# EGG Motor Imagery Tasks Classification Using Neural Network

Gongjie Qi
905429380
gongjie@g.ucla.edu

Chenyu Wang
805436446
chenyuwang814@ucla.edu

Yifei Chen
705444102
cyf@ucla.edu

Gaofang Sun
104853165
gaofang56@gmail.com

## Abstract

*In this project, we designed neural networks combining Convolutional Neural Network, Gated Recurrent Unit and Long Term Short Memory. We applied them to the electroencephalography (EEG) data and tested each of their performances in predicting the four classes of imagery tasks classification. The four classes includes left hand (class 1), right hand (class 2), both feet (class 3), and tongue (class 4). Further more, preprocessing technique is applied on the time series data.*

## 1. Introduction

According to *BCI*[1], The training data provided are 2115 trials of time series data of 1000 data points from 22 different electrodes installed on subjects' scalps. The outcomes are 4 classes of motory imaginary tasks. There are 9 subjects in the dataset. Inspecting the dataset, we decided to construct the following neural networks to make prediction.

### 1.1. Data Pre-processing

The 1000 data from each electrodes are time series. We see that the data sampled appears to have a large scale of noise. To mitigate the error introduced to the classification accuracy, we added a Savitzky-Golay filter to reduce noise and smoothen the time series data in both training set and the test set. Example filtered data is shown in Figure 1.

In addition to filtering the data, data augmentation is also applied to improve the regularization. We chose augmentations on the time-series data including Jittering (adding Gaussian noise to the data), Scaling (multiply the data by a scaling factor) and Down-Sampling (Repeating data at odd or even indices). In the end we have the augmented training set in $5 \times 2115$

The example augmented data with different approaches are shown in Figure 2.

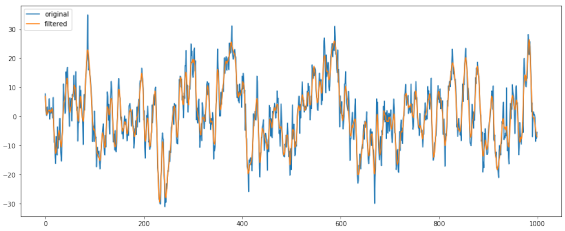


Figure 1. Filtered data compared to original data: Orange line represents the filtered data and the blue line represents the original data.

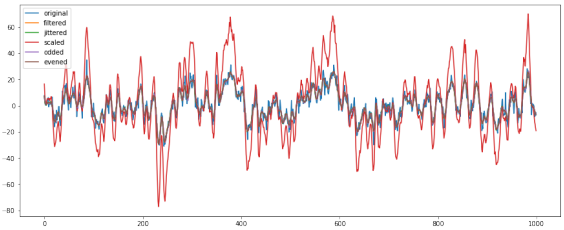


Figure 2. Augmented time-series data: The data of different type of augmentation are as labeled.

### 1.2. 1-D Convolutional Neural Network

The architecture of our naive CNN model is shown in Figure 7. This model consists of 3 convolutional parts and 3 fully-connected layers, the dimension of the first convolutional layer is 22 corresponding to the 22 channels of the signal. Each convolutional part includes a 1D convulutional layer, a batchnorm layer, an Exponential Linear Unit (ELU) activation layer, a 1D maxpool layer and a dropout layer.

Since the input data is a time sequence signal, the features between time steps are relevant, thus the first 3 convolutional parts can effectively extract and learn such features. Then the fully-connected layers can map the features into the one-hot vectors for the classification task.

### 1.3. Long short-term memory (LSTM)

The architecture of the LSTM model consists of 3 bidirectional LSTM layers and a fully connected layer. Since the input data is a time series, we try to use RNN to deal with the classification task which may be better than CNN. Because RNN is able to extract information more effectively along time steps dimension than CNN.

