the input data; however, it did not improve the training or validation accuracy, so we just abandon that approach. In the future, we could do data augmentations and down-sampling to increase training dataset to search a better results. With unknown knowledge of the independence between each electrode, another further direction is to extract the feature of electrode montage information. More specifically, we could expand the 2-dimensional EEG signal to 3-dimensional[5]

Appendix of Method

5. Model 1: CNN

Since the convolution neural network is basically use to extract features, we firstly applied 1D CNN (Table 1) and 2D CNN (Table 2)separately.

5.1. 1D deep CNN

For 1D CNN, we take the input as an image with 22 channels. As the data is related to time sequence, we want to use convolution layer to condense time information. We also add batchnorm to avoid gradient vanish and explosion. We add pooling layer after each batchnorm and double the channels to keep the information.

Table 1: 1D deep CNN Architecture

Layer	Operation
$32 \times Conv1D(1, 37)$	ReLU BatchNorm Dropout (0.2) MaxPool2D(2,2)
$64 \times Conv1D(1, 31)$	ReLU BatchNorm Dropout (0.2) MaxPool2D (2, 2)
$128 \times Conv1D(1, 27)$	ReLU BatchNorm Dropout (0.2) MaxPool2D (2)
$256 \times Conv1D(1, 25)$	ReLU BatchNorm Dropout (0.2) MaxPool2D (2)
$512 \times Conv1D(1, 9)$	ReLU BatchNorm Dropout (0.2) MaxPool2D (2)
Linear	
Softmax	

5.2. 2D CNN

For 2D CNN, we take the input signal as a 22×1000 image and design a CNN to do typical image classification. We use 1D and 2D convolution kernel to do convolution, followed by batch normalization and drop-out layer, the latter of which is used to reduce over-fitting. In order to better express the feature in the input image, we first do a 1D convolution on the time sequence, that is after convolution, the point in the output indicate the overall feature in 0.004 seconds. After that, we apply two 2D convolution layers on the output to combine features between channels and time sequence.

Table 2: 2D CNN Architecture

Layer	Operation
$16 \times Conv2D(1, 22)$	ReLU BatchNorm Dropout (0.2)
$32 \times Conv2D(2, 32)$	ReLU BatchNorm Dropout (0.2) MaxPool2D (2, 2)
$64 \times Conv2D(8, 4)$	ReLU BatchNorm Dropout (0.2) MaxPool2D (2)
Linear	
Softmax	

6. Model 2: RNN

6.1. Basic RNN

Considering the input is a time series, a simply approach is to use one-layer LSTM. However, with the experiments, we found that a larger number of LSTM memory cells improved training accuracy well, but prone to lead overfitting, a dropout layer with dropout rate 0.25 is added after the LSTM layer, which is a solid approach to avoid overfitting. The output layer uses a softmax activation to compute the probabilities of each class to obtain the prediction of the input.

6.2. CRNN

Table 3 shows the detailed architecture of three 1-dimensional CNN stacking before RNN.

6.3. RCNN

Table 4 shows the detailed architecture of five 1-dimensional CNN stacking after RNN.

Table 3: CRNN Architecture

Layer	Notes	Operation
$32 \times Conv1D(1, 37)$	$padding = 0, stride = 1$	ReLU BatchNorm Dropout (0.2)
$64 \times Conv1D(1, 31)$		ReLU BatchNorm Dropout (0.2) MaxPool2D (2, 2)
$128 \times Conv1D(1, 27)$		ReLU BatchNorm Dropout (0.2) MaxPool2D (2)
LSTM	$hidden_size = 128, num_layer = 1$	
Linear		
Softmax		

Table 4: RCNN Architecture

Layer	Notes	Operation
LSTM	$hidden_size = 22, num_layer = 1$	
$32 \times Conv1D(1, 37)$	$padding = 0, stride = 1$	ReLU BatchNorm Dropout (0.2)
$64 \times Conv1D(1, 31)$		ReLU BatchNorm Dropout (0.2) MaxPool2D (2, 2)
$128 \times Conv1D(1, 27)$		ReLU BatchNorm Dropout (0.2) MaxPool2D (2)
$256 \times Conv1D(1, 27)$		ReLU BatchNorm Dropout (0.2) MaxPool2D (2)
Linear		
Softmax		

Appendix of Table of Performance

Table 5: Performance on different Algorithm

Model name	Validation ACC	Test Acc
1D deep CNN	66.4%	57.8%
2D CNN	67.5%	55.1%
Vanilla RNN	53.1%	51.4%
Conv-RNN	68.5%	63.6%
RCNN	62.7%	56.7%

Table 6: Accuracy on different test dataset

test dataset name	Test Accuracy
$Sub1$	60%(trained model)
$Sub2$	20%
$Sub3$	52%
$Sub4$	42%
$Sub5$	46%
$Sub6$	44%
$Sub7$	36%
$Sub8$	34%
$Sub9$	46%

References

[1] A. S. Greaves. Classification of eeg with recurrent neural networks. 2014.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[3] S. Roy, I. Kiral-Kornek, and S. Harrer. Chrononet: A deep recurrent neural network for abnormal eeg identification. *arXiv preprint arXiv:1802.00308*, 2018.

[4] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[5] D. Zhang, L. Yao, X. Zhang, S. Wang, W. Chen, and R. Boots. Eeg-based intention recognition from spatio-temporal representations via cascade and parallel convolutional recurrent neural networks. *arXiv preprint arXiv:1708.06578*, 2017.