Neural Network Architectures for EEG Action Classification

Timothy Do
UID: 406302424
timothydo@ucla.edu

Brandon Kam
UID: 305684721
bkam@g.ucla.edu

Josh McDermott
UID: 305555111
jmcder000@ucla.edu

Steve Zang
UID: 205753022
zangbruin007@g.ucla.edu

Abstract

EEG classification plays a crucial role in various applications such as brain-computer interfaces, healthcare, and neuroscientific research. This paper presents an investigation into the evaluation of classification performance over time in EEG classification architectures applied on the BCI dataset. We discuss the methods, results, and implications of evaluating classification as a function of time, providing insights for future research in this domain. Overall, we have found that with data preprocessing augmentation via subsampling, mean/max pooling, and noise injections, the EEGNet model achieves an overall accuracy of 75.4% across all subjects.

1. Introduction

EEG classification plays a crucial role in various applications such as brain-computer interfaces, healthcare, and neuroscientific research. In fields of computer vision, the progressive development of convolutional neural networks (CNNs), recurrent neural networks (RNNs) and vision transformers have all resulted in each improving accuracy in the analogous task of image classification [1]. This paper presents an investigation into the evaluation of classification architectures including CNN, RNN, and Transformer types. Specifically, this paper investigates the optimization of test accuracies among all subjects, among individual subjects, across time samples, and with model ensembling. These experiments were conducted on the BCI dataset [2] for motor imagery action classification.

1.1. Data Preprocessing and Augmentation

Before the BCI dataset is fed into the model, the data gets preprocessed to make learning easier. To ensure that the models only learn from the most relevant features, the EEG signals get trimmed to only contain the first *N* time samples, where *N* is a tunable hyperparameter for each model. The trimmed signals would then undergo a maxpooling of every two samples in order for our models to focus on learning from the trends of the time-series data.

Because the BCI training and validation dataset is small, data augmentation must be done to achieve better accuracy. We have performed two forms of data augmentation on the training dataset. Instead of maxpooling the data in preprocessing, we take the average of every two samples and add random Gaussian noise. Additionally, we subsample the original time-series data by taking every odd time sample then every even time sample and adding random Gaussian noise. By utilizing different low-pass filters and adding noise to the data, we ensure that our models are only focused on learning from the trends of the data instead of the signal amplitudes.

1.2. Training

We used the Adam optimizer to train all of our models. Hyperparameters were tuned using a grid search over the optimizer learning rate (from 1e-4 to 1e-2), batch size (from 64 to 1024), number of epochs (up to 250), and the number of time samples in the data.

1.3. Neural Network Architectures Overview

1.3.1 BasicCNN

The BasicCNN (DeepConvNet) architecture [3] is utilized in this study for EEG classification. This architecture comprises multiple convolutional layers followed by maxpooling layers and fully connected layers. To evaluate the classification performance of BasicCNN over time, the EEG data is divided into temporal windows or epochs. Each epoch undergoes processing by the BasicCNN model to classify EEG signals into predefined classes.

1.3.2 HybridCNN-LSTM

The HybridCNN-LSTM architecture combines convolutional and recurrent layers, leveraging the strengths of CNNs and LSTMs (Long Short-Term Memory) for EEG classification, as shown effective [4]. In this study, we investigate the performance of HybridCNN-LSTM in classifying EEG signals over time by combining the convolutional features from the BasicCNN with an LSTM layer for temporal learning.

1.3.3 EEGNet

The EEGNet [5] architecture is a CNN-based architecture that utilizes both Depthwise convolutions and Separable convolutions to learn about the frequency-spatial features EEG data. The input data is first passed through temporal convolution to learn frequency filters, then through depthwise convolution to learn F_1 frequency-spatial feature maps. Afterwards, separable convolutions are used to individually learn the kernel map for each feature using F_2 pointwise convolutions to optimally combine the outputs together.F_1 & F_2 are architectural hyperparameters varying the frequency-spatial resolution and temporal dependence for learning the features across different EEG classes. In this study, we investigate the performance of EEGNet using the implementation from [6].

1.3.4 EEGNet-LSTM

The EEGNet-LSTM [7] architecture applies an LSTM layer after the output of the separable convolutional layers in EEGNet to learn the temporal dependence of EEG features across training time. The LSTM learns from the temporal summary from each individual feature map outputted from the separable convolutions. The motivation is that by combining the temporal summary from EEGNet's separable convolutions and learning of temporal sequences from LSTM, classification accuracy will increase by learning more about the temporal sequence.

