2-Layer Convolutional Neural Network for EEG Motor Imagery Classification

Sarthika Chimmula
University of California, Los Angeles
Computational and Systems Biology
sarthikac@g.ucla.edu

Abstract

In this research endeavor, the primary objective is to develop and evaluate diverse convolutional neural network (CNN), Long Short-Term Memory (LSTM), hybrid CNN-LSTM, and hybrid CNN-gated recurrent unit (GRU) architectures for the prediction of motor imagery tasks based on electroencephalogram (EEG) data. The dataset comprises EEG recordings from 22 electrodes, sampled at 1000 discrete time steps [1]. A total of 2115 trials were conducted, with each trial originating from a distinct test subject. The experimental cohort consisted of 9 subjects actively participating in the study.

We systematically examined various architectures, ultimately identifying the vanilla 2-layer CNN as the best performing architecture. This configuration demonstrated the highest accuracy, achieving a 75.2% on test data. The findings not only underscore the efficacy of CNNs in decoding EEG patterns related to motor imagery tasks but also highlight the significance of selecting appropriate architecture in enhancing predictive performance.

1. Introduction

The choice of neural network architectures for decoding EEG data is influenced by the unique characteristics of the data and the nature of the tasks involved.

CNNs were chosen to be analyzed since they are adept at capturing spatial relationships within multidimensional data, making them well-suited for extracting relevant features from electrode arrays. They also possess translation-invariant properties, which can be advantageous when dealing with EEG data that may exhibit variability in electrode placements across subjects. This allows the network to learn relevant features regardless of spatial shifts. Previously, a deep ConvNet, as well as a shallow ConvNet were implemented in the Brain-Computer Interface (BCI) competition using the same EEG dataset and achieved high performances [2].

LSTM architecture was also chosen to be implemented on this data since they capture temporal dependencies inherent in EEG signals, and this may be useful since the EEG data we have is separated across time steps. LSTMs, as standalone models, are well-suited for tasks where the sequential nature of the data is paramount. If the temporal aspect of EEG signals is the primary focus and spatial relationships are less critical, using LSTMs alone can be a suitable choice.

A hybrid CNN-LSTM model was also implemented as well since motor imagery tasks involve both spatial and temporal dynamics. Combining CNNs with LSTMs enables the model to capture both spatial and temporal dependencies in the EEG data. LSTMs, with their ability to capture long-range dependencies, complement the local feature extraction capabilities of CNNs, offering a more comprehensive understanding of EEG data. In a past study, LConvnet, a hybrid CNN-LSTM architecture, actually outperformed other EEG classification models, including EEGNet, Deep ConvNet and Shallow ConvNet, in detecting epilepsy through EEG signals [4].

The last architecture that is implemented is a CNN-GRU architecture which are equipped with gating mechanisms that regulate the flow of information within the network. These mechanisms, namely the update gate and reset gate, help the network to selectively update and forget information from previous time steps, which can be beneficial for handling noisy or redundant information in EEG data. They also have fewer parameters than LSTMs, which can make them faster to train and less prone to overfitting, especially when dealing with limited EEG data. The motivation behind using this architecture was that previously, a hierarchical bidirectional GRU model showed more robust classification performance than baseline models on EEG-based emotion classification [5]. Bidirectional GRUs can effectively capture contextual information from both past and future time steps. This can be particularly beneficial when dealing with sequences where both past and future context are important for making predictions or classifications, such as the EEG dataset.

Ultimately, the choice of architecture depends on the specific characteristics of the EEG data, the nature of the motor imagery tasks, and the desired trade-off between spatial and temporal information processing.

1.1. Data Preprocessing

Various data preprocessing and cleaning methodologies were explored and contrasted. Initially, a reduction in the number of time bins from 1000 to 800 was executed to streamline computational efficiency while preserving crucial temporal intricacies. However, our experimentation revealed that maintaining all 1000 timesteps resulted in a slightly superior test accuracy, likely due to its capacity to capture a more comprehensive dataset encompassing additional timesteps.

Subsequently, a strategic approach to data augmentation was undertaken by employing maxpooling and average pooling techniques along the temporal dimension, utilizing a subsample factor of 2 and introducing Gaussian noise. This augmentation strategy resulted in an augmented dataset containing four times the original number of trials, thereby enriching the diversity and robustness of the training data.

Upon dataset augmentation, the data was partitioned into training and validation subsets utilizing 80-20, 85-15, and 90-10 splits. Remarkably, the 90-10 split, when employed on the optimal architecture, a two-layer Convolutional Neural Network (CNN), exhibited superior test accuracy compared to the alternative splits.

