EEG 4-class Motor Classification

William Shih
UID: 805682201
williamshih@ucla.edu

Wonjin Lee
UID: 805739113
wonjinlee1202@ucla.edu

Abstract

This project focuses on the classification of electroencephalograph (EEG) data from the Brain-Computer Interaction (BCI) Competition [1]. The objective is to categorize the data into four distinct motor imaginary tasks: left hand movement, right hand movement, both feet movement, and tongue movement. This classification is achieved using a combination of Convolutional Recurrent Neural Network (CRNN) and Convolutional Neural Network (CNN). The report conducts a comparative analysis of the performance of CRNN and CNN, both for individual subjects and across all subjects. Additionally, the report explores and compares the temporal variations in EEG data durations. Optimizing our CNN-GRU model, we were able to observe a 71.6% testing accuracy.

1. Introduction

This report constructs four networks: a Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and two hybrid models using both (CRNN).

1.1. CNN

The first architecture we started designing was a basic CNN, a simple model that performs well at learning features from data. Compared to the conventional 2D filter used in many CNNs, we opted to use a 10x1 1D filter in order to better extract features based on the dimension of the time dumps, rather than also including that of the EEG electrodes as well.

Our initial CNN design consisted of a single Convolutional layer, with a dense softmax classification layer at the end. But upon experimentation, we found that our validation accuracy seemed to be plateauing at around the 50% test accuracy range. Thus, in order to increase the complexity of our model, we modified our initial design to include a second Convolutional layer.

After each of our two CNN layers, we incorporated Dropout with a probability 0.5 as well, as suggested by Schirrmeister's ConvNet model [2]. These Dropout layers randomly set some inputs of the layer to 0, making it analogous to training an ensemble of networks and thus providing regularizing effect. In addition, we introduced a kernel regularizer (L2 regularizer with strength 0.03) to further mitigate overfitting to the training data.

1.2. GRU

Given the temporal aspect of the EEG dataset, we naturally incorporated an RNN in our experimentation to take advantage of this time dimension. Specifically, we utilized a Gated Recurrent Unit (GRU), a simple yet effective kind of RNN that introduces fewer parameters and computations than other RNN architectures like LSTM. Given our limited resources and lack of access to GPUs while training, this was especially helpful in training various iterations of RNNs and fine-tuning our hyperparameters.

1.3. CNN-LSTM

We aimed to combine the strengths of both CNN and RNN, specifically Long Short-Term Memory (LSTM), to capture both spatial features and temporal dependencies. The architecture of our CNN-LSTM model, outlined in Table 6. We iterated upon existing code, fine-tuning hyperparameters to enhance performance [3].

1.4. CNN-GRU

Although we obtained adequate results with our CNN model and GRU model, we aimed to take advantage of both architectures in order to implement a post-CNN model that could attain a test accuracy of over 70%. We accomplished this by taking the original CNN model architecture, and adding a GRU layer after the second Convolutional layer.

Upon experimenting with different hyperparameter settings and architecture modifications, we found that incorporating an Early Stopping feature to this model helped increase our final test accuracy; in general, it allowed us to prevent overfitting and reduce training time.

2. Results

2.1. Classifying single subjects

The accuracy for classifying a single subject with only training data from the same subject is summarized in Table 1. The accuracy for classifying a single subject with training data from all subjects in summarized in Table 2. This was done for all 9 subjects in the dataset with 4 different models. The classification accuracies have a large range. Single subject training and classification had an accuracy range from 32.0% to 78.7% across all models and subjects. Single subject classification and training across all subjects had an accuracy range from and 46.0% to 87.2% across all models and subjects.

2.2. Classifying all subjects

The best performance for classifying all subjects for each of the 4 models is shown in Table 2. Our best performing model was the CNN+LSTM model that achieved an accuracy of 72.5% in classifying all subjects. We also trained the CNN-GRU model, another post-CNN architecture, to achieve an accuracy of 71.6%.