1.3.5 EEGConformer

The EEGConformer architecture is a transformer convolutional based architecture tailored for EEG decoding [8]. It consists of a convolution layer, a self-attention head, and a fully-connected output. The purpose of integrating the self-attention head in between the convolutional layer and the classifier output is because with self-attention the limited receptive field in the convolutional layer can be assisted in learning the global temporal dependencies across neurons. The Braindecode [9] library was imported for its implementation of EEGConformer, and the dataset was preprocessed to be compatible with its input shape.

Table 1. Best Test Accuracy with Finetuned Models

Model	Accuracy	Chunk	Batch	Step
BasicCNN	65.50%	300	64	1E-03
HybridCNN-LSTM	70.40%	450	64	1E-04
EEGNet	75.40%	400	64	1E-04
EEGNet-LSTM	75.20%	300	128	1E-03
EEGConformer	71.10%	400	128	2E-04

2. Results

2.1. Optimizing Accuracy Across all Subjects

Referring to Table 1, the highest testing accuracy was achieved by EEGNet, with a result of 75.4%. EEGNet-LSTM followed closely with an accuracy of 75.2%. HybridCNN-LSTM and EEGConformer performed similarly, achieving accuracy of 70.4% and 71.1%, respectively. BasicCNN had the lowest test accuracy among the models, with a result of 65.5%.

2.2. Optimizing Accuracy for Only Subject 1

EEGNet was chosen for this experiment because of its adaptability to be trained with a limited number of samples. The training and validation set was optimally split into 90 and 10 percent. Optimizing the model for subject 1 (index 0) yielded a choice of hyperparameters that had lower batch size (i.e., from 64 to 16), lower F_1 & F_2 from the overall model (i.e. both from 16 to 8), and a choice of no data augmentation being applied. This configuration yielded 70% accuracy on subject 1 test samples and 39% when applied to the entire test set.

2.3. Evaluating Classification as a Function of Time

The graphs in figure 5 show the test accuracy of each of our models as a function of the number of time samples in the data. According to the graphs, the test accuracy peaks at around 600 - 800 time samples before slowly falling off at around 800 - 1000 time samples across all models.

2.4. Ensembling of Models across Subjects

Table 2. Individual/Ensemble Accuracies on Overall Dataset

Subject	Accuracy	Subject	Accuracy
1	0.44	6	0.38
2	0.36	7	0.41
3	0.47	8	0.42
4	0.35	9	0.41
5	0.42	**Ensemble**	**0.56**

Training separate EEGNets on individual subjects occasionally yielded models that performed better on that same subject's test data (when compared to a comprehensive model with identical testing). For example, Table 3 shows subject 5's 80.9% test accuracy compared to 74% from the comprehensively trained model. In Table 2, we see that the best attempt at ensembling these models yielded an overall accuracy of only 56%, using majority voting on an EEGNet ensemble.

3. Discussion

3.1. Optimizing Accuracy Across All Subjects

Based on Table 1, EEGNet achieves the highest accuracy in compare to all other models, mainly attribute to its depthwise separable convolutional layers: depthwise convolution and pointwise convolution. Depthwise convolution applies a separate convolutional filer to each channel of the input data independently, which helps capture spatial information within each channel. Pointwise Convolution then applied to combine the information across channels, which helps in learning inter-channel relationships. Thus, EEGNet effectively processes EEG data by extracting discriminative features, which lead to a high accuracy. In addition, adding an LSTM layer in EEGNet-LSTM doesn't always lead to better results. EEGNet is already good at extracting temporal features using separable convolutions, so the LSTM adds unnecessary complexity without improving accuracy.

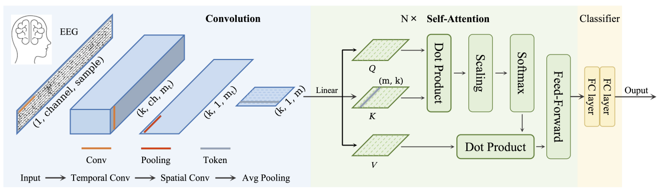


Figure 1. EEG Conformer architecture from [8].