However, it was discerned that conducting preprocessing prior to data split introduces a latent issue of data leakage. This phenomenon occurs when information from the validation set inadvertently influences preprocessing decisions, leading to an inflated evaluation of model performance. Consequently, validation accuracies consistently surpassed training accuracies, with subsequent test accuracy falling notably short of validation accuracy.

To circumvent the inherent challenge of data leakage, a proactive approach was adopted, wherein the train/validation split was executed before preprocessing. This method ensured the preservation of data integrity and facilitated a more accurate evaluation of model performance.

1.2. 2-Layer CNN Architecture

The 2-Layer CNN architecture is shown in Figure 4. It consists of two convolutional blocks, each of which consists of a 2D-convolutional layer with kernel size of (5,5) with the exponential linear unit used as the activation function, then a maxpool layer with pool size of (3,1), followed by a batch normalization layer and a dropout layer. The first convolutional layer had 50 filters and the second layer had 100 filters. The last layer of the architecture was a fully connected layer with the softmax activation function

1.3. 2-Layer LSTM Architecture

2 layers were also used for the vanilla LSTM model as well, since using more than 2 layers would both be computationally expensive as well as may lead to overfitting. Each LSTM layer was followed by the tanh activation function, and then a dropout layer of 0.5. Again, the last layer of the architecture was a fully connected layer with the softmax activation function. This architecture is shown in Figure 5.

1.4. 4-Layer CNN, 1-Layer LSTM Hybrid Architecture

For the hybrid CNN-LSTM model, a 4-Layer CNN followed by one LSTM layer was implemented. This is shown on Figure 6. Other hybrid architectural combinations, such as 2-Layer CNN followed by one LSTM layer and two LSTM layers followed by 1 CNN layer, were also analyzed, but this architecture produced the best performance. This architecture had 4 convolutional blocks, each consisting of a 2D-convolutional layer, a maxpool layer, a batch normalization layer, and a dropout layer, with 25, 50, 75, and 100 filters used in the first, second, third, and fourth layers respectively. Then, after these convolutional blocks was a single LSTM layer with dropout of 0.4. Again, the last layer was a fully connected layer with softmax activation.

1.5. 4-Layer CNN, 1-Layer GRU Hybrid Architecture

For the hybrid CNN-GRU model, the same architecture as the 4-Layer CNN, 1-Layer LSTM architecture was implemented, except the LSTM layer was replaced by a bidirectional GRU layer with 128 units and a dropout of 0.4 to see if this had a difference in performance. Again, the last layer was a fully connected layer with softmax activation function. This is shown in Figure 7.

2. Results

The 2-layer Convolutional Neural Network (CNN) emerged as the top-performing architecture, achieving a notable test accuracy of 75.17% when trained and evaluated across all subjects. Its training accuracy on the last epoch (epoch 150) was 92.37% and the validation accuracy was 71.01%. This performance surpassed its counterparts, namely the Long Short-Term Memory (LSTM), hybrid CNN-LSTM, and CNN-Gated Recurrent Unit (GRU) architectures, which yielded test accuracies of 35.34%, 69.72%, and 65.45% respectively.

The optimized hyperparameters for the 2-layer CNN comprised the following configurations: SGD optimizer, categorical cross-entropy loss function, learning rate of 1e-3, 150 epochs, batch size of 32, Exponential Linear Unit (ELU) activation function, a 92-8 train-validation split, and

weight decay set to 0.01. These parameters collectively contributed to the CNN's robust performance in image classification tasks.

2.1. Training on One Subject, Testing on Same Subject and All Subjects

See Figure 1. Following the determination of the best performing model, the 2-layer CNN, subsequent analysis focused exclusively on its performance. Specifically, the test accuracy derived from training on a single subject and subsequently testing on the same subject was contrasted with the scenario where the training subject was distinct from the test subject, encompassing all subjects involved in the experiment.

Across all instances where training and testing involved a single subject, the average test accuracy was 54.67%. Additionally, across all instances where training occurred on a single subject while all subjects were included in the test dataset, the average test accuracy was 36.62%. These averages serve to provide comprehensive insights into the model's generalization capabilities under different experimental conditions. Moreover, it was observed that in all cases where the test subject coincided with the corresponding training subject, the test accuracies were consistently higher compared to scenarios where the model was evaluated on all subjects collectively.

