Brain Activity Classification using EEG Signals

Khushbu Pahwa
UID: 805498854
ECE Dept., UCLA
khushbu16pahwa@g.ucla.edu

Kunal Kishore
UID: 505848811
ECE Dept., UCLA
kunalkishore@ucla.edu

Abstract

In this project, we build a Machine Learning model for a Brain-Computer Interface(BCI) system to classify Brain activity from EEG signals. We use the BCI Competition IV 2a dataset [3] recorded from nine subjects who performed four Motor Imagery(MI) tasks. We check the viability of using a model on a subject that was trained on a different set of subjects. We also discuss building individual models optimized for each subject. Three main architectures have been explored and compared in detail, namely Convolutional Neural Networks, a hybrid Convolutional Neural Network - Long Short Time Memory (CNN-LSTM) model, and SkipNet architecture that leverages anchored-STFT and an adversarial data augmentation scheme called 'Fast Gradient Sign Method' (FGSM). We achieved 70% accuracy using the CNN model, 71% accuracy using the CNN-LSTM model, and 76% accuracy using the SkipNet model.

1. Introduction

In this part of the report, we will lay out the motivation for the choice of architectures for the classification of the EEG 2a four class Motor Imagery Dataset in section 1.1, 1.2 and 1.3. We will also discuss the pre-processing and data-augmentation strategies employed in this project for the different architectures that led to performance boost.

1.1. Motivation for architectures and Comparison

1.1.1 CNN

We observe from the exploratory data analysis conducted on the EEG dataset that there is high correlation between adjacent time steps and the EEG features across the 22 channels are also correlated. Since Convolutional Neural Networks are developed from the idea that neighborhood pixels are highly correlated - using the notion of kernels or filters or feature maps, CNN becomes an obvious choice of model architecture to establish the base - line for the Motor Imagery Classification Task [5] as it is a good feature extractor to capture the relevant across time steps and channels. The initial layers attempt to reduce the time dimensions whereas the latter layers attempt to reduce the feature dimensions.

We use a 4 block CNN architecture followed by a dense layer with softmax activation. Each CNN block comprises Conv2D layer with ELU activation , which is followed by MaxPooling2D layer to perform the max pooling operation over the kernels. This is then followed by Batch Normalization which is added to add regularization and to make the model more robust to weight initialization. The last layer of each CNN block is the Dropout layer to prevent the model from overfitting. The architecture is shown in Figure 1 and explained in detail in the Methods Section of the report.

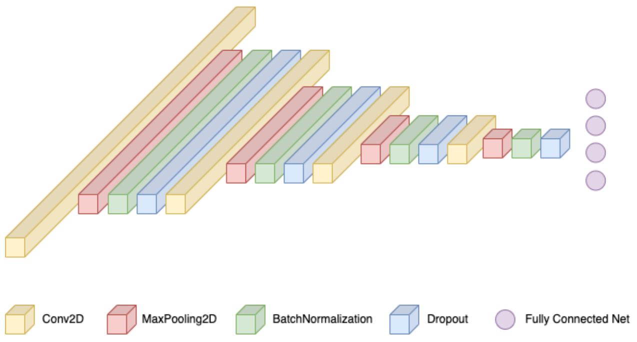

Figure 1. CNN Model Architecture

1.1.2 CNN - LSTM

The motivation for a hybrid CNN - LSTM model is to combine the excellent feature extraction capabilities of the CNN model with the temporal component extraction capabilities of the LSTM model. LSTM is a variant of the Recurrent Neural Network architecture that overcomes the limitations of RNN - namely vanishing gradient.While a CNN alone can also extract information to some extent from a feature set with time series data, the LSTM is far better suited for long temporal sequences.