2.3. Classifying with different time bin durations

The accuracy for classifying all subjects over different time duration with all 4 different models are summarized in Figure 1. The graphs are superimposed so that trends that are common can be readily visualized.

3. Discussion

This section examines the comparisons of outcomes across various networks, particularly in single-subject and multi-subject classification. Additionally, it delves into our decisions regarding hyperparameters and architectural selections. Lastly, it discusses the preceding outcomes across different time intervals.

3.1. Data Preprocessing Techniques

We implemented various different data augmentation techniques in order to better optimize the dataset to be trained by our models. The EEG data is first trimmed to remove the last 200 time bins, leaving us to train on the first 800 out of the original 1000 time bins. This was implemented upon observing plots of the data in relation to time, where we observed that the later portions of data seemed to be noisier and fail to resemble a clear pattern that could be trained.

After performing this initial data trimming, we employ several other data augmentation techniques, with the intent of artificially increasing the size of the training dataset to help improve the generalization ability of our models. Specifically, we performed maxpooling, averaging, and subsampling on intervals of 4 time dumps. This allowed us to increase and diversify our training dataset, which in turn decreases overfitting of our models on training data.

3.2. Compare between one vs all subjects

When comparing the approach of training a given model on data from all subjects versus training on data from just one subject to classify that one subject, several key insights emerge. Firstly, the model trained on data from all subjects tends to be more generalizable and have relatively better classification accuracy. This increased generalizability holds true even when employing data augmentation techniques. These techniques include smoothing, maxing, and trimming the data to generate additional replicas. However, despite these efforts, there may still be insufficient variation within the dataset to effectively train a robust model. This limitation arises from the inherent diversity among subjects and the need for ample variability in the training data to build a reliable model.

However, it is important to acknowledge that the size of the test data is relatively small, around 50 samples per subject. Consequently, there is more of an element of randomness in the testing process. This randomness can influence the performance metrics of the model.

It is also worth noting that classification of subjects who consistently perform well in single subject training also tend to achieve high accuracy relative to others in all subject training scenarios, and a less apparent, but similar pattern holds for subjects with low performance. This observation suggests a correlation between a subject's ability to consistently execute actions and the effectiveness of the model trained on their data. The consistency in performance across training scenarios likely reflects the subject's proficiency in performing the specific action, which is in this case performing of one of the four motor actions. To validate this hypothesis, we plotted some of the data to visualize; however, it still remained difficult to verify consistency of the signals across subjects.

3.3. Choice of architectures

We see that the hybrid models that combine both CNN and RNN outperform the purely CNN and RNN models by a significant margin, which was consistent with our hypothesis (Table 3).

This is likely attributed to several factors. Firstly, EEG signals inherently contain both spatial and temporal information, making them well-suited for the complementary strengths of CNNs and RNNs. By integrating both architectures, the hybrid model can effectively leverage both spatial and temporal information simultaneously. Additionally, the hybrid model's ability to handle variable-length sequences inherent in EEG data, thanks to the inherent properties of RNNs, ensures robust performance across different dura-

tions of recorded activities. Moreover, the combination of CNNs and RNNs offers inherent regularization benefits, reducing the risk of overfitting by enforcing local connectivity and capturing long-term dependencies in the data. This approach not only enhances feature representation but also improves generalization capabilities, resulting in better classification performance for discriminating different motor actions based on neural signals.

3.4. Compare between different time bin durations

To gain insight into the impact of different time bin durations on classification accuracy across all subjects, we experimented with durations ranging from 100 to 1000 time bins, with a step size of 100. Across all models, we observed a noticeable initial increase in classification accuracy, occurring roughly between 100 and 500 time bins, reaching peak performance towards the latter end of this range. Following this peak, we observed either a plateauing or a decrease in accuracy. (Figure 1).

These findings suggest that a significant portion of the data crucial for classification is typically concentrated within the initial few hundred time bins, with relatively less essential information present in the latter portion. Visualizing several data samples across subjects reinforced this observation, revealing that most neural signal information is indeed contained within the first few hundred time bins. These segments exhibit more pronounced fluctuations likely correlated with the specific actions the subjects are attempting. In contrast, the latter half of the data tends to display less activity, characterized by flatter signals.