In contrast, our initial expectation was that EEGConformer would yield high accuracy because it leverages the self-attention mechanism to capture long-range dependencies by allowing each timestep of data to attend to all others. However, our results show that EEGConformer was only our third best performing model. EEGConformer, being based on a Transformer, requires significantly more training data than what we had after data augmentation (7000 samples). Additionally, the effectiveness of self-attention is hindered by the small timestep of 400, which limits the model's ability to fully utilize the temporal information and may result in an increased risk of overfitting, as evidenced by the trends observed in Figure 4. The declining training loss contrasted with the gradually increasing validation loss indicates overfitting, potentially compromising its generalization capability.

Across all the models, the modified HybridCNN-LSTM has the most reasonable loss curves since its validation loss gradually decrease along with its training loss (Figure 4). To achieve this improvement, we made two key modifications on the original HybridCNN-LSTM presented in C247 discussion 7: We removed the FC layer from the last Conv. block to the start of LSTM to maintain sequence order, and switching from 2D to 1D Conv. kernels preserved spatial information, preventing sequence disorder in the output as shown in Figure 7.

3.2. Optimizing Accuracy for Only Subject 1

From the results in 2, it is evident that the EEGNet model trained only on subject 1 doesn't generalize well to the rest of the subject's testing data. Training on limited data is heavily prone to overfitting, and it involves decreasing model complexities in hyperparameters (e.g., F_1 & F_2) to attenuate this. Decreasing batch size in order to optimize test accuracy results in noisy gradient estimates seen in the loss trajectories in Figure 2. Data augmentation on limited data is heavily biased towards the training set. Only learning the features of a particular subject with noisy gradients doesn't account for the subject to subject variation in features, as shown in Figure 3 when evaluating the best overall EEGNet model onto each subject individually.

3.3. Evaluating Classification as a Function of Time

For a few time samples, the test accuracy is low because there are not enough features for any of the models to generalize well for the overall dataset. As more features get exposed to the models, the test accuracy starts to increase up to a certain point at around 800 time samples. The test accuracy seems to decrease after that because there are too many features, which causes potential overfitting.

3.4. Ensembling of Models across Subjects

Anatomical variations seem plausible to cause different data patterns across subjects. Additionally, when the EEG data was grouped by subject and averaged across the trials, it seemed to be very noisy. Some subjects' average data would follow similar trends, but other subjects had relatively distinctive patterns. This led us to consider exploring personalized modeling, with the potential for ensembling to be used in the end. While any individualized models would obviously be less generalizable in application, there is still use for them, and perhaps some future subject grouping would allow for a middling approach to be fruitful.

To that end, we considered majority voting, as well as a max-confidence voting that simply predicted on the most confident model. We tested ensembling with BasicCNN, HybridCNN-LSTM, and EEGNet. EEGNet performed the best, but ultimately, the results were poor across the architectures. For all ensembles, some of the 9 subject specific models would perform well on their isolated test set, but others would not. See Table 3 for a breakdown for the EEGNet Ensemble. Our suspicion is that there is simply not enough data on any single subject to make this a feasible approach; the individually trained models seem extremely prone to overfitting. However, we still feel that some type of ensembling on more individualized models seems worth exploring in this domain.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020. 1

[2] Clemens Brunner, Robert Leeb, and Gernot Müller-Putz. Bci competition 2008–graz data set a. 2008. 1

[3] 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 brain mapping and decoding of movement-related information from the human EEG. *CoRR*, abs/1703.05051, 2017. 1

[4] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. *Sensors*, 16(1), 2016. 1

[5] Vernon J. Lawhern, Amelia J. Solon, Nicholas R. Waytowich, Stephen M. Gordon, Chou P. Hung, and Brent J. Lance. Eegnet: A compact convolutional network for eeg-based brain-computer interfaces. *CoRR*, abs/1611.08024, 2016. 2

[6] Zhi Zhang, Sheng hua Zhong, and Yan Liu. TorchEEGEMO: A deep learning toolbox towards EEG-based emotion recognition. *Expert Systems with Applications*, page 123550, 2024. 2

[7] Iago Henrique de Oliveira and Abner Cardoso Rodrigues. Empirical comparison of deep learning methods for eeg decoding. *Frontiers in Neuroscience*, 16, 2023. 2

[8] Yonghao Song, Qingqing Zheng, Bingchuan Liu, and Xiaorong Gao. Eeg conformer: Convolutional transformer for eeg decoding and visualization. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 31:710–719, 2023. 2, 3