2.2. Training on All Subjects, Testing on One Subject and All Subjects

See Figure 2. We further examined test accuracies by training the model on the entirety of the dataset and subsequently testing it on each individual subject. Remarkably, the test accuracies obtained in this scenario consistently surpassed 70%. When compared to the test accuracy achieved in Figure 1, where a single subject was employed for both training and testing, these accuracies notably exhibited a marked improvement. This discrepancy highlights the robustness and effectiveness of the model when trained on a diverse dataset encompassing multiple subjects, as opposed to training on individual subjects in isolation.

2.3. Test Accuracy Over Time Steps

See Figure 3. In our analysis, we also examined the impact of increasing the number of time steps on test accuracy. As we increased the frequency of data updates, we observed a corresponding improvement in test accuracy. This trend suggests that the timeliness and frequency of data updates positively influence the model's ability to generalize and make accurate predictions and suggest that all 1000 time steps in the data are important to the model’s predictive performance.

3. Discussion

In the context of EEG data for motor imagery classification, the superior performance of CNNs over LSTMs may stem from the nature of the data itself, where spatial features play a critical role compared to temporal features. EEG signals are primarily characterized by their spatial distribution across scalp electrodes, representing localized neural activity. CNNs excel at capturing spatial patterns due to their hierarchical feature extraction process through convolutional layers. These spatial patterns correspond to distinct neural activations associated with different motor imagery tasks. In contrast, while LSTMs are adept at modeling temporal dependencies over sequential data, they may struggle to effectively capture the intricate spatial correlations present in EEG signals. Thus, the emphasis on spatial features in EEG data likely aligns more closely with the strengths of CNN architectures, leading to their superior performance in motor imagery classification tasks.

Among these architectures, the vanilla LSTM model exhibited the poorest performance. This outcome aligns with expectations, as the absence of a CNN component limits the model's ability to effectively capture spatial features within the images.

Hyperparameter optimization played a crucial role in enhancing the models' performance. Surprisingly, the Stochastic Gradient Descent (SGD) optimizer outperformed the Adam optimizer for the 2-layer CNN architecture, leading to an approximate 2% boost in test accuracy. This phenomenon could be attributed to the relatively smaller batch size and the inherent noise introduced by SGD, which might facilitate exploration of a wider range of weight configurations, ultimately aiding in escaping from local minima and achieving better generalization performance.

Conversely, the Adam optimizer demonstrated superior performance in the Hybrid CNN-LSTM architecture. This may be because Adam incorporates inherent regularization effects through the adaptive learning rates, which help prevent overfitting in complex models like Hybrid CNN-LSTM.

Overall, achieving high test accuracy when training on all subjects and testing on one subject suggests that the model effectively captures common features, and that the chosen architecture is robust for the task. However, analysis of the consistently low test accuracies for single-subject training and testing on all subjects suggests that the dataset may exhibit heterogeneity across subjects. Therefore, careful consideration is needed to ensure that the model's performance generalizes well to unseen subjects and isn't solely reliant on subject-specific characteristics present in the training data.

References

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

[2] Schirrmeister, R.T., Springenberg, J.T., Fiederer, L.D.J., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W. and Ball, T. (2017), Deep learning with convolutional neural networks for EEG decoding and visualization. Hum. Brain Mapp., 38: 5391-5420. https://doi.org/10.1002/hbm.23730

[3] Omar SM, Kimwele M, Olowolayemo A, Kaburu DM. Enhancing EEG signals classification using LSTM-CNN architecture. Engineering Reports. 2023;e12827. doi: 10.1002/eng2.12827

[4] J. X. Chen, D. M. Jiang and Y. N. Zhang, "A Hierarchical Bidirectional GRU Model With Attention for EEG-Based Emotion Classification," in IEEE Access, vol. 7, pp. 118530-118540,2019,doi:10.1109/ACCESS.2019.2936817. keywords: {Electroencephalography; Brain modeling; Logic gates ;Feature extraction; Data models; Adaptation models; Deep learning; Hierarchical; bidirectional GRU;attention; EEG;emotion classification}