Our CNN-LSTM architecture consists of the same 4-CNN blocks as used in the above section. We retained the same model configurations for the CNN part of the hybrid architecture because we attained high test accuracy 70% by employing the model architecture used in the previous section. These 4 CNN blocks are then followed by a TimeDistributed(Flatten()) layer and a Dense() layer to make the output of the CNN blocks compatible with the desired input shape of the LSTM layer. This is then followed by the LSTM layer, and finally there is a Dense Layer consisting of 4 neurons (for the 4 tasks / classes) with softmax activation function. For each CNN block in the CNN - LSTM architecture, we tune the number of filters, kernel size and activation function for each Conv2D layer. For the MaxPooling2D layer, we tune the pool size. For the Dropout layer, we tune the keep prob factor or the amount of dropout. For the LSTM block , we tune the no of lstm layers, the no of units in each lstm layer, in addition to optimizing for the dropout and recurrent-dropout in each layer. The architecture is shown in Figure 2 and detailed in the Methods Section of the report.

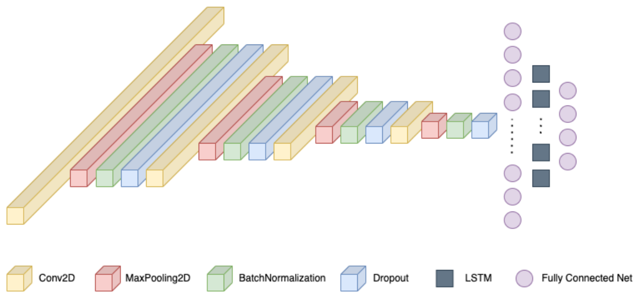

Figure 2. Hybrid CNN-LSTM Model Architecture

1.1.3 SkipNet

While both of the models discussed so far would work well, they don't make use of any domain knowledge about EEG signals or Signal Processing in general at all to make the model simpler. To combat this, we took inspiration from the paper [1] , that performs signal processing on EEG signals to generate spectrograms that can then be fed into CNN like architectures.

The Skip-Net comprises two convolutional layers. The first convolutional layer uses filters that convolve on the time axis and extracts frequency domain features along the time axis. In a similar fashion, the second convolutional layer extracts the time-domain features. Additive skip connection is used to combine the extracted frequency and time domain features to prevent the loss of any information which in turn improves the classification performance of the Skip-Net compared to other classifiers. The proposed model contains significantly less trainable parameters as compared to its counterparts proposed and thus reduces the risk of overfitting. The architecture is shown in Figure 3 and is explained in detail in the Methods Section of the report.

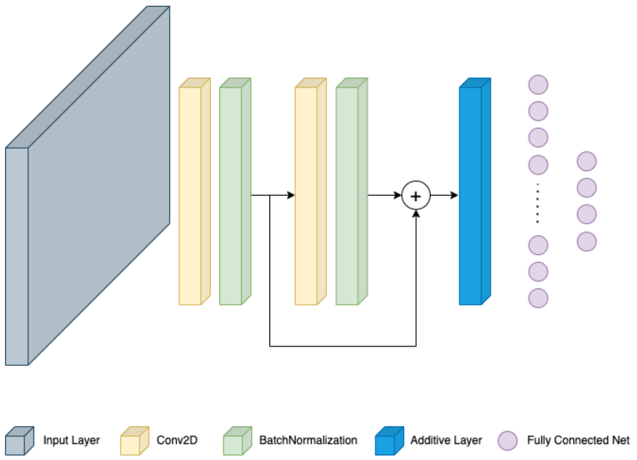

Figure 3. SkipNet Model Architecture

1.2. Preprocessing and Data Augmentation

We observed that most of the variation in the EEG data is constrained in the first 500 time samples. This can be clearly seen by plotting the average EEG signal across all the channels for each of the 4 classes. And this observation was further verified by training and evaluating the CNN model on different trimmed lengths of the samples, varying from 100 to 1000 samples. [6, 2]

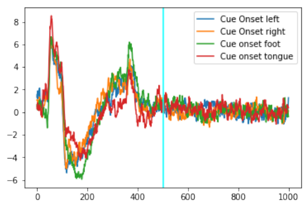

Figure 4. Avg EEG Signals

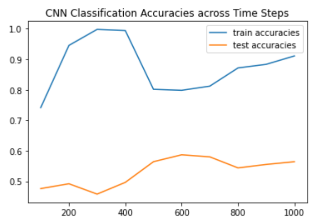

Figure 5. Accuracy vs Trim Size

1.2.1 CNN and CNN - LSTM

The experimentation phase involved rigorous hyperparameter tuning for the different architectures- exploring and implementing different preprocessing strategies like noise addition, smoothing by averaging, sub-sampling across time-axis, max-pooling, filtering, continuous-wavelet transform, and adversarial based augmentation strategies. We use similar pre-processing and data augmentation strategies to evaluate the CNN and CNN-LSTM models.