[9] 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*, aug 2017. 2

A. Experimental Results

Table 3. Subject Test Accuracy, General Model/Subject Models

Subject	Subject Accuracy	Full Model Accuracy
1	62%	72%
2	54%	64%
3	66%	74%
4	68%	66%
5	80.9%	74%
6	65.3%	67%
7	70%	76%
8	64%	70%
9	80.9%	83%

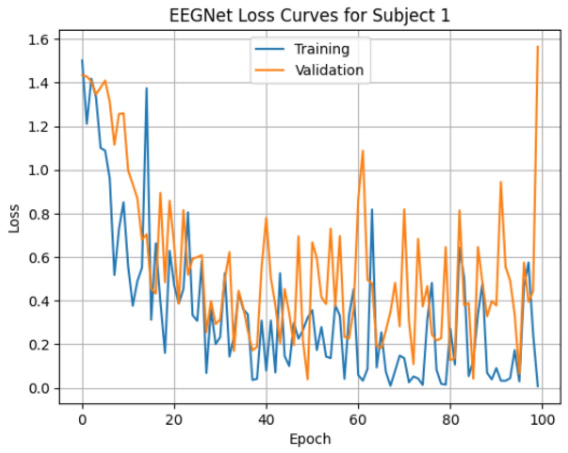


Figure 2. Subject 1 Optimized EEGNet Loss Trajectory