3.5. Choice of hyperparameters

We conducted a grid search to determine the optimal hyperparameters for our CNN+LSTM model, specifically focusing on the learning rate and number of epochs. Using reasonable step sizes, we systematically evaluated various combinations of these parameters to identify the configuration that yielded the highest performance. We discovered that the optimal settings for our model were 150 epochs and a learning rate of 10^{-3} (Figure 2).

In addition to fine-tuning the hyperparameters, we also explored different optimizers available in the Keras library. After thorough evaluation, we ultimately selected the Adam optimizer for our CNN+LSTM model.

The hyperparameters involved with weight regularization were also thoroughly fine-tuned, particularly during our design of the CNN architecture. After various iterations of training models with varying regularization norms and strengths, we arrived at the choice of L2 regularization with a strength of 0.03. This strength is larger than what we had originally started training with, but it successfully assists the model in preventing overfitting, and we thus extended this hyperparameter to choice to our other models as well.

With these hyperparameters in mind, we applied similar reasonable values to our other models.

References

[1] A. BCI Competition IV. BCI Competition IV, www.bbci.de/competition/iv/.

[2] C. Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W., Ball, T. (2017). Deep learning with convolutional neural networks for EEG decoding and visualization. Human Brain Mapping, 38(11), 5391–5420. DOI: 10.1002/hbm.23730

[3] M.Tonmoy, Discussion 7 DL W24.ipynb

Appendix

A. Algorithm Performance Metrics

SubjectID	CNN	GRU	CNN+LSTM	CNN+GRU
0	44.0%	44.0%	56.0%	44.0%
1	52.0%	44.0%	48.0%	32.0%
2	64.0%	62.0%	70.0%	34.0%
3	46.0%	42.0%	47.0%	38.0%
4	78.7%	44.7%	74.5%	61.7%
5	51.0%	40.8%	44.9%	46.9%
6	58.0%	60.0%	72.0%	68.0%
7	64.0%	50.0%	48.0%	50.0%
8	70.2%	61.7%	66.2%	57.4%

Table 1. Single-Subject Training and Classification

SubjectID	CNN	GRU	CNN+LSTM	CNN+GRU
0	66.0%	56.0%	62.0%	60.0%
1	56.0%	54.0%	46.0%	58.0%
2	74.0%	66.0%	80.0%	60.0%
3	68.0%	62.0%	60.0%	60.0%
4	78.7%	68.0%	80.9%	70.2%
5	65.3%	49.0%	63.3%	61.2%
6	64.0%	66.0%	58.0%	68.0%
7	66.0%	62.0%	74.0%	72.0%
8	70.2%	59.6%	87.2%	76.6%