### 1.4. Convolutional Neural Network combined with LSTM

The architecture of the combined neural network is shown in Figure 6, which combines our CNN and LSTM model to get better performance. The proposed model consists of 3 convolutional parts, 3 bidirectional LSTM layers and a fully-connected layer sequentialy.

In this model, the CNN part can extract features between time steps first which can reduce the difficulty of LSTM from directly extracting features from such a long time series with highly correlation between adjacent time steps.

We get rid of the dropout layer of the third convolutional part and add one after the LSTM layers. In this way, we can reduce the information loss of the input for LSTM layers.

### 1.5. Convolutional Neural Network with GRU

As shown in Figure 8, The architecture of this model is similar to the previous model , we just change the LSTM to GRU since it is easier for us to train a GRU than an LSTM due to the lack of the cell state to see if there will be any increase to the performance.

## 2. Results

The results measured with different criteria are shown in the following section.

### 2.1. Data Pre-processing

The resultant test accuracy of models using raw data, SavGol filtered data and augmented data are presented in Table 1.

### 2.2. Accuracy for each subject

The training and test accuracy of each subject is presented in Figure 3., Figure 4 and Table 2. The model used is CNN+LSTM and the data used is smoothed without data augmentation.

### 2.3. Accuracy across all subjects

Data across all subjects are run with different models, for 40 epochs. The resultant accuracies are presented in Table 3. The data here is filtered data, without any data augmentation.

### 2.4. Accuracy as a function of time

Training and testing accuracy on different time period of the time-series data is presented in Figure 5. The model used is CNN+LSTM and the data is filtered without augmentation

## 3. Discussion

In this section, we discussed our method for data augmentation and our consideration on how we optimize our model. We also shown the performance of our model on different subjects and time duration.

### 3.1. Effect of Data Augmentation

From the results table shown in 2.1, we observe that the Savitzky-Golay smoothing filter shows general improvement for all models. This proves that denoising on the data before training is helpful on improving classification.

However, after augmentation, there is less improvement on test accuracy except for naive 1-d CNN. One possible reason is that, since all five portions of data are concatenated together to form the augmented training data set, the unaugmented data takes up only one fifth of the data. Thus resulting the training model deviating one bit too much from the actual data. Thus resulting a decrease in the accuracies. One possible way to modify this is to attenuate the noise we inject to the dataset so that it contributes to the model's regularization and still not lose authenticity to the data.

### 3.2. Choice of hyperparameters and Neural Network

We performed grid search to find the optimal combination of hyperparameters and architectures. We tried different number of CNN layers and LSTM layers, also different orders of them. We also tired to tune the number of filters in CNN layers and hidden dimension in LSTM layers.

As for the CNN part, increasing the number of convolutional layers or reducing the number of max pooling layers will cause overfitting, while in contrast too much information of time series in the data may be lost. In addition, increasing the filter size can enlarge the reception field of CNN but this may also cause overfitting.

As for the LSTM part, bidirectional LSTM will get better performance than the normal one based on our observation.

We also tried a variety of activation functions for CNN and LSTM respectively, it came to the CNN part that Exponential Linear Unit(ELU) is a better choice and for LSTM part, the default activation function (tanh) will be a better choice.

Thus we finally chose the hybrid architecture with the optimal hyperparameters we found and we chose adam with default parameters as our optimizer.

### 3.3. Performance on Each subject vs All subjects

Training across all the subjects gives us the test accuracy as high as 68.172% (using CNN+LSTM). On the other hand, training on some of the subjects individually gave a test accuracy as high as 76%. (>68.172%) this partly proves our intuition that training should be optimised on each individual subjects because different human being have nuances in their inner brain function.

However there are also subjects whose training show far less accuracies, the lowest being 38%. This can be due to the lack of data on each of the subjects, being only around 235 for training and 50 for testing. This number of data can easily cause overfitting to the training set when we use a deep neural network. Thus we suggest that more data should be collected on each individual subjects so that each of models are best optimised for corresponding individual.

