# Deep Neural Network Architectures for EEG Signal Classification

Samuel Chua
University of California, Los Angeles
Computer Science and Engineering
samuelchua2001@gmail.com

Izaac Tay
University of California, Los Angeles
Computer Science
izaac@g.ucla.edu

Rakesh Selvaraj
University of California, Los Angeles
Electrical Engineering
rakeshselvaraj@g.ucla.edu

Alex Puga
University of California, Los Angeles
Electrical Engineering
alexpuga@ucla.edu

## Abstract

*Our paper aims to classify electroencephalography (EEG) data as part of the Brain-Computer Interaction (BCI) competition. We evaluate and compare the effectiveness of different deep learning architectures, including Vanilla Convolutional Neural Networks (CNNs), Deep Convolutional Networks (DeepConvNets), Shallow Convolutional Networks (ShallowConvNets), EEGNet, and hybrid architectures like CNN with Recurrent Neural Networks and Long Short-Term Memory (LSTM) (CNN-RNN-LSTM). Specifically, the architectures are tested on a single subject and with all subjects. Additionally, time duration variations in EEG data are also taken into consideration and compared. Our findings indicate that EEGNet achieved the highest test accuracy of 78%, demonstrating its potential for accurate EEG signal classification across many subjects.*

## 1. Introduction

In this paper, we construct the following networks: Vanilla Convolutional Neural Networks (CNNs), Deep Convolutional Networks (DeepConvNets), Shallow Convolutional Networks (ShallowConvNets), EEGNet, and a hybrid architecture, CNN-RNN-LSTM.

### 1.1. Models

#### 1.1.1 Vanilla Convolutional Neural Network (CNN)

Our Vanilla Convolutional Neural Network consists of four convolutional blocks. Each block integrates convolutional, pooling, normalization, and dropout layers to progressively extract hierarchical features from the input EEG signals. The first convolutional block has 25 filters and an ELU activation function. Subsequent convolutional blocks increase the filter count. The final output layer is a dense layer with softmax activation which classifies the input into one of four output classes. Since the first three layers aim to reduce the time-dimension and the final layer reduce the feature-dimension, the vanilla CNN is a good choice of architecture to learn such features.

#### 1.1.2 Deep Convolutional Networks (DeepConvNet)

Our DeepConvNet has the same basic structure as the vanilla CNN, but consists of more convolutional blocks and trainable parameters. Each block typically comprises a convolutional layer followed by batch normalization, ReLU activation, max-pooling, and dropout layers. This deep structure allows DeepConvNet to learn hierarchical features from the input EEG signals, capturing both low-level temporal patterns and higher-level abstract representations. Additionally, DeepConvNet uses smaller kernel sizes and strides in its convolutional layers compared to Vanilla CNN to capture fine-grained temporal features in EEG signals. It employs 1D convolutions with small kernel sizes to extract local patterns effectively. Lastly, DeepConvNet incorporates batch normalization and dropout layers much more extensively throughout its architecture to accelerate training convergence, improve model generalization, and mitigate overfitting.

#### 1.1.3 Shallow Convolutional Networks (ShallowConvNet)

ShallowConvNet has the same structure as Vanilla CNN but differs in depth of layers. It consists of only two convolutional layers followed by batch normalization, ReLU activation, average pooling, dropout, and fully connected layers. This architecture is simpler and more straightforward. Additionally, ShallowConvNet uses relatively bigger kernel sizes and strides compared to

Vanilla CNN. Therefore, ShallowConvNet captures more extensive spatial information from the input data and preserves dominant features.

#### 1.1.4 EEGNet

EEGNet, inspired by advancements in computer vision [3], introduces the use of depthwise and separable convolutions to construct an EEG-specific network. These convolutional techniques are well-known in computer vision tasks for their ability to capture spatial features efficiently while reducing the number of trainable parameters. By applying these techniques to EEG data, EEGNet encapsulates several well-known EEG feature extraction concepts, including optimal spatial filtering and filter-bank construction, while simultaneously addressing the challenges of limited training data and computational complexity.