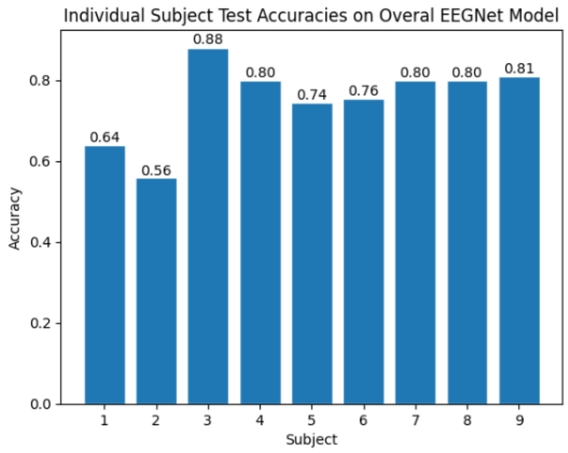


Figure 3. Individual Test Accuracies on Overall EEGNet

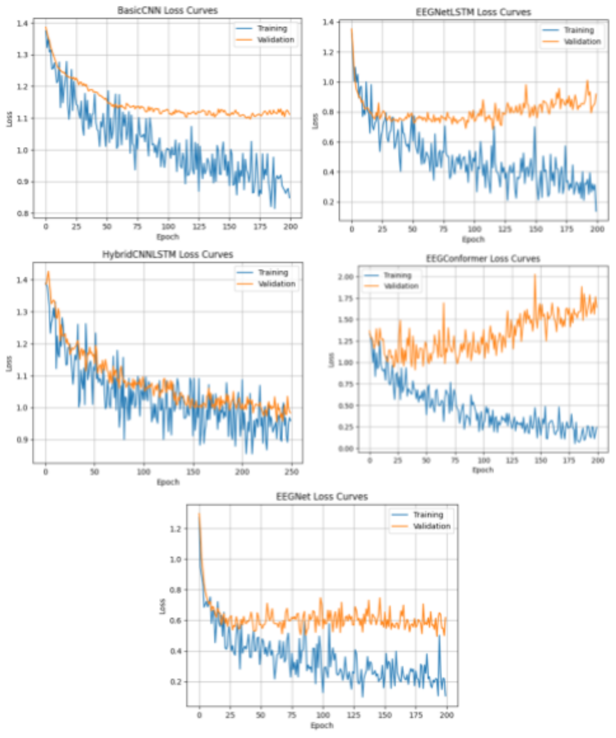


Figure 4. Loss Trajectories of Different Models

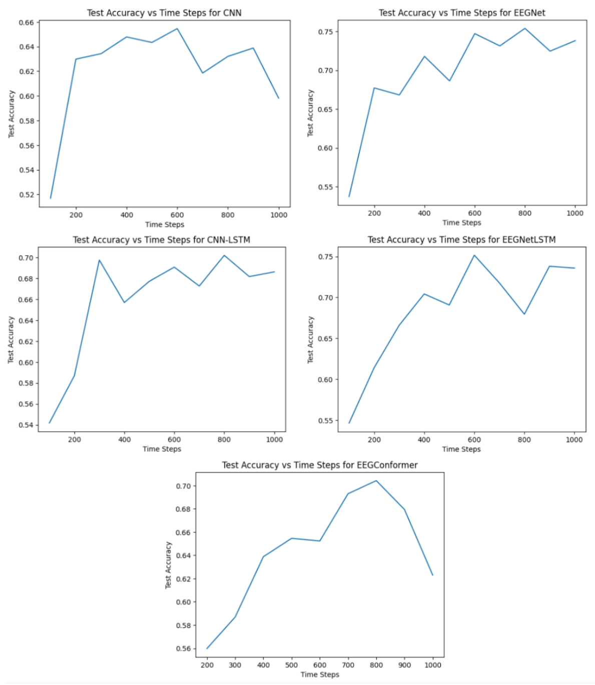
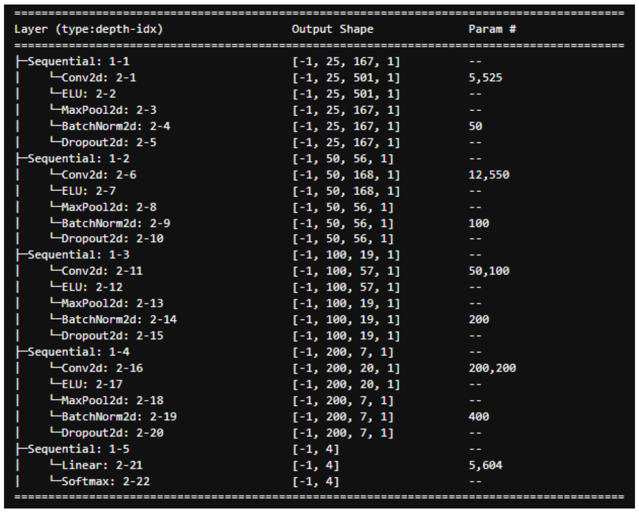


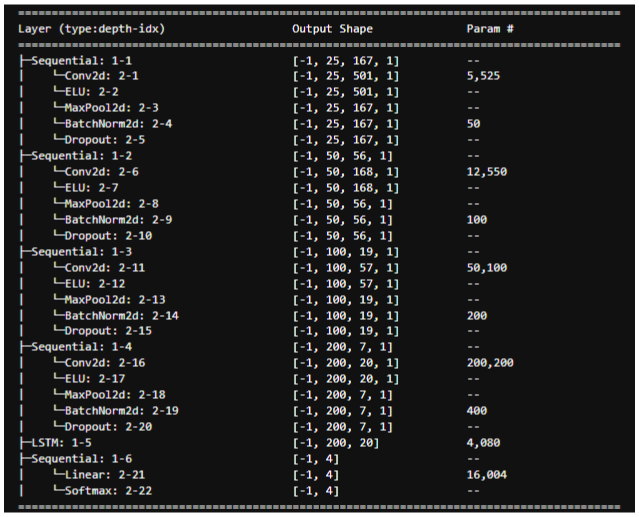
Figure 5. Test Accuracies vs. Time Steps for Different Models

B. Architectures



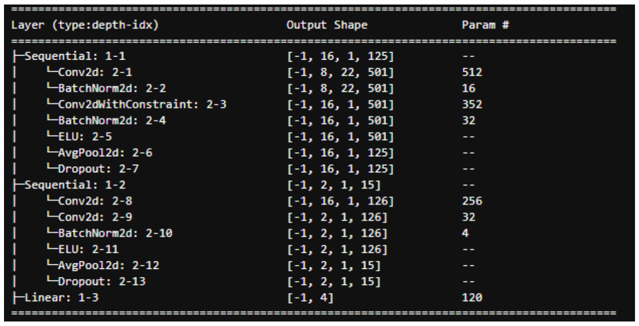
```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
├─Sequential: 1-1                        [-1, 25, 167, 1]          --
|    └─Conv2d: 2-1                       [-1, 25, 501, 1]          5,525
|    └─ELU: 2-2                          [-1, 25, 501, 1]          --
|    └─MaxPool2d: 2-3                    [-1, 25, 167, 1]          --
|    └─BatchNorm2d: 2-4                  [-1, 25, 167, 1]          50
|    └─Dropout2d: 2-5                    [-1, 25, 167, 1]          --
├─Sequential: 1-2                        [-1, 50, 56, 1]           --
|    └─Conv2d: 2-6                       [-1, 50, 168, 1]          12,550
|    └─ELU: 2-7                          [-1, 50, 168, 1]          --
|    └─MaxPool2d: 2-8                    [-1, 50, 56, 1]           --
|    └─BatchNorm2d: 2-9                  [-1, 50, 56, 1]           100
|    └─Dropout2d: 2-10                   [-1, 50, 56, 1]           --
├─Sequential: 1-3                        [-1, 100, 19, 1]          --
|    └─Conv2d: 2-11                      [-1, 100, 57, 1]          50,100
|    └─ELU: 2-12                         [-1, 100, 57, 1]          --
|    └─MaxPool2d: 2-13                   [-1, 100, 19, 1]          --
|    └─BatchNorm2d: 2-14                 [-1, 100, 19, 1]          200
|    └─Dropout2d: 2-15                   [-1, 100, 19, 1]          --
├─Sequential: 1-4                        [-1, 200, 7, 1]           --
|    └─Conv2d: 2-16                      [-1, 200, 20, 1]          200,200
|    └─ELU: 2-17                         [-1, 200, 20, 1]          --
|    └─MaxPool2d: 2-18                   [-1, 200, 7, 1]           --
|    └─BatchNorm2d: 2-19                 [-1, 200, 7, 1]           400
|    └─Dropout2d: 2-20                   [-1, 200, 7, 1]           --
├─Sequential: 1-5                        [-1, 4]                   --
|    └─Linear: 2-21                      [-1, 4]                   5,604
|    └─Softmax: 2-22                     [-1, 4]                   --
==========================================================================================
```

Figure 6. BasicCNN Model Architecture



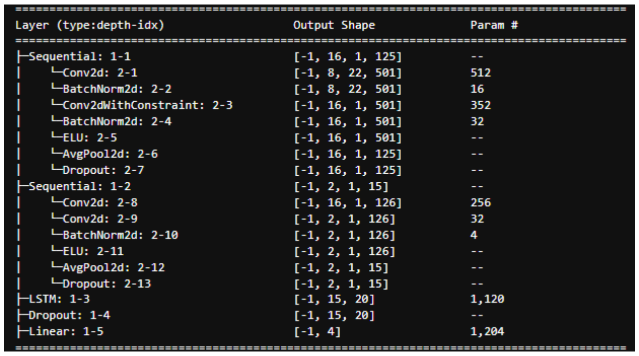
```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
├─Sequential: 1-1                        [-1, 25, 167, 1]          --
|    └─Conv2d: 2-1                       [-1, 25, 501, 1]          5,525
|    └─ELU: 2-2                          [-1, 25, 501, 1]          --
|    └─MaxPool2d: 2-3                    [-1, 25, 167, 1]          --
|    └─BatchNorm2d: 2-4                  [-1, 25, 167, 1]          50
|    └─Dropout: 2-5                      [-1, 25, 167, 1]          --
├─Sequential: 1-2                        [-1, 50, 56, 1]           --
|    └─Conv2d: 2-6                       [-1, 50, 168, 1]          12,550
|    └─ELU: 2-7                          [-1, 50, 168, 1]          --
|    └─MaxPool2d: 2-8                    [-1, 50, 56, 1]           --
|    └─BatchNorm2d: 2-9                  [-1, 50, 56, 1]           100
|    └─Dropout: 2-10                     [-1, 50, 56, 1]           --
├─Sequential: 1-3                        [-1, 100, 19, 1]          --
|    └─Conv2d: 2-11                      [-1, 100, 57, 1]          50,100
|    └─ELU: 2-12                         [-1, 100, 57, 1]          --
|    └─MaxPool2d: 2-13                   [-1, 100, 19, 1]          --
|    └─BatchNorm2d: 2-14                 [-1, 100, 19, 1]          200
|    └─Dropout: 2-15                     [-1, 100, 19, 1]          --
├─Sequential: 1-4                        [-1, 200, 7, 1]           --
|    └─Conv2d: 2-16                      [-1, 200, 20, 1]          200,200
|    └─ELU: 2-17                         [-1, 200, 20, 1]          --
|    └─MaxPool2d: 2-18                   [-1, 200, 7, 1]           --
|    └─BatchNorm2d: 2-19                 [-1, 200, 7, 1]           400
|    └─Dropout: 2-20                     [-1, 200, 7, 1]           --
├─LSTM: 1-5                              [-1, 200, 20]             4,080
├─Sequential: 1-6                        [-1, 4]                   --
|    └─Linear: 2-21                      [-1, 4]                   16,004
|    └─Softmax: 2-22                     [-1, 4]                   --
==========================================================================================
```

Figure 7. HybridCNN-LSTM Architecture



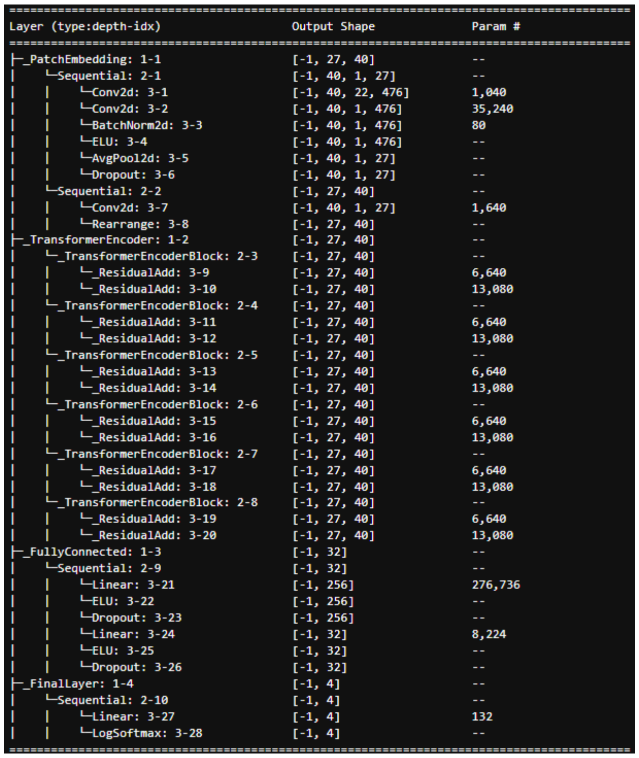
```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
├─Sequential: 1-1                        [-1, 16, 1, 125]          --
|    └─Conv2d: 2-1                       [-1, 8, 22, 501]          512
|    └─BatchNorm2d: 2-2                  [-1, 8, 22, 501]          16
|    └─Conv2dWithConstraint: 2-3         [-1, 16, 1, 501]          352
|    └─BatchNorm2d: 2-4                  [-1, 16, 1, 501]          32
|    └─ELU: 2-5                          [-1, 16, 1, 501]          --
|    └─AvgPool2d: 2-6                    [-1, 16, 1, 125]          --
|    └─Dropout: 2-7                      [-1, 16, 1, 125]          --
├─Sequential: 1-2                        [-1, 2, 1, 15]            --
|    └─Conv2d: 2-8                       [-1, 16, 1, 126]          256
|    └─Conv2d: 2-9                       [-1, 2, 1, 126]           32
|    └─BatchNorm2d: 2-10                 [-1, 2, 1, 126]           4
|    └─ELU: 2-11                         [-1, 2, 1, 126]           --
|    └─AvgPool2d: 2-12                   [-1, 2, 1, 15]            --
|    └─Dropout: 2-13                     [-1, 2, 1, 15]            --
├─Linear: 1-3                            [-1, 4]                   120
==========================================================================================
```

Figure 8. EEGNet Architecture



```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
├─Sequential: 1-1                        [-1, 16, 1, 125]          --
|    └─Conv2d: 2-1                       [-1, 8, 22, 501]          512
|    └─BatchNorm2d: 2-2                  [-1, 8, 22, 501]          16
|    └─Conv2dWithConstraint: 2-3         [-1, 16, 1, 501]          352
|    └─BatchNorm2d: 2-4                  [-1, 16, 1, 501]          32
|    └─ELU: 2-5                          [-1, 16, 1, 501]          --
|    └─AvgPool2d: 2-6                    [-1, 16, 1, 125]          --
|    └─Dropout: 2-7                      [-1, 16, 1, 125]          --
├─Sequential: 1-2                        [-1, 2, 1, 15]            --
|    └─Conv2d: 2-8                       [-1, 16, 1, 126]          256
|    └─Conv2d: 2-9                       [-1, 2, 1, 126]           32
|    └─BatchNorm2d: 2-10                 [-1, 2, 1, 126]           4
|    └─ELU: 2-11                         [-1, 2, 1, 126]           --
|    └─AvgPool2d: 2-12                   [-1, 2, 1, 15]            --
|    └─Dropout: 2-13                     [-1, 2, 1, 15]            --
├─LSTM: 1-3                              [-1, 15, 20]              1,120
├─Dropout: 1-4                           [-1, 15, 20]              --
├─Linear: 1-5                            [-1, 4]                   1,204
==========================================================================================
```

Figure 9. EEGNet-LSTM Architecture



```
==========================================================================================
Layer (type:depth-idx)                   Output Shape              Param #
==========================================================================================
├─_PatchEmbedding: 1-1                   [-1, 27, 40]              --
|    └─Sequential: 2-1                   [-1, 40, 1, 27]           --
|    |    └─Conv2d: 3-1                  [-1, 40, 22, 476]         1,040
|    |    └─Conv2d: 3-2                  [-1, 40, 1, 476]          35,240
|    |    └─BatchNorm2d: 3-3             [-1, 40, 1, 476]          80
|    |    └─ELU: 3-4                     [-1, 40, 1, 476]          --
|    |    └─AvgPool2d: 3-5               [-1, 40, 1, 27]           --
|    |    └─Dropout: 3-6                 [-1, 40, 1, 27]           --
|    └─Sequential: 2-2                   [-1, 27, 40]              --
|    |    └─Conv2d: 3-7                  [-1, 40, 1, 27]           1,640
|    |    └─Rearrange: 3-8               [-1, 27, 40]              --
├─_TransformerEncoder: 1-2               [-1, 27, 40]              --
|    └─_TransformerEncoderBlock: 2-3     [-1, 27, 40]              --
|    |    └─_ResidualAdd: 3-9            [-1, 27, 40]              6,640
|    |    └─_ResidualAdd: 3-10           [-1, 27, 40]              13,080
|    └─_TransformerEncoderBlock: 2-4     [-1, 27, 40]              --
|    |    └─_ResidualAdd: 3-11           [-1, 27, 40]              6,640
|    |    └─_ResidualAdd: 3-12           [-1, 27, 40]              13,080
|    └─_TransformerEncoderBlock: 2-5     [-1, 27, 40]              --
|    |    └─_ResidualAdd: 3-13           [-1, 27, 40]              6,640
|    |    └─_ResidualAdd: 3-14           [-1, 27, 40]              13,080
|    └─_TransformerEncoderBlock: 2-6     [-1, 27, 40]              --
|    |    └─_ResidualAdd: 3-15           [-1, 27, 40]              6,640
|    |    └─_ResidualAdd: 3-16           [-1, 27, 40]              13,080
|    └─_TransformerEncoderBlock: 2-7     [-1, 27, 40]              --
|    |    └─_ResidualAdd: 3-17           [-1, 27, 40]              6,640
|    |    └─_ResidualAdd: 3-18           [-1, 27, 40]              13,080
|    └─_TransformerEncoderBlock: 2-8     [-1, 27, 40]              --
|    |    └─_ResidualAdd: 3-19           [-1, 27, 40]              6,640
|    |    └─_ResidualAdd: 3-20           [-1, 27, 40]              13,080
├─_FullyConnected: 1-3                   [-1, 32]                  --
|    └─Sequential: 2-9                   [-1, 32]                  --
|    |    └─Linear: 3-21                 [-1, 256]                 276,736
|    |    └─ELU: 3-22                    [-1, 256]                 --
|    |    └─Dropout: 3-23                [-1, 256]                 --
|    |    └─Linear: 3-24                 [-1, 32]                  8,224
|    |    └─ELU: 3-25                    [-1, 32]                  --
|    |    └─Dropout: 3-26                [-1, 32]                  --
├─_FinalLayer: 1-4                       [-1, 4]                   --
|    └─Sequential: 2-10                  [-1, 4]                   --
|    |    └─Linear: 3-27                 [-1, 4]                   132
|    |    └─LogSoftmax: 3-28             [-1, 4]                   --
==========================================================================================
```

Figure 10. EEGConformer Architecture