### 3.4. Performance on Different Time Windows

When evaluating the performance over different function of time. We used CNN+LSTM model and chose time slot from 300 to 1000 (the corresponding time periods are (0:time slot)). From the results, test accuracy was improved along with the increment of function of time. Starting from function of time = 700, which we thought was a required length for reasonable classification, the accuracy arrived a relatively high level and stable. When it came to function of time = 900, we got the highest accuracy = 65.91% (see in figure 5).

We thought that the overall tendency met our expectation: At the beginning stage, when sample length was too short (which ignore too much information as well as easily causing overfitting ) and especially the majority of data didn't belong to motor imagery part (which was an important determinant). The test result should be relatively bad. Along with the increment of function of time, our model can be trained under more useful information, which will improve the performance of prediction in the testing stage.

## References

[1] C. Brunner, R. Leeb, G. R. Muller-Putz, A. Schlogl, and G.Pfurtscheller. Bci competition 2008 graz data set a. 2008. 1

| Model | Raw | Filtered | Augmented |
|---|---|---|---|
| 1-d CNN | 50.113% | 55.079% | 64.560% |
| Naive LSTM | 27.540% | 29.120% | 27.314% |
| CNN+LSTM | 59.594% | 68.172% | 65.914% |
| CNN+GRU | 57.562% | 64.334% | 62.302% |

Table 1. Test accuracy using different stages of data augmentation.

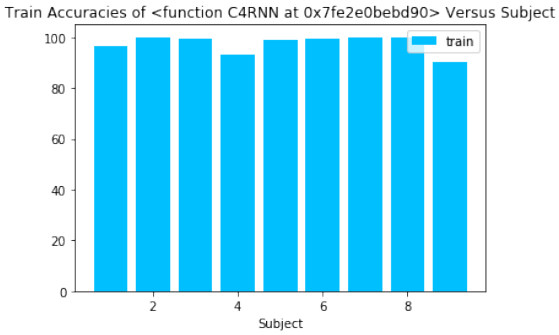


Figure 3. Training accuracy on each subject.

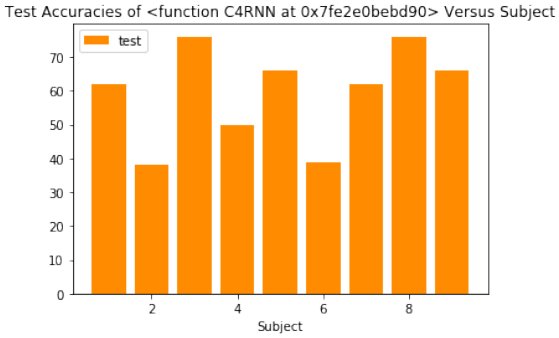


Figure 4. Test accuracy on each subject.

| Subject ID | Train Accuracy | Test Accuracy |
|---|---|---|
| 0 | 96.62% | 62.00% |
| 1 | 100.0% | 38.00% |
| 2 | 99.58% | 76.00% |
| 3 | 93.16% | 50.00% |
| 4 | 98.72% | 65.96% |
| 5 | 99.58% | 38.78% |
| 6 | 100.0% | 62.00% |
| 7 | 100.0% | 76.00% |
| 8 | 90.04% | 65.96% |

Table 2. Accuracy of each subject

| Model | Train Accuracy | Test Accuracy |
|---|---|---|
| 1-d CNN | 99.953% | 55.079% |
| Naive LSTM | 99.858% | 29.120% |
| CNN+LSTM | 98.440% | 68.172% |
| CNN+GRU | 99.574% | 64.334% |

Table 3. Accuracy across all subjects with different models (filtered data)

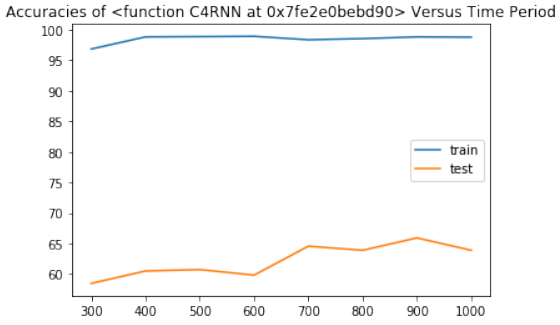


Figure 5. Train and Test accuracy on different time period.