Subjects Used for Training	Subjects Used for Testing	Test Accuracy
0	0	50.00%
0	0, 1, 2, 3, 4, 5, 6, 7, 8	37.25%
1	1	48.00%
1	0, 1, 2, 3, 4, 5, 6, 7, 8	34.76%
2	2	48.00%
2	0, 1, 2, 3, 4, 5, 6, 7, 8	36.12%
3	3	45.00%
3	0, 1, 2, 3, 4, 5, 6, 7, 8	35.67%
4	4	72.34%
4	0, 1, 2, 3, 4, 5, 6, 7, 8	41.54%
5	5	61.22%
5	0, 1, 2, 3, 4, 5, 6, 7, 8	33.86%
6	6	70.00%
6	0, 1, 2, 3, 4, 5, 6, 7, 8	45.15%
7	7	40.00%
7	0, 1, 2, 3, 4, 5, 6, 7, 8	30.70%
8	8	57.45%
8	0, 1, 2, 3, 4, 5, 6, 7, 8	34.54%

Figure 1: Comparison of Test Accuracies Between Training One Subject and Testing on Same Subject vs. Training on One Subject and Testing on All Subjects

Subjects Used for Training	Subjects Used for Testing	Test Accuracy
0, 1, 2, 3, 4, 5, 6, 7, 8	0, 1, 2, 3, 4, 5, 6, 7, 8	75.19%
0, 1, 2, 3, 4, 5, 6, 7, 8	0	71.25%
0, 1, 2, 3, 4, 5, 6, 7, 8	1	74.24%
0, 1, 2, 3, 4, 5, 6, 7, 8	2	80.00%
0, 1, 2, 3, 4, 5, 6, 7, 8	3	74.00%
0, 1, 2, 3, 4, 5, 6, 7, 8	4	78.72%
0, 1, 2, 3, 4, 5, 6, 7, 8	5	77.55%
0, 1, 2, 3, 4, 5, 6, 7, 8	6	72.00%
0, 1, 2, 3, 4, 5, 6, 7, 8	7	80.00%
0, 1, 2, 3, 4, 5, 6, 7, 8	8	72.00%

Figure 2: Comparison of Test Accuracies Between Training on All Subjects, Testing on All subjects and Training on All Subjects, Testing on One Subject

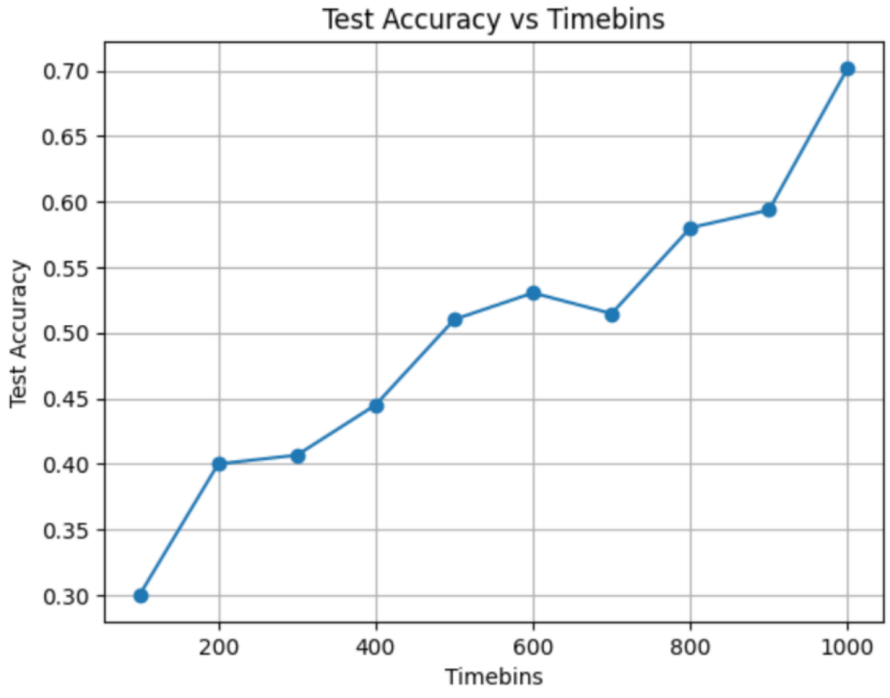

Figure 3: Plot of Test Accuracy vs Timebins over every 100 Timebins for 2-layer CNN. Test Accuracy increases as the number of timebins increases.

Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 500, 1, 50)	27550
max_pooling2d_13 (MaxPooling2D)	(None, 167, 1, 50)	0
batch_normalization_13 (BatchNormalization)	(None, 167, 1, 50)	200
dropout_13 (Dropout)	(None, 167, 1, 50)	0
conv2d_14 (Conv2D)	(None, 167, 1, 100)	125100
max_pooling2d_14 (MaxPooling2D)	(None, 56, 1, 100)	0
batch_normalization_14 (BatchNormalization)	(None, 56, 1, 100)	400
dropout_14 (Dropout)	(None, 56, 1, 100)	0
flatten_5 (Flatten)	(None, 5600)	0
dense_5 (Dense)	(None, 4)	22404

Total params: 175654 (686.15 KB)
Trainable params: 175354 (684.98 KB)
Non-trainable params: 300 (1.17 KB)

Figure 4: 2-Layer CNN Architecture

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 500, 100)	49200
dropout (Dropout)	(None, 500, 100)	0
lstm_1 (LSTM)	(None, 100)	80400
dropout_1 (Dropout)	(None, 100)	0
dense (Dense)	(None, 4)	404

Total params: 130004 (507.83 KB)
Trainable params: 130004 (507.83 KB)
Non-trainable params: 0 (0.00 Byte)

Figure 5: 2-Layer LSTM Architecture

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 500, 1, 25)	13775
max_pooling2d_5 (MaxPooling2D)	(None, 167, 1, 25)	0
batch_normalization_5 (BatchNormalization)	(None, 167, 1, 25)	100
dropout_7 (Dropout)	(None, 167, 1, 25)	0
conv2d_6 (Conv2D)	(None, 167, 1, 50)	31300
max_pooling2d_6 (MaxPooling2D)	(None, 56, 1, 50)	0
batch_normalization_6 (BatchNormalization)	(None, 56, 1, 50)	200
dropout_8 (Dropout)	(None, 56, 1, 50)	0
conv2d_7 (Conv2D)	(None, 56, 1, 75)	93825
max_pooling2d_7 (MaxPooling2D)	(None, 19, 1, 75)	0
batch_normalization_7 (BatchNormalization)	(None, 19, 1, 75)	300
dropout_9 (Dropout)	(None, 19, 1, 75)	0
conv2d_8 (Conv2D)	(None, 19, 1, 100)	187600
max_pooling2d_8 (MaxPooling2D)	(None, 7, 1, 100)	0
batch_normalization_8 (BatchNormalization)	(None, 7, 1, 100)	400
dropout_10 (Dropout)	(None, 7, 1, 100)	0
flatten_3 (Flatten)	(None, 700)	0
dense_6 (Dense)	(None, 40)	28040
reshape_3 (Reshape)	(None, 40, 1)	0
lstm_5 (LSTM)	(None, 10)	480
dense_7 (Dense)	(None, 4)	44

Total params: 356064 (1.36 MB)
Trainable params: 355564 (1.36 MB)
Non-trainable params: 500 (1.95 KB)

Figure 6: Hybrid 4 Layer CNN-1 Layer LSTM Architecture

Layer (type)	Output Shape	Param #
conv2d_74 (Conv2D)	(None, 500, 1, 32)	17632
max_pooling2d_72 (MaxPooling2D)	(None, 167, 1, 32)	0
batch_normalization_72 (BatchNormalization)	(None, 167, 1, 32)	128
dropout_72 (Dropout)	(None, 167, 1, 32)	0
conv2d_75 (Conv2D)	(None, 167, 1, 64)	51264
max_pooling2d_73 (MaxPooling2D)	(None, 56, 1, 64)	0
batch_normalization_73 (BatchNormalization)	(None, 56, 1, 64)	256
dropout_73 (Dropout)	(None, 56, 1, 64)	0
conv2d_76 (Conv2D)	(None, 56, 1, 96)	153696
max_pooling2d_74 (MaxPooling2D)	(None, 19, 1, 96)	0
batch_normalization_74 (BatchNormalization)	(None, 19, 1, 96)	384
dropout_74 (Dropout)	(None, 19, 1, 96)	0
conv2d_77 (Conv2D)	(None, 19, 1, 128)	307328
max_pooling2d_75 (MaxPooling2D)	(None, 7, 1, 128)	0
batch_normalization_75 (BatchNormalization)	(None, 7, 1, 128)	512
dropout_75 (Dropout)	(None, 7, 1, 128)	0
flatten_17 (Flatten)	(None, 896)	0
dense_33 (Dense)	(None, 40)	35880
reshape_17 (Reshape)	(None, 40, 1)	0
bidirectional_17 (Bidirectional)	(None, 256)	100608
dense_34 (Dense)	(None, 4)	1028

Total params: 668716 (2.55 MB)
Trainable params: 668076 (2.55 MB)
Non-trainable params: 640 (2.50 KB)

Figure 7: Hybrid 4 Layer CNN-1 Layer GRU Architecture