Convolutional Neural Network (CNN) for EEG 4-class Motor Classification

Jason Tay
UID: 805599377
jtay20@g.ucla.edu

Henry Wang
UID: 405691801
henrywang3@g.ucla.edu

Lily Zhou
UID: 505750593
lzhou0714@g.ucla.edu

Abstract

This paper reports the findings on various machine learning models performing multiclass classification for electroencephalography (EEG) data collected from a non-invasive electrode with four classes, corresponding to four different imaginary motor tasks of these body parts: left hand, right hand, feet, and tongue. In this project, a Convolutional Neural Network (CNN) was implemented to get a baseline performance followed by testing various combinations implementing self-attention, Long Short-Term Memory (LSTM), and Multi-head Attention (MHA) with the base CNN model. This resulted in 5 models: CNN, CNN + LSTM, CNN + self-attention, CNN + MHA, CNN + LSTM + SA. All models were trained with the entire dataset containing all subjects. The best testing accuracy achieved in this project was 70.4% with CNN+MHA.

1. Introduction

Convolutional Neural Networks (CNNs) have emerged as a popular method for decoding raw Electroencephalography (EEG) signals in motor imagery classification tasks. EEG signals are complex and noisy, making it challenging to extract meaningful Although other methods, like support vector machine (SVM) and multi-layer perceptron (MLP) can also be used, CNNs are better suited for a variety of reasons. EEG signals contain a high degree of noise, so the tolerance of the CNN model to learn from different parts of input data makes it more robust. In addition, CNNs can learn features automatically through filters, whereas with SVM and MLP, some manual feature engineering may be necessary. CNNs can learn local features from the input data through their convolutional layers, capture spatial information using 2D convolutions, and capture temporal information using 1D convolutions or recurrent layers. As a result, they can handle high-dimensional data, learn meaningful features automatically, and take advantage of both spatial and temporal information to produce accurate and efficient classifications in motor imagery classification tasks.

Regardless of the method, pre-processing is necessary in order to overcome various issues like low signal-to-noise ratios. Popular techniques include data trimming, max-pooling, averaging with noise, cropping, and spatial transforms, of which we implemented the first 3 after some experimentation with spatial transforms did not show improvement.

In this project, we implemented 5 CNN-based models using a hybrid of LSTM, self-attention, and MHA with the base CNN model. In particular, we tested the base CNN model, CNN with LSTM, CNN with self-attention, CNN with MHA, and lastly CNN with LSTM and MHA.

1.1 CNN Model

For our study, we implemented a CNN model for motor imagery classification tasks. The model consisted of four convolutional layers, with each convolutional block containing a convolutional layer followed by max pooling, batch normalization, and dropout. This configuration is standard for CNN models, and it provided a strong foundation for our future models when we added more complex features. During our experimentation, we discovered that this model had an exceptional ability to learn complex features from the input data, resulting in high accuracy and robustness. However, we also noted that the model required extensive tuning of hyperparameters to achieve optimal performance.

1.2 Transformer and Self-Attention

A self-attention Transformer maps the EEG signal embeddings using an embedding layer, which is followed by self-attention and feedforward layers in a series. The self-attention layer helps weigh the importance of signal parts and the feedforward layer applies non-linear transformations for learning complex relationships between the input and output. The output of the CNN model is then fed to the self-attention Transformer. The self-attention mechanism captures the complex relationships between different time steps and helps to learn relevant parts of the input signal, which is particularly useful for EEG classification tasks where the signal can be highly complex and non-stationary, and where long-range dependencies and correlations between different time steps are important.

1.3 Multi-head Attention (MHA)

Multi-head attention extends self-attention to enable a model to attend to multiple, separate parts of the input sequence at the same time using several attention mechanisms, or heads. It transforms the input sequence into multiple queries, keys, and values, each of which is independently processed through the attention mechanism to calculate its own attention weights. The final set of weights is produced by combining the attention weights of all the heads, which are used to compute the weighted sum of the values to model complex relationships between different parts of the input sequence. This property is particularly useful for

EEG classification tasks where long-range dependencies and correlations between different time steps are important.