Table 2. Single-Subject Classification, Trained on All-Data

Model	Accuracy
CNN	65.5%
GRU	63.7%
CNN+LSTM	72.5%
CNN+GRU	71.6%

Table 3. Optimized Testing Accuracy Across All Subjects

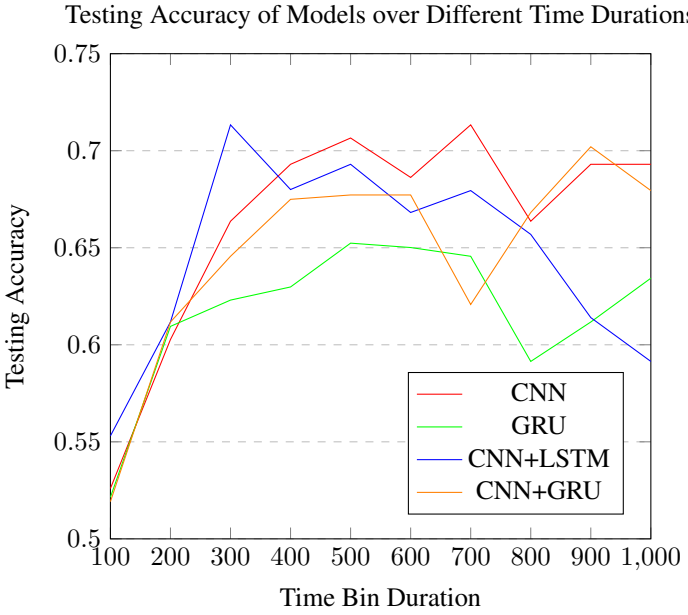

Figure 1. Accuracy Over Different Time Bin Durations

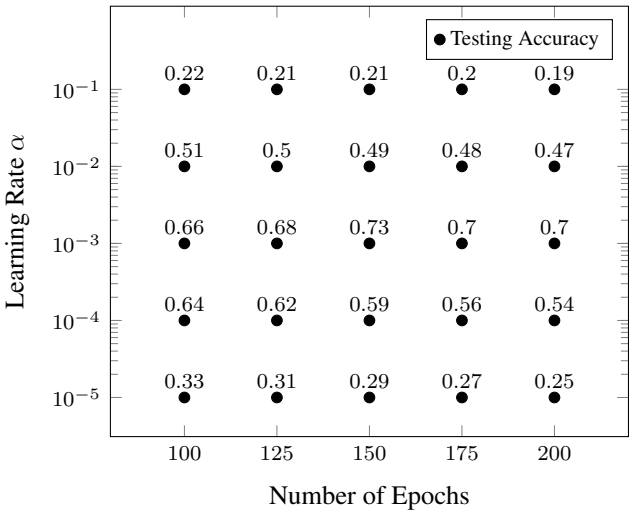

Figure 2. CNN+LSTM Learning Rate and Epochs Grid Search

B. Model Architectures

Layer (type)	Output Shape	Parameters
1D Convolution	200, 22	4862
1D Max Pooling	100, 22	0
Dropout	100, 22	0
1D Convolution	100, 22	4862
2D Max Pooling	50, 22	0
Dropout	50, 22	0
Flatten	1100	0
Dense	64	70464
Dropout	64	0
Dense	4	260

Table 4. CNN Architecture.

Layer (type)	Output Shape	Parameters
GRU	200, 16	1920
Flatten	3200	0
Dropout	3200	0
Dense	4	12804

Table 5. GRU Architecture.

Layer (type)	Output Shape	Parameters
2D Convolution	400, 1, 25	13775
2D Max Pooling	134, 1, 25	0
Batch Normalization	134, 1, 25	100
Dropout	134, 1, 25	0
2D Convolution	134, 1, 50	31300
2D Max Pooling	45, 1, 50	0
Batch Normalization	5, 1, 50	200
Dropout	45, 1, 50	0
2D Convolution	5, 1, 100	125100
2D Max Pooling	15, 1, 100	0
Batch Normalization	15, 1, 100	400
Dropout	15, 1, 100	0
2D Convolution	15, 1, 200	500200
2D Max Pooling	5, 1, 200	0
Batch Normalization	5, 1, 200	800
Dropout	5, 1, 200	0
2D Convolution	5, 1, 400	2000400
2D Max Pooling	2, 1, 400	0
Batch Normalization	2, 1, 400	1600
Dropout	2, 1, 400	0
Flatten	800	0
Dense	100	80100
Reshape	100, 1	0
LSTM	50	10400
Dense	4	204

Table 6. CNN+LSTM Architecture.

Layer (type)	Output Shape	Parameters
1D Convolution	200, 22	4862
1D Max Pooling	100, 22	0
Dropout	100, 22	0
1D Convolution	100, 22	4862
2D Max Pooling	50, 22	0
Dropout	50, 22	0
1D Convolution	50, 22	4862
Dropout	50, 22	0
GRU	50, 44	8976
Dropout	50, 44	0
Flatten	2200	0
Dense	64	140864
Dropout	64	0
Dense	4	260

Table 7. CNN+GRU Architecture.