# ECE C247 Project Submussion

Zhuohao Li
zhuohaol@ucla.edu

Kangning Li
kl32@ucla.edu

Zilai WANG
zilaiwang2001@ucla.edu

## Abstract

*Brain-computer interactions (BCIs) have been used for medical applications such as neural control of prosthetic artificial limbs. Current BCI research are with noninvasive approaches based on Electroencephalogram (EEG). EGG data analysis involves interpreting the electrical activity of the brain recorded. This process includes examining the patterns and rhythms of brain waves to diagnose conditions, understand brain functions, and even decode cognitive states. Deep learning has revolutionized image processing and computer vision through multiple end-to-end learning strategies, including BCI domain. In this project, we developed multiple methods of emerging deep learning techniques ranging from CNNs to RNNs with Attention mechanisms to boost our understanding and interpretation of dancing with EEG dataset. We carefully tuned the hyperparameters of the DL models and designed their architecture as well as applied multiple well-developed pre-trained models. We evaluated a range of architectures including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRU), and Attention mechanisms. Our study sheds light on their efficacy in decoding complex EEG datasets. Our results indicate the precision of our methods could be raised up to 70% and we also discussed the reasons why several issues happened during our design. Our code is avaliable at https://github.com/Zhuohao-Li/CS-C247/tree/main/Project.*

## 1. Introduction

EEG (electroencephalography) datasets are a form of time-series data and are often suitable for use with models capable of processing time-series data, such as recurrent neural networks (RNNs) and their variants (e.g., the Long Short-Term Memory Network, LSTM, and the Gated Recurrent Unit, GRU [1]).RNNs are especially designed to process sequential data because they can transfer information between time steps, which makes them theoretically suitable to process EEG data, or any form of time series data.

RNNs are able to take advantage of the temporal dependencies in EEG data to recognize patterns and features in the data stream that change over time. This is useful for EEG data analysis because EEG signals often contain information about EEG activity over time, which can be used for a variety of applications, such as mood recognition, sleep stage classification, and neurological disease diagnosis.

In this case, we decide to test the GRU model on the dataset. And we also test different hyperparameter settings and their influence on the performance.

Compared with LSTM, GRU has fewer parameters and has two gates: Reset gate and Update gate. It trains faster and requires less computational resources.

## 2. Results

### 2.1. CNN

We first explore the performance of traditional CNN-based methods. We tried different architecture of CNN from base-CNN, complex CNN, and residual connected CNN (ResNet). There are many other kinds of architecture applied in EGG datasets before. [8] discussed potential architectures for processing raw EGG datasets and it provided several implementation of deep, shallow, and residual convolutional neural networks. [7] provides a solution called EGGNet, a compact CNN for classification and interpretation of EEG-based BCIs. It introduced the use of depthwise and separable convolutions, which is previously used in computer vision [2], to construct an EEG-specific network that encapsulates several well-known EEG feature extraction concepts.

Here, we briefly introduce the architectures we explore regarding the CNN.

1. *Based CNN*: We first implemented a base and simple CNN with only 5 blocks (4 for hidden layers and 1 for output classifier). We trained the model for 50 epochs and tuned the parameters with cross validation.

2. *Complex CNN*: We then deepen the base CNN to further add more convolution blocks to it to 50 blocks. We still perform layer by layer sequential connection behavior. We trained the model for 50 epochs.

Table 1. Results in three architectures of CNNs

| Architecture | train loss | vali loss | accuracy |
|---|---|---|---|
| Based | 0.70 - 2.19 | 0.84 - 1.86 | 63.2% |
| Complex | 1.80 - 3.22 | 1.49 - 9. 26 | 60.1% |
| Residual | 1.79 - 4.11 | 1.23 - 5.66 | 68.2% |

3. *Residual CNN*: [4] proposed a residual connection applied to deep neural networks to prevent the risks of overfitting. We implelemented a residual connection layer of ResNet-50 and perform the same evaluation processes.

We talked about the architecture in the appendix.

Table 1 indicates the results for our CNN-based models. For more disscuss, please refer to 3.

## 2.2. GRU

In this model, we exploit the ResNet(Residual Network) [3], GRU, and also a FullyConnect layer to compose the entire model. We exploit the pre-trained Resnet, which gives us a well-trained initial weights. Because the ResNet is pretrained on ImageNet, which is a very large dataset, it can extract many useful features of the image. We first give the input data to ResNet to output 512 features. Then pass the features to GRU to get 128 features. Finally, after being processed by the FC, we get the output and test the performance.