This was built upon the baseline CNN model with an added MultiHeadAttention layer (num_heads = 8) followed by a LayerNormalizationlayer (similar to batch normalization but per example instead of per batch) and finally a GlobalAveragePooling2D layer. We experimented with positional encoding but found that it did not result in better performance. One possible explanation is that the relative position of the input tokens may already be implicitly captured by the model's architecture and the data preprocessing work, so the additional complexity may even be detrimental to performance.

1.4 CNN with LSTM

When a CNN and LSTM are used together, the CNN is used to extract spatial features from the EEG signals, while the LSTM is used to capture the sequential dependencies in the data. This model in theory can effectively learn both spatial and temporal patterns in the data. The output of the last convolutional block is flattened and fed into a fully connected layer with 120 units, which is followed by a reshape operation and an LSTM layer with 10 units. Finally, a dense layer with a softmax activation function is added as the output layer with 4 classes, which are the possible EEG states.

2. Results

We trained on EEG data from the dataset provided in "BCI Competition IV dataset 2a" (http://www.bbci.de/competition/iv/desc_2a.pdf). The dataset consists of EEG data from nine subjects, each performing four different motor imagery tasks (left hand, right hand, foot, and tongue) while wearing an EEG cap. Each of the 9 subjects has 22 channels associated with them, with which data was collected over 1000 timestamps with band-pass filtering between 0.5 Hz and 100 Hz and notch filtering at 50 Hz for noise suppression.

We evaluate our model across all subjects using the training data. As shown in Table 1, we found that the CNN with MHA, CNN with SA and LSTM, and the base CNN had the best testing accuracies of 70.71%, 70.43%, and 70.09%, respectively. The remaining models underperformed as compared to the base CNN model. The CNN with LSTM and the CNN with self-attention had testing accuracies of 67.83% and 68.06%. We also observed a high correlation between the training and validation/testing accuracy, suggesting that our model had a high degree of complexity and avoided overfitting while also maintaining a fast training speed (~2-3 seconds per epoch, batch size of 32). Each model had varying converging speeds, which was often due to the number of layers and parameters. The base CNN model only took 50 epochs, but the other models took significantly longer to train. CNN+LSTM, CNN+MHA, CNN+SA, and CNN+LSTM+SA took 183, 218, 300, and 190 epochs to train, respectively.

We observed that over time, the difference between the training and validation accuracy increases as the model begins to overfit to the training data and the decrease in validation loss becomes less with each successive epoch. For example, this phenomenon can be seen most clearly in Figure 3 after epoch 60; The train and validation accuracy curves begin to diverge with a greater difference between them after each subsequent epoch.

3. Discussion

Although CNN was the simplest model, it was still able to effectively learn spatial patterns in the EEG data and achieve a testing accuracy of 70.09%. CNN+LSTM was able to capture both spatial and temporal patterns and should be more versatile than just CNN alone, it had a lackluster performance. It is possible that the input features were already being captured well in the CNN since EEG data can be interpreted as image data and would thus be suitable for a CNN. With CNN+MHA, the multi-head attention mechanism allowed it to select the most relevant parts of the input data. This model performed better than both CNN and CNN+LSTM, however, it was more complex and computationally expensive as well.

One possibility for these results was that the input data was not balanced well. Although the training set was generally uniformly distributed, there was bias in the test set towards the ‘Right Hand’ class with less distribution for the remaining three classes. It is possible that this imbalance resulted in not only the gap between training and validation accuracy in the 70% range but also a bias away from the CNN+LSTM model. Further analysis could be done via data augmentation and increasing/changing the test set to determine whether this is the case.