1.2.2 SkipNet

For the SkipNet architecture, the preprocessing is done using anchored STFT(explained in the model architecture sec-

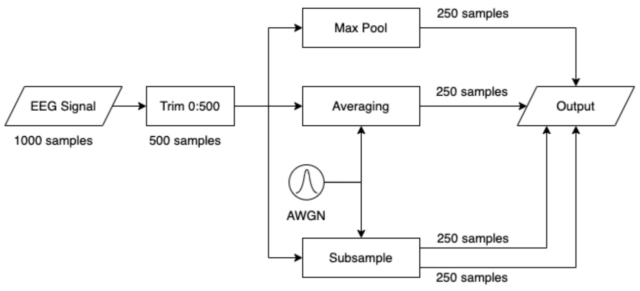

Figure 6. Data Preprocessing for CNN and CNN-LSTM models

tion), and data augmentation is done using adversarial attacks, in order to imitate the state-of-the-art model performance attained by the authors in the paper, for our specific subset 2a of the BCI MI EEG dataset. This SkipNet model architecture, and the associated pre-processing techniques are obtained from the paper [1] , and we attempted to employ the strategies proposed in the paper for the 2a subset of the EEG Motor Imagery Task of BCI competition. The key enablers of this paper are the anchored-STFT and the adversarial and generative model-based data augmentation strategies. Conventional STFT uses a fixed-length window for mapping time domain signal into frequency domain, resulting in a trade-off between temporal and spectral resolution which is critical for feature extraction. Thus, the paper proposed an extension of short time Fourier Transform (STFT) that uses multiple windows of variable sizes for the transformation, called anchored-STFT.The use of anchored-STFT enables better feature extraction. Figures 7 and 8 show the transformation of an EEG signal from a time series to a spectral image using this anchored STFT approach which is then fed to the model for classification.

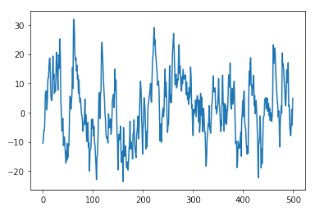

Figure 7. Original EEG Signal

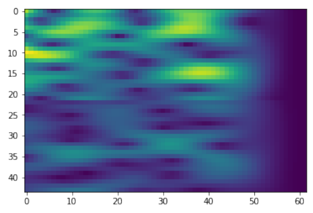

Figure 8. Spectral Image

Secondly, they proposed generative model-based data augmentation methods called Gradient Norm adversarial augmentation (GNAA) and FGSM(Fast Gradient Sign Method) in order to enhance the robustness - since obtaining large, labeled data sets is still a challenge in training deep learning models for BCI applications. These are adversarial attacks, FGSM employs the sign of the gradient weighted by a factor called epsilon, and GNAA the gradient norm. Figure 9 shows an adversarial generated spectral image on a correctly classified training sample, which is then verified to be wrongly predicted by the model. This is then used to re-train the model to achieve better validation and test accuracy.

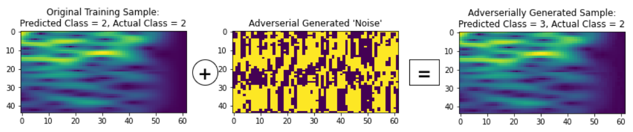

Figure 9. Data Augmentation for SkipNet model

2. Results

SkipNet architecture from the paper [1] outperformed CNN-LSTM and CNN architectures, thus demonstrating that even shallow CNN architecture with few trainable parameters can enhance the classification accuracy.This model's performance is followed by CNN-LSTM architecture which does well on all subjects models but does not generalize well for individual subjects. The CNN model also attains high accuracy but again suffers from overfitting and has poor generalization on datasets with different data distributions.