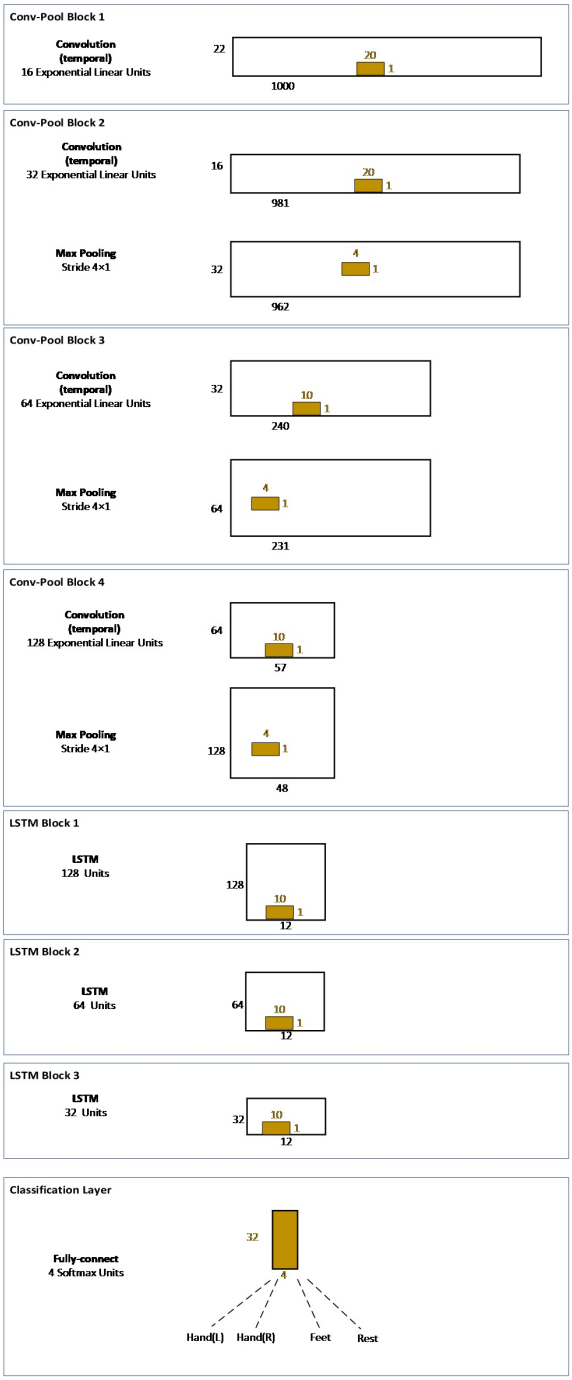


Figure 6. CNN+LSTM

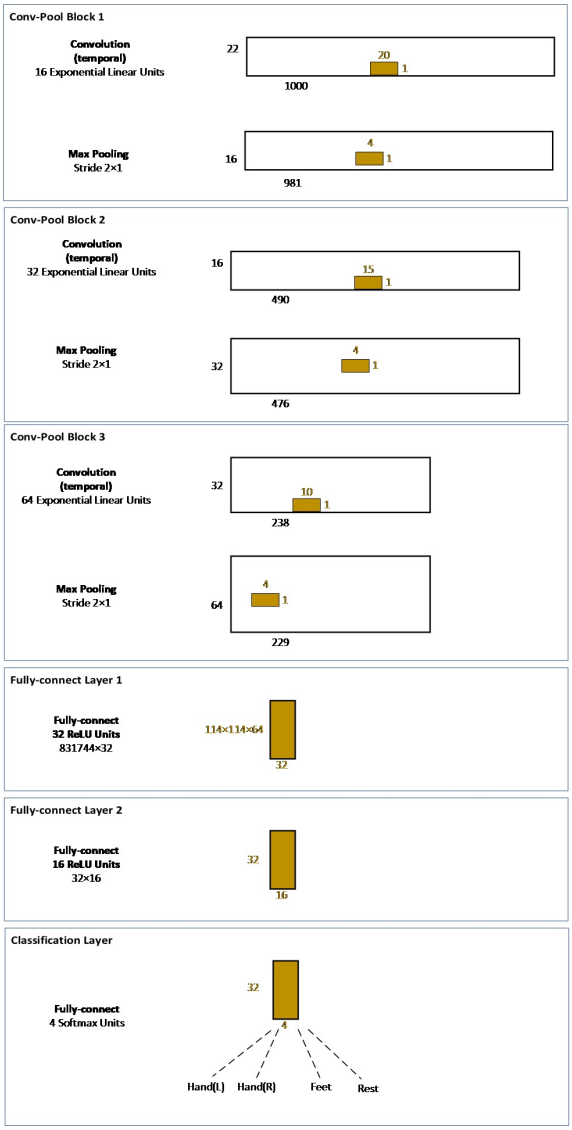