3.1 Improving Model Performance

After data preprocessing and input standardization were completed, we worked on improving the performance of each of our models. For example, the CNN + self-attention model was able to reach ~60% test accuracy without many hyperparameter adjustments. Increasing the number of heads and adjusting other factors such as dropout allowed us to reach up to ~68% test accuracy. We believe that dropout was able to significantly improve the models’ performance by

preventing overfitting and encouraging the LSTM models to learn more generalized features. The best validation accuracy we reached was 74%, and was not done by performing PCA or data augmentation, but instead by modifying the activation functions in the convolutional layers, using a combination of GELU and SWIFT instead of ReLU. We believe this may have been due to the following: GELU and SWISH produce smoother gradients, reducing the risk of exploding gradients which we found had occurred with ReLU under certain hyperparameters; they improve the expressiveness of the model, and in particular, GELU and SWISH were found to perform well specifically on image classification, which has many similarities to the EEG data.

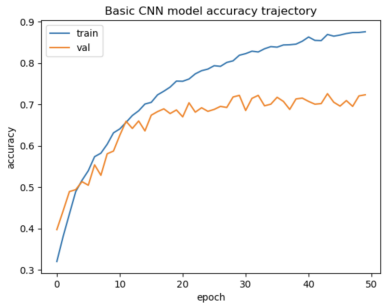


Figure 1. Test accuracy for base CNN model

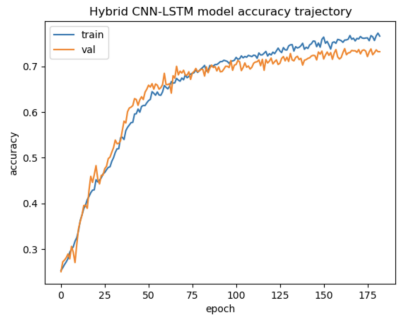


Figure 2. Test accuracy for hybrid CNN-LSTM model

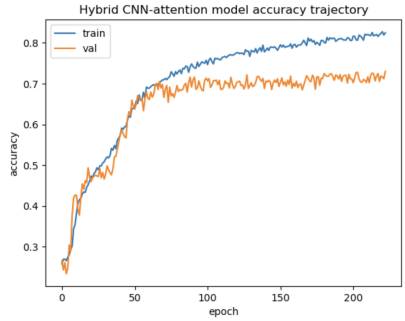


Figure 3. Test accuracy for hybrid CNN + self-attention model

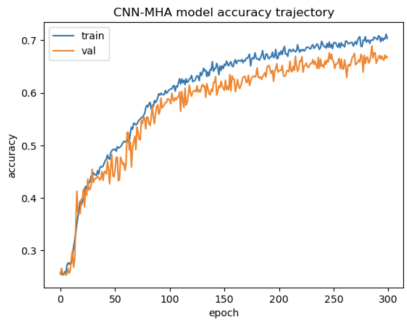


Figure 4. Test accuracy for hybrid CNN-MHA model

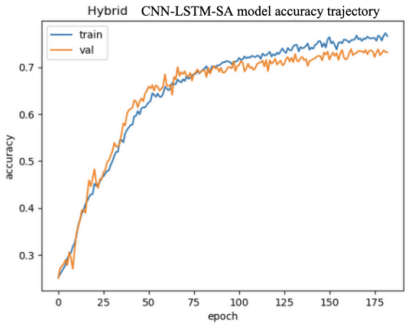
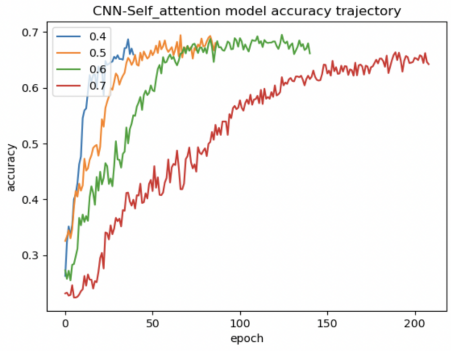