Model	Test Accuracy
CNN	70.03%
CNN+LSTM	71.11%
SkipNet	75.95%

3. Discussion

With preprocessing and data augmentation pipeline as demonstrated in Fig 6, both CNN and CNN-LSTM achieved significant performance boost.

For SkipNet architecture, we observed that enhancement in the decoding capabilities of the EEG signals is brought about by its novel preprocessing and data augmentation techniques involving the use of anchored STFTs and the adversarial data augmentation achieved through FGSM and GNAA attacks.This technique works well for different data distributions, as is evident from Table IV (in Methods).

We inferred that even a shallow CNN architecture with low computational overhead as used in SkipNet can attain high performance, while eliminating the problem of overfitting.

We inferred that the model optimized for all subjects when re-trained and evaluated on individual subjects, did not generalize well.

When we optimized and trained the model for subject 1, and evaluated for other subjects, we did not attain the same performance level on all subjects, thus, we seek to develop subject-independent / caliberation-free models using technqiues like common spatial pattern (CSP), common spatiospectral pattern (CSSP), filter bank CSP (FBCSP), and Bayesian spatio-spectral filter optimization (BSSFO).[4].

References

[1] O. Ali, M. Saif-ur Rehman, S. Dyck, T. Glasmachers, I. Iossifidis, and C. Klaes. Enhancing the decoding accuracy of eeg signals by the introduction of anchored-stft and adversarial data augmentation method. *Scientific Reports*, 12(1):1–19, 2022. 2, 3

[2] J.-S. Bang and S.-W. Lee. Motor imagery classification based on cnn-gru network with spatio-temporal feature representation. *arXiv preprint arXiv:2107.07062*, 2021. 2

[3] C. Brunner, R. Leeb, G. Müller-Putz, A. Schlögl, and G. Pfurtscheller. Bci competition 2008–graz data set a. *Institute for Knowledge Discovery (Laboratory of Brain-Computer Interfaces), Graz University of Technology*, 16:1–6, 2008. 1

[4] O.-Y. Kwon, M.-H. Lee, C. Guan, and S.-W. Lee. Subject-independent brain–computer interfaces based on deep convolutional neural networks. *IEEE transactions on neural networks and learning systems*, 31(10):3839–3852, 2019. 3

[5] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung, and B. J. Lance. Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces. *Journal of neural engineering*, 15(5):056013, 2018. 1

[6] Y. Pei, Z. Luo, Y. Yan, H. Yan, J. Jiang, W. Li, L. Xie, and E. Yin. Data augmentation: Using channel-level recombination to improve classification performance for motor imagery eeg. *Frontiers in Human Neuroscience*, 15:113, 2021. 2

Methods and Supplementary Materials

I. Model Architectures and Experiments

In this section, we describe the exact architecture used for each of the models along with the intuition and reasoning behind picking each of the hyperparameters in the model.

A. CNN

The first model we used is a Convolutional Neural Network with 4 convolutional layers stacked before a single dense layer for the final classification task. We added MaxPooling after each convolution along with BatchNormalization and Dropout for regularization. The model takes input of shape $(N_t \times 1 \times N_c)$ where N_t is the length of a single EEG signal, and N_c is the number of channels/electrodes. Its parameters for each layer are described in Table I.

Idx	Layer Name	Num Filters	Size	Padding	Activation
1	Convolutional Layer	25	10 x 1	'same'	ELU
2	MaxPool Layer	-	3 x 1	'same'	-
3	Batch Normalization				
4	Dropout (0.5)				
5	Convolutional Layer	50	10 x 1	'same'	ELU
6	MaxPool Layer	-	3 x 1	'same'	-
7	Batch Normalization				
8	Dropout (0.5)				
9	Convolutional Layer	100	10 x 1	'same'	ELU
10	MaxPool Layer	-	3 x 1	'same'	-
11	Batch Normalization				
12	Dropout (0.5)				
13	Convolutional Layer	200	10 x 1	'same'	ELU
14	MaxPool Layer	-	3 x 1	'same'	-
15	Batch Normalization				
16	Dropout (0.5)				
17	Flatten				
18	Dense with 4 Neurons				Softmax

TABLE