We change the settings: FC layer dimensions, learning rate, and GRU dimensions. On average, we can get high accuracy on the train set, and nearly 0.5 accuracy for the validation dataset. Here are some results about the settings.

For Fig.3, we change the GRU output dimension to 128. The training accuracy achieves 0.7, validation accuracy achieves 0.5. For Fig.4, we change the decay rate from 0.001 to 0.003. Compared with Fig.3, the train accuracy increases a lot, and the average validation accuracy improves a little. For Fig.5, we change the batch size to 64, we can see the performance become worse. The reason might be because of the overfitting. For Fig.6, we change the GRU layer from 3 to 4. the train accuracy decreases to 0.5. The reason be because of the overfitting.

So overall, we choose the best settings: decay rate=0.003,GRU output dimension=128,Gru layer=3, batch size=32. It can achieve 0.9 train accuracy and 0.55 validation accuracy.

## 2.3. LSTM

Then, we exploit the LSTM(Long short-term memory) [5].We tried differencde architecture of LSTM from the sequence-to-sequence (Seq2Seq) LSTM with attention and shallow-LSTM. Here, we briefly introduce the architectures we explore regarding the LSTM.

Here, we briefly introduce the architectures we explore regarding the LSTM.

1. *shallow LSTM*: The input will firstly be input to one LSTM layer which outputs 64 hidden units. Then it is further passed to FC (fully connected) layer, which is composed of serveral linear layers, Batchnorm, and ReLU activation function. We train shallow LSTM model with 100 epochs.

2. *Seq2Seq LSTM with attention*: We then explored the Seq2Seq LSTM with attention, a much more complex model than shallow LSTM. It is composed of one encoder and one decoder with attention. The encoder is used to process the input sequence and generate a context for each time step.The decoder is used to decode the encoder's context into an output sequence, using attention to focus on different parts of the input sequence at each step. We trained the Seq2Seq LSTM with attention with 100 epochs

We talked about the architecture in the appendix.

Table 2 indicates the results for our LSTM-based models. For more disscuss, please refer to 3.

Table 2. Results in three architectures of LSTMs

| Architecture | train loss | vali loss | accuracy |
|---|---|---|---|
| Shallow | 0.071 | 4.45 | 26.5% |
| Seq2Seq with attention | 0.373 | 3.31 | 25.1% |

## 2.4. CNN+LSTM

Then, we exploit the combination of CNN and LSTM. RaSCNN (Recurrent Attention-based Spatial Convolutional Neural Network) [6] is used to combine convolutional neural network layers with an LSTM layer that includes an attention mechanism. This model is designed for tasks that benefit from both spatial feature extraction (via CNN) and sequential data processing with attention to sepecific time steps (via LSTM).

This model structure contains spatial convolutional layers, temporal convolutional layers, attention LSTM layer, and dense layers. Each spatial convolutional layers captures spatial features, with batch normalization and spatial dropout applied afterward. The temporal convolutional layers extract temporal dynamics using filters, followed by batch normalization, average pooling, and dropout. The attention LSTM layer incorporates attention mechanism to focus on relevant parts of the input sequence. This could be visualized as a layer that takes both the output of the

Table 3. Results of CNN+LSTM

| Architecture | train loss | vali loss | accuracy |
|---|---|---|---|
| RaSCNN | 0.7443 | 1.4671 | 72.3% |

temporal convolutions and its own previous states to compute attention weights and produce a weighted representation. Each dense layer applies transformations with 'selu' activation, followed by batch normalization and dropout for regularization. We trained this model with 250 epochs.

Table 3 indicates the results for our CNN+LSTM model. For more disscuss, please refer to 3.

## 3. Discussion

### 3.1. CNN

*CNN*: We found that simply adding more layers to the CNNs does not always cause better performance. This is maybe because overfitting. The original EGG dataset is not really large in the scale, so we would like to do data augmentation to against overfitting. Residual block is a good way to prevent that, as we can see improvement when using residual connection compared with simple complex CNNs.

### 3.2. GRU

We exploit the ResNet to exploit the pretrained weights to improve the performance. It is because the Resnet is trained on the Imagenet. And we also change the hyperparameter to test the performance. We can see larger GRU layers would make worse performance because the depth of the network is too large. Increasing the batch size would cause the same effect as increasing the depth of layers.