Figure 5. Test accuracy for hybrid CNN-LSTM-SA model



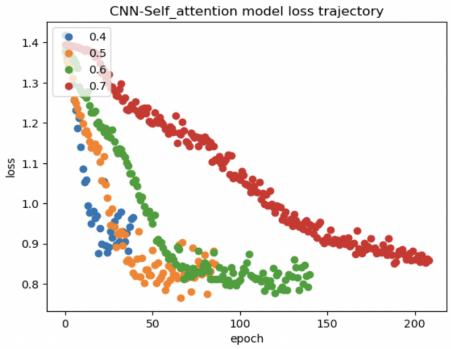


Figure 6. Self-attention model accuracy and model loss trajectory for different dropout values

4. Testing Results

	Validation Accuracy	Test Accuracy
CNN	0.7233	0.7009029
CNN+LSTM	0.6830	0.6783296
CNN+SA	0.7102	0.6805869
CNN+MHA	0.7342	0.7071106
CNN + LSTM + SA	0.7320	0.7042890

Table 1. Validation and test accuracy for CNN, CNN+LSTM, CNN+SA, CNN+MHA, CNN+LSTM+SA

Dropout	**Test Accuracy**
0.4	0.66252821
0.5	0.680033597
0.6	0.68171554804
0.7	0.6551918983

Table 2. Test accuracies for CNN with Self-attention with varying values of dropout

5. Appendix

The below sections describe the details of the architecture for each model and provide further details on the training process.

5.1 Training details

We split 20% of the training dataset to be used as the validation dataset with the remaining 80% as the training data. All models used the Adam optimizer with a learning rate set to 1e-3. Prior to training, all data was prepared in the same process: trimming, max-pooling, averaging, adding noise, and subsampling. We also employed early stopping with a patience of 20.

5.2 CNN

The CNN model consists of 4 convolution blocks with each one performing Conv2D, MaxPooling2D, BatchNormalization, and Dropout. The first filter had a value of 15-30, the second was 40-60, the third was from 80-120, and the final one was 180-220. All layers used ELU as their activation function. Finally, on the output layer, we flattened the input, and output the FC layer with a softmax activation. A dropout value of 0.5 was used for each layer.

5.3 CNN+LSTM

The CNN+LSTM model used a similar structure to the CNN model above, but dropout values of 0.6 were used instead. Furthermore, the first two convolution blocks used Swish as their activation functions and the remaining two convolution blocks used GELU. After the fourth layer, we applied a flattening operation to the output of the CNN block, added an FC layer with 100 units, and then reshaped the output of the FC layer to (120, 1). Finally, this was fed into the LSTM with a dropout value of 10, a recurrent dropout value of 0.1, and a size of 10). The output of the LSTM is then connected to the softmax activation.

5.4 CNN+Self-attention

For the CNN+Self-attention model, we fed the output of the fourth CNN block to the self-attention block. The self-attention block used GELU as its activation function and applied a GlobalAveragePooling1D before connecting to the output layer.

5.5 CNN+MHA

The CNN+MHA is similar to the above model, but CNN+MHA used an MHA instead of the self-attention block. We used 8 heads for the MHA with LayerNormalization (epsilon set to 1e-6) and GlobalAveragePooling2D.

5.6 CNN+LSTM+Self-attention

This model simply used the LSTM block after the fourth CNN block followed by the Multi-head attention block. We used an LSTM with units equal to 124, a dropout rate of 0.6, and a recurrent dropout rate of 0.1. The Multi-head attention block had 8 heads and had LayerNormalization with an epsilon of 1e-6 and GlobalAveragePooling1D.

6. References

Hou, X., Zhang, H., & Jiang, J. (2021). Numerical Simulation of 2D Electron Gas under a Magnetic Field in a Quantum Dot with an Artificial Neural Network. Journal of Physics: Conference Series, 1732(1), 012026. https://doi.org/10.1088/1742-6596/1732/1/012026

Kim, D., Kim, J., & Kang, S. (2020). Deep learning-based EEG analysis for sleep stage classification: A systematic review. arXiv preprint arXiv:2006.16362.