Figure 7. Naive CNN

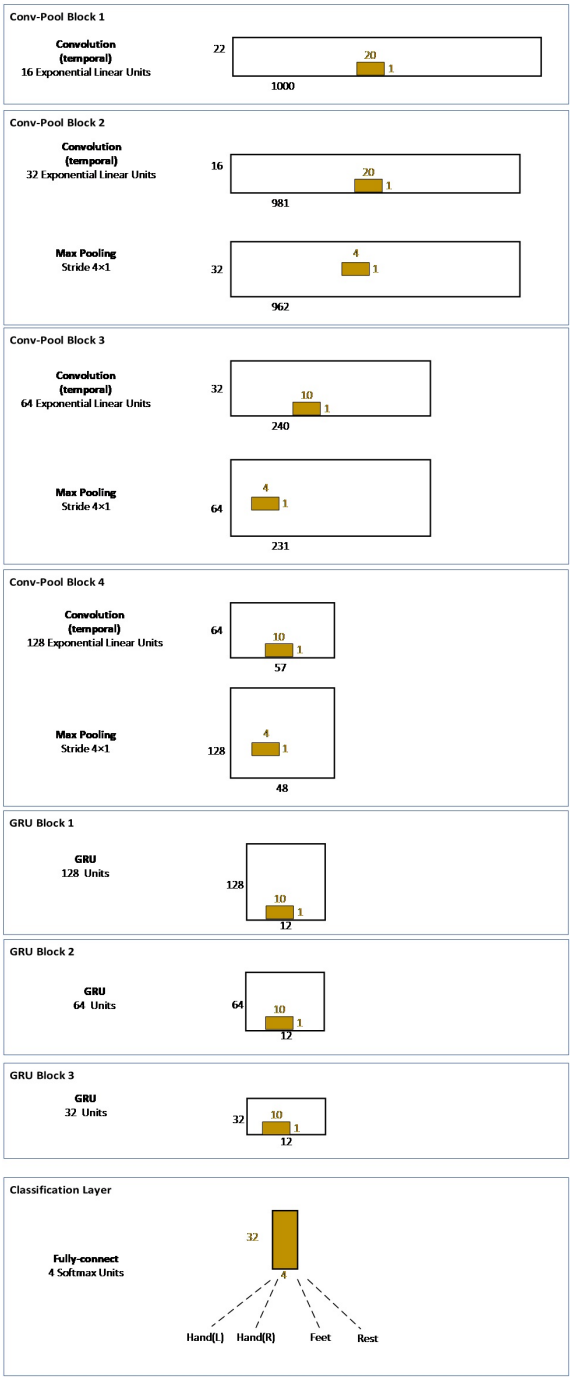


Figure 8. CNN+GRU