Table 4. Test accuracy vs Number of Subjects

| Number of Subjects | Test accuracy |
|---|---|
| 6 | 0.35 |
| 3 | 0.38 |
| 1 | 0.52 |

So we can see that when we increase the number of subjects, the classification accuracy for subject 1 would decrease. It might be because when the larger dataset causes the network to learn a wider range of features, which may decrease the classification accuracy for a specific target, for example subject 1.

Based on this table, we can see that when we increase the time series, the test accuracy increase. It might be because increasing the dataset size, will make the network learn more features and make more precise predictions.

Table 5. Test accuracy vs Time series

| Time series | Test accuracy |
|---|---|
| 1000 | 0.51 |
| 800 | 0.44 |
| 500 | 0.38 |

### 3.3. LSTM

We notice that Seq2Seq LSTM with attention model has serious overfitting problem. While the training loss continuously decreases, the validation loss is increasing. The accuracy is around 25%, showing that the trained model does not learn how to classify the test dataset. The reason why Seq2Seq LSTM with attention model has poor generality is its high complexity with training on limited amount of dataset. Then we simplifed the model, using shallow LSTM. Similarly, we found that shallow LSTM also has serious overfitting problem. This is because the dimensionality of the input data of LSTM models are much larger than the CNN models, LSTM models tend to suffer from overfitting problem with training on the limited amount of dataset.

### 3.4. CNN+LSTM

Combined with CNN and LSTM, RaSCNN model performs better than either CNN or LSTM model. There are several reasons about it. Firstly, RaSCNN utilizes CNN layers to extract high-level spatial features from the input data before processing it with recurrent layers. Also, LSTM layers are employed to capture the temporal dependencies and sequences within the data after spatial feature extraction. This combination allows RaSCNN to not only recognize spatial patterns but also how these patterns evolve over time. The inclusion of an attention mechanism enables the model to focus on the most relevant parts of the input sequence for making prediction. This is particularly useful in scenarios where only specific segments of the sequence are informative for the task at hand. By weighting the input data's parts differently, the model can potentially improve its accuracy and interpretability. The use of CNNs for feature extraction reduces the dimensionality of the input data fed into the LSTM, which can help mitigate overfitting by simplifying the problem space. Also, the attention mechanism focuses the model;s capacity on the most important parts of thre data, which can also contribute to reducing overfitting.

## 4. Appendix: Table

Table 6. Summary of the results of models across all subjects

| Architecture | Train Loss | Validation Loss | Accuracy |
|---|---|---|---|
| Based CNN | 0.70 - 2.19 | 0.84 - 1.86 | 63.2% |
| Complex CNN | 1.80 - 3.22 | 1.49 - 9. 26 | 60.1% |
| Residual CNN | 1.79 - 4.11 | 1.23 - 5.66 | 68.2% |
| Shallow LSTM | 0.071 | 4.45 | 26.5% |
| Seq2Seq LSTM with attention | 0.373 | 3.31 | 25.1% |
| ResNet+GRU | 0.208 | 1.5889 | 52% |
| CNN+LSTM (RaSCNN) | 0.7443 | 1.4671 | **72.3%** |

Table 7. Test accuracy vs Number of Subjects of GRU model

| Number of Subjects | Test accuracy |
|---|---|
| 6 | 0.35 |
| 3 | 0.38 |
| 1 | 0.52 |

Table 8. Test accuracy vs Time series of GRU model

| Time series | Test accuracy |
|---|---|
| 1000 | 0.51 |
| 800 | 0.44 |
| 500 | 0.38 |

## 5. Appendix: Methods

Table 9. ResNet+GRU structures

| Layer | Output features |
|---|---|
| ResNet | 512 |
| GRU | 128 |
| Linear | 64 |
| ReLu | 64 |
| Linear | 32 |
| Relu | 32 |
| Linear | 16 |
| ReLu | 16 |
| Linear | 4 |

Table 10. RNN structures

| LSTM model | hidden units | number of layers | dropout rate | optimizer | lr | weight decay |
|---|---|---|---|---|---|---|
| Shallow LSTM | 32 | 1 | 0.4 | Adam | 0.0005 | 0.0005 |
| Seq2Seq LSTM with attention | 32 | 2 | 0.5 | Adam | 0.0005 | 0.0005 |
| Resnet+ GRU | 32 | 2 | 0.5 | Adam | 0.0005 | 0.0005 |

Table 11. Rascnn structure

| Layer | Output Shape |
|---|---|
| input (InputLayer) | [(None, 1000, 22)] |
| reshape (Reshape) | (None, 1000, 22, 1) |
| spatial conv 0 (Conv2D) | (None, 1000, 12, 100) |
| spatial conv 1 (Conv2D) | (None, 1000, 1, 100) |
| BachNorm | (None, 1000, 1, 100) |
| spatial dropout2D | (None, 1000, 1, 100) |
| temporal conv 0 | (None, 977, 1, 40) |
| temporal conv 1 | (None, 954, 1, 40) |
| temporal conv 2 | (None, 931, 1, 40) |
| temporal conv 3 | (None, 908, 1, 40) |
| BachNorm | (None, 908, 1, 40) |
| Average Pooling2d | (None, 45, 1, 40) |
| spatial dropout2D | (None, 45, 1, 40) |
| reshape (Reshape) | (None, 45, 40) |
| Bidirectional | (None, 45, 152) |
| Batch Norm | (None, 45, 152) |
| Flatten | (None, 6840) |
| Dense | (None, 4) |

## 6. Appendix:Figure

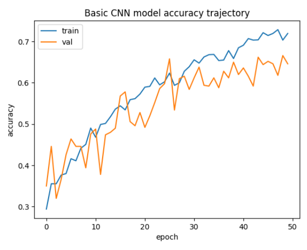

Figure 1. CNN Accuracy

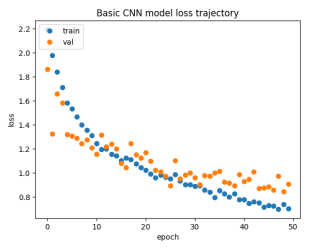

Figure 2. CNN Loss

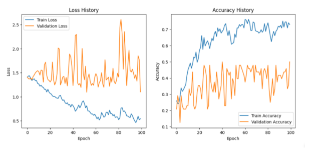

Figure 3. GRU output dimension: 128

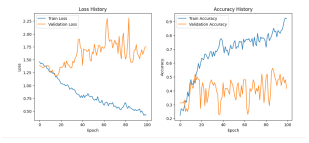

Figure 4. Decay rate: 0.003. Same dimension

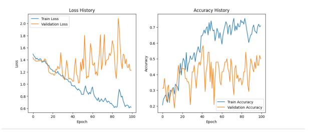

Figure 5. Batch size: 64

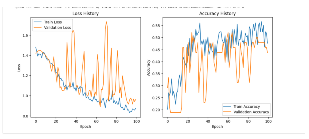

Figure 6. GRU layer: 4

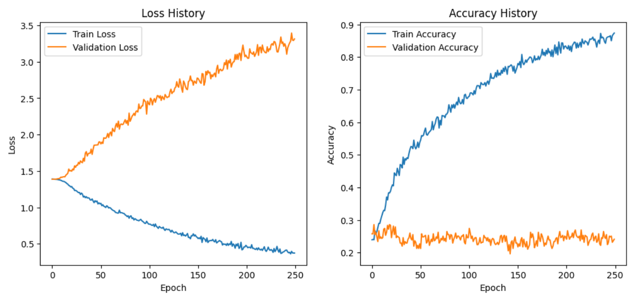

Figure 7. Seq2Seq LSTM with attention

## References

[1] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Association for Computational Linguistics*, 2014. 1

[2] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017. 1

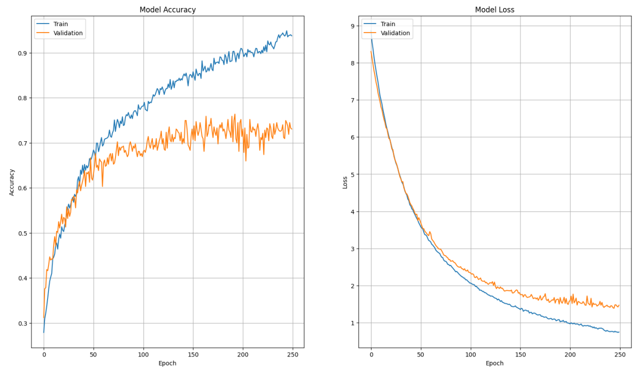

Figure 8. Rascnn

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385, Dec. 2015. 2

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[6] Demetres Kostas, Elizabeth W Pang, and Frank Rudzicz. Machine learning for meg during speech tasks. *Scientific reports*, 9(1):1609, 2019. 2

[7] Vernon J Lawhern, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance. Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces. *Journal of neural engineering*, 15(5):056013, 2018. 1

[8] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human brain mapping*, 38(11):5391–5420, 2017. 1