#### 1.1.5 CNN-RNN-LSTM (Recurrent Neural Network)

The CRNN consists of four convolutional parts, each similar in structure. Convolution is applied along the time-dimension to capture the highly correlated features from adjacent time steps, as the long length of the time-dimension can pose challenges for the learning ability of the subsequent Recurrent Neural Network (RNN) layers. Between the convolutional parts and the fully-connected layer, three bidirectional Long Short-Term Memory (LSTM) layers are incorporated. To prevent the input of LSTM layers from missing crucial information, some dropout layers in the convolutional parts are removed, while a dropout layer is added after the LSTM layers to prevent overfitting. Given that the dataset is a time series, utilizing an RNN architecture like LSTM is preferred over CNN. This allows the model to effectively gather information along the time-dimension, thereby enhancing its ability to capture temporal dependencies and make accurate predictions. Hybrid architectures like this combine the strengths of both CNNs and recurrent neural networks (RNNs), offering a flexible framework for capturing spatial and temporal information [4].

## 2. Data Preprocessing

### 2.1. Trimming

To visualize the dataset, we categorize samples by label and plot their averaged EEG signals for one channel. As can be observed in Plot 1, the sampled signals in the first half of the range generally display much clearer and more pronounced trends compared to the noisier, flatter signals in the second half. We reason that while removing half of the timesteps might drastically reduce the data set and risk underfitting, the model will indeed benefit from training over a less noisy dataset. We also conducted an experiment to compare model accuracies across time intervals.

### 2.2. Augmentation

#### 2.2.1 Sub-sampling

First, a maxpooling operation is applied across the time dimension, with window size 2 time steps. This operation downsamples the data by taking the maximum value in each non-overlapping window individually across the trimmed dataset, which effectively reduces the number of time bins from 500 to 250.

Second, we apply subsampling that generates multiple versions of the original data by taking every 2nd time bin, starting from indices 0 to 1. Each subsampled version is then stacked to form an augmented dataset.

#### 2.2.2 Averaging and Noise Insertion

We compute the mean across non-overlapping windows of size 2 time steps and add Gaussian noise with a mean of 0.0 and a standard deviation of 0.5 to the averaged data. The resulting data is then concatenated with the previously maxpooled data to augment the dataset before further sub-sampling is applied.

#### 2.2.3 Sequential Cropping

Sequential cropping segments each EEG trial into smaller, sequential, overlapping time windows. This increases the volume of training data by generating multiple data segments from a single trial. We performed cropping with a sample length of 200 and stride of 100.

## 3. Results and Discussion

### 3.1. Overall Results

The test accuracy on a single subject, for all subjects and all subjects over different time durations are shown in the Appendix.

### 3.2. Model Analysis

#### 3.2.1 Single Subject Comparison

Vanilla CNN achieved the highest accuracy when tested on subject 0. Because of the simplicity of the architecture, the model is better able to generalize better by focusing on features that are common across subjects during training.

DeepConvNet achieved the lowest accuracy when tested on subject 0. This is likely due to the deep structure and increased complexity of the model. It is prone to overfitting especially when trained on a limited dataset. Subject 0's EEG patterns may deviate significantly from the training distribution, causing the

DeepConvNet to overfit and perform poorly on this particular subject.

Surprisingly, EEGNet performed slightly worse than Vanilla CNN. This could be due to differences in training data representation. EEGNet's specialized architecture may focus more on extracting EEG-specific features as shown by Lawhern et al, while Vanilla CNN's generic architecture might learn more generalizable features. In single-subject testing, the specific features learned by Vanilla CNN may better align with the patterns present in the individual subject's EEG data, leading to slightly inferior EEGNet performance.

### 3.2.2 All Subject Comparison

EEGNet achieved the highest accuracy when tested across all subjects. EEGNet's architecture is optimized to extract relevant features from EEG signals while minimizing the number of trainable parameters. By leveraging depthwise and separable convolutions, EEGNet can efficiently capture both spatial and temporal dependencies within the data, enabling it to learn meaningful representations that generalize well across different subjects.

DeepConvNet was able to achieve relatively high accuracy of 71.2% because it was capable of capturing more intricate patterns in the training data, especially after data augmentation. Hence with the larger dataset, the complex architecture was well-utilised to generalize across different subjects.

ShallowConvNet and CNN-RNN-LSTM perform the worst. Because of the overly simple model design, ShallowConvNet may struggle to capture the hierarchical features present in EEG data, leading to suboptimal performance. While CNN-RNN-LSTM may offer advantages in capturing both spatial and temporal dependencies in the data, it also introduces additional complexity and potential challenges in training and optimization. If not properly tuned, the intricate interactions between different components of the model could lead to suboptimal performance across subjects.

### 3.2.3 All Subject over Varying Time Period Comparison

Across the time range of 250 to 1000, time = 750 performed the best. For all models, there is an overall increasing accuracy as the time period passes, in which time = 750 is optimal for subject testing accuracy. At the beginning, there is not enough information for the models to capture the training data representation, justifying the lower accuracy. As we increase the time duration, models are able to learn more local features and achieve higher accuracy.

CNN–RNN-LSTM performed the best in terms of classification accuracy. CNN layers in the architecture can efficiently capture local temporal features within short time windows. These features are then passed to the RNN (LSTM) layers, which can capture longer-term temporal dependencies across varying time periods. By combining both CNN and LSTM layers, the model can effectively extract hierarchical features at different temporal scales, enabling robust representation learning across varying time periods.

## 3.3. Hyperparameter Tuning

When training the model, we applied early stopping and learning rate adjustment. Early stopping prevents the model from overfitting and encourage the model to learn generalizable patterns from the training data. This helped to increase test accuracy with early stopping as compared to models without early stopping. Using smaller learning rates resulted in a 3% test accuracy increase as it allowed the model to reach gradient optima without overshooting.

## 3.4. Processing Techniques

Without any data processing, the baseline Vanilla CNN model achieves 66% test accuracy. After processing, (trimming, maxpooling, averaging, and subsampling) the model's test accuracy increased to 71.3%.

Sequential cropping resulted in a high validation accuracy of 94.5%, but low test accuracy of 41.3%. This discrepancy suggests potential overfitting to the training data. Thus, sequential cropping was not included in our final model though we included the visualization of the data in Plot 2. One contributing factor could be the impact of noise and artifacts in EEG data. Given the susceptibility of EEG signals to various noise sources—such as environmental electrical noise, muscle movements, and eye blinks—sequential cropping might accentuate these issues. Particularly, if the test dataset contains pronounced noise characteristics not adequately represented in the training set, this could lead to a degradation in model performance on unseen data.

## References

[1] BCI Competition IV. BCI Competition IV, https://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., & Ball, T. (2017). Deep learning with convolutional neural networks for EEG decoding and visualization. Human Brain Mapping, 38(11), 5391–5420. https://doi.org/10.1002/hbm.23730

[3] Lawhern, V. J., Solon, A. J., Waytowich, N. R., Gordon, S. M., Hung, C. P., & Lance, B. J. (2018). EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces. Journal of Neural Engineering, 15(5), 056013. https://doi.org/10.1088/1741-2552/aace8c

[4] S. Roy, I. Kiral-Kornek, and S. Harrer. Chrononet: A deep recurrent neural network for abnormal eeg identification. CoRR, abs/1802.00308, 2018.

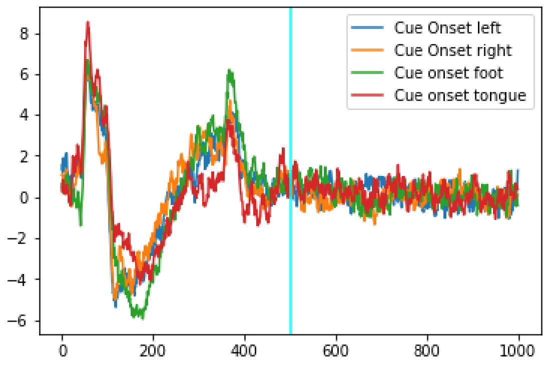
## A. Appendix: Model Performance and Architecture

| Model | Accuracy |
|---|---|
| CNN (without data pre-processing) | 60.0% |
| **CNN** | **70.0%** |
| DeepConvNet | 42.0% |
| ShallowConvNet | 48.0% |
| EEGNet | 50.0% |
| CNN-RNN-LSTM | 50.0% |

Table 1: Test Accuracy of Single Subject (Subject 0)

| Model | Accuracy |
|---|---|
| CNN (without data pre-processing) | 68.6% |
| CNN | 71.3% |
| DeepConvNet | 71.2% |
| ShallowConvNet | 61.0% |
| **EEGNet** | **78.0%** |
| CNN-RNN-LSTM | 61.2% |

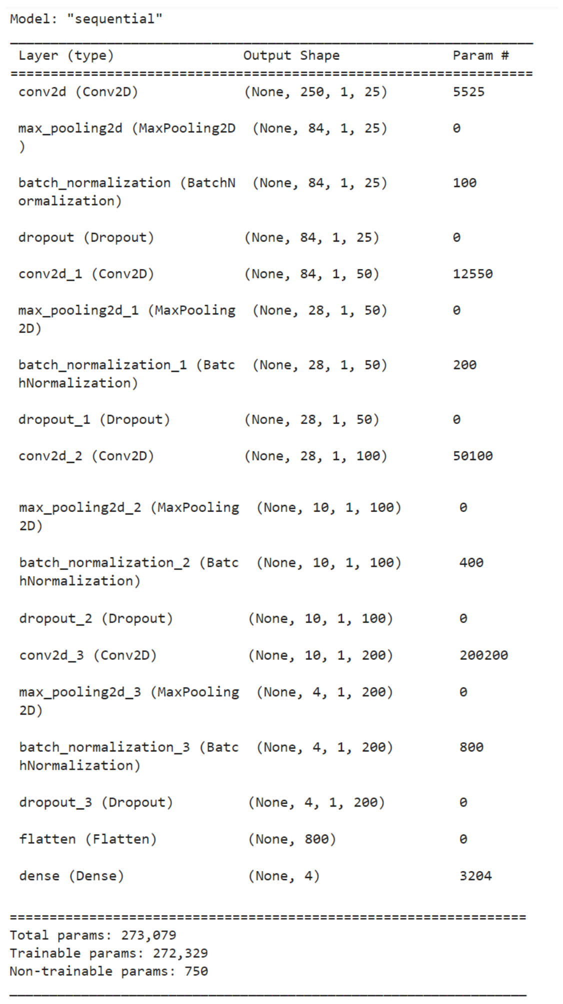
Table 2: Test Accuracy across all subjects



Plot 1: Timestep (x-axis) against average EEG signal (y-axis, potential) for one channel

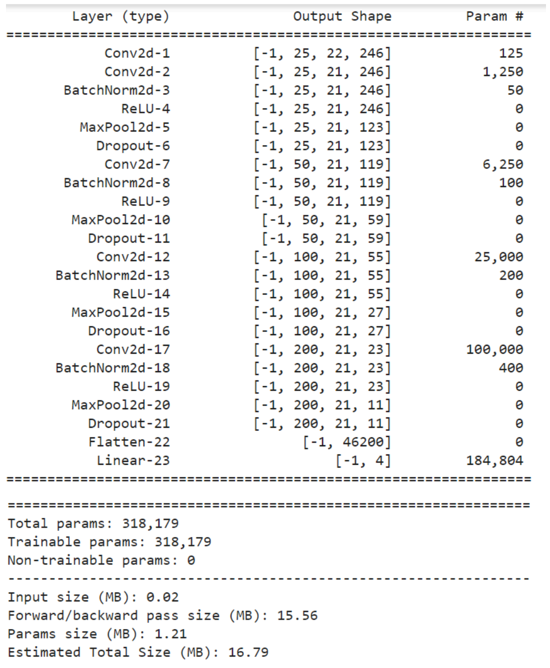
| Model | Accuracy | Time Period |
|---|---|---|
| CNN | 27.1% | 500 |
| DeepConvNet | 37.3% | 1000 |
| ShallowConvNet | 40.7% | 750 |
| EEGNet | 35.6% | 750 |
| **CNN-RNN-LSTM** | **58.0%** | **750** |

Table 3: Test Accuracy over Time Period (Range is from 250 to 1000 with time step 250)



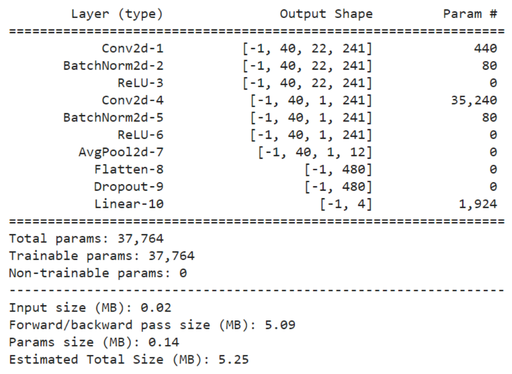
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 250, 1, 25)        5525

 max_pooling2d (MaxPooling2D  (None, 84, 1, 25)        0
 )

 batch_normalization (BatchN  (None, 84, 1, 25)        100
 ormalization)

 dropout (Dropout)           (None, 84, 1, 25)         0

 conv2d_1 (Conv2D)           (None, 84, 1, 50)         12550

 max_pooling2d_1 (MaxPooling  (None, 28, 1, 50)        0
 2D)

 batch_normalization_1 (Batc  (None, 28, 1, 50)        200
 hNormalization)

 dropout_1 (Dropout)         (None, 28, 1, 50)         0

 conv2d_2 (Conv2D)           (None, 28, 1, 100)        50100


 max_pooling2d_2 (MaxPooling  (None, 10, 1, 100)        0
 2D)

 batch_normalization_2 (Batc  (None, 10, 1, 100)        400
 hNormalization)

 dropout_2 (Dropout)         (None, 10, 1, 100)         0

 conv2d_3 (Conv2D)           (None, 10, 1, 200)         200200

 max_pooling2d_3 (MaxPooling  (None, 4, 1, 200)         0
 2D)

 batch_normalization_3 (Batc  (None, 4, 1, 200)         800
 hNormalization)

 dropout_3 (Dropout)         (None, 4, 1, 200)          0

 flatten (Flatten)           (None, 800)                0

 dense (Dense)               (None, 4)                  3204

=================================================================
Total params: 273,079
Trainable params: 272,329
Non-trainable params: 750
_________________________________________________________________
```

Figure 1: Architecture of Vanilla CNN (Loss Function: Cross-entropy. Optimizer: Adam)



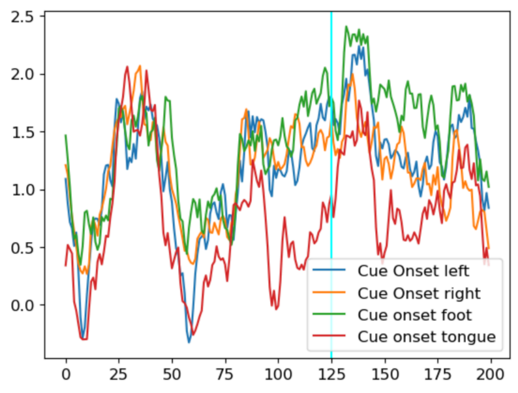
```
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 25, 22, 246]             125
            Conv2d-2          [-1, 25, 21, 246]           1,250
       BatchNorm2d-3          [-1, 25, 21, 246]              50
              ReLU-4          [-1, 25, 21, 246]               0
         MaxPool2d-5          [-1, 25, 21, 123]               0
           Dropout-6          [-1, 25, 21, 123]               0
            Conv2d-7          [-1, 50, 21, 119]           6,250
       BatchNorm2d-8          [-1, 50, 21, 119]             100
              ReLU-9          [-1, 50, 21, 119]               0
        MaxPool2d-10           [-1, 50, 21, 59]               0
          Dropout-11           [-1, 50, 21, 59]               0
           Conv2d-12          [-1, 100, 21, 55]          25,000
      BatchNorm2d-13          [-1, 100, 21, 55]             200
             ReLU-14          [-1, 100, 21, 55]               0
        MaxPool2d-15          [-1, 100, 21, 27]               0
          Dropout-16          [-1, 100, 21, 27]               0
           Conv2d-17          [-1, 200, 21, 23]         100,000
      BatchNorm2d-18          [-1, 200, 21, 23]             400
             ReLU-19          [-1, 200, 21, 23]               0
        MaxPool2d-20          [-1, 200, 21, 11]               0
          Dropout-21          [-1, 200, 21, 11]               0
          Flatten-22                [-1, 46200]               0
           Linear-23                    [-1, 4]         184,804
================================================================
================================================================
Total params: 318,179
Trainable params: 318,179
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.02
Forward/backward pass size (MB): 15.56
Params size (MB): 1.21
Estimated Total Size (MB): 16.79
```

Figure 2: Architecture of DeepConvNet (Loss Function: Cross-entropy. Optimizer: Adam)

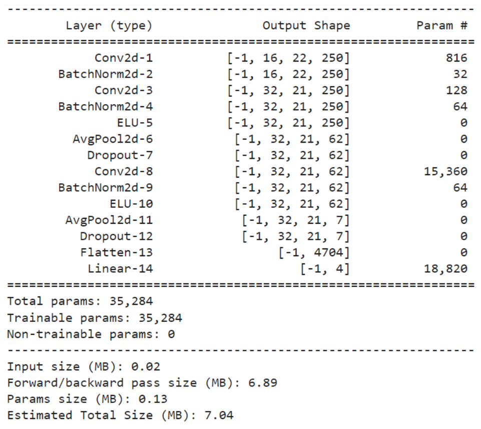


```
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 40, 22, 241]             440
       BatchNorm2d-2          [-1, 40, 22, 241]              80
              ReLU-3          [-1, 40, 22, 241]               0
            Conv2d-4           [-1, 40, 1, 241]          35,240
       BatchNorm2d-5           [-1, 40, 1, 241]              80
              ReLU-6           [-1, 40, 1, 241]               0
         AvgPool2d-7            [-1, 40, 1, 12]               0
           Flatten-8                  [-1, 480]               0
           Dropout-9                  [-1, 480]               0
           Linear-10                    [-1, 4]           1,924
================================================================
Total params: 37,764
Trainable params: 37,764
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.02
Forward/backward pass size (MB): 5.09
Params size (MB): 0.14
Estimated Total Size (MB): 5.25
```

Figure 3: Architecture of ShallowConvNet (Loss Function: Cross-entropy. Optimizer: Adam)

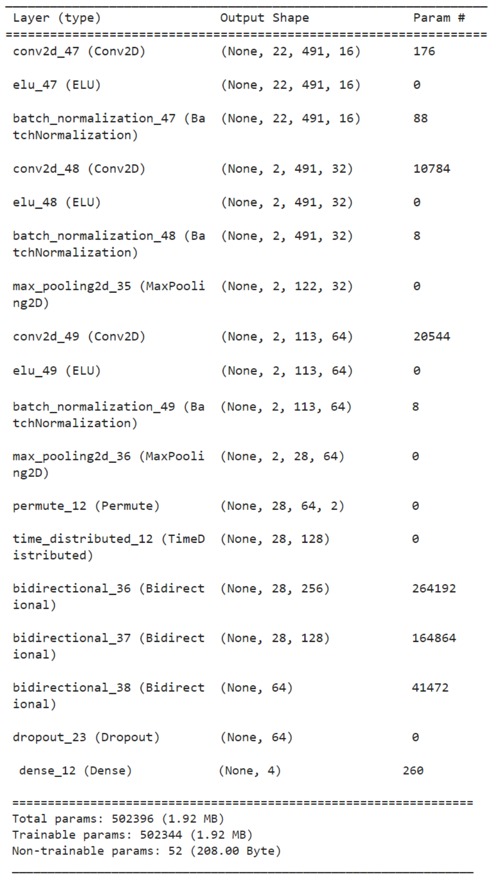


Plot 2: Sequentially-cropped sample of EEG signal



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 16, 22, 250]             816
       BatchNorm2d-2          [-1, 16, 22, 250]              32
            Conv2d-3          [-1, 32, 21, 250]             128
       BatchNorm2d-4          [-1, 32, 21, 250]              64
               ELU-5          [-1, 32, 21, 250]               0
         AvgPool2d-6           [-1, 32, 21, 62]               0
           Dropout-7           [-1, 32, 21, 62]               0
            Conv2d-8           [-1, 32, 21, 62]          15,360
       BatchNorm2d-9           [-1, 32, 21, 62]              64
             ELU-10           [-1, 32, 21, 62]               0
       AvgPool2d-11            [-1, 32, 21, 7]               0
         Dropout-12            [-1, 32, 21, 7]               0
         Flatten-13                 [-1, 4704]               0
          Linear-14                    [-1, 4]          18,820
================================================================
Total params: 35,284
Trainable params: 35,284
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.02
Forward/backward pass size (MB): 6.89
Params size (MB): 0.13
Estimated Total Size (MB): 7.04
```

Figure 4: Architecture of EEGNet (Loss Function: Cross-entropy. Optimizer: Adam)



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_47 (Conv2D)          (None, 22, 491, 16)       176

 elu_47 (ELU)                (None, 22, 491, 16)       0

 batch_normalization_47 (Ba  (None, 22, 491, 16)       88
 tchNormalization)

 conv2d_48 (Conv2D)          (None, 2, 491, 32)        10784

 elu_48 (ELU)                (None, 2, 491, 32)        0

 batch_normalization_48 (Ba  (None, 2, 491, 32)        8
 tchNormalization)

 max_pooling2d_35 (MaxPooli  (None, 2, 122, 32)        0
 ng2D)

 conv2d_49 (Conv2D)          (None, 2, 113, 64)        20544

 elu_49 (ELU)                (None, 2, 113, 64)        0

 batch_normalization_49 (Ba  (None, 2, 113, 64)        8
 tchNormalization)

 max_pooling2d_36 (MaxPooli  (None, 2, 28, 64)         0
 ng2D)

 permute_12 (Permute)        (None, 28, 64, 2)         0

 time_distributed_12 (TimeD  (None, 28, 128)           0
 istributed)

 bidirectional_36 (Bidirect  (None, 28, 256)           264192
 ional)

 bidirectional_37 (Bidirect  (None, 28, 128)           164864
 ional)

 bidirectional_38 (Bidirect  (None, 64)                41472
 ional)

 dropout_23 (Dropout)        (None, 64)                0

  dense_12 (Dense)           (None, 4)                260

=================================================================
Total params: 502396 (1.92 MB)
Trainable params: 502344 (1.92 MB)
Non-trainable params: 52 (208.00 Byte)
_________________________________________________________________
```

Figure 5: Architecture of CNN-RNN-LSTM (Loss Function: Cross-entropy. Optimizer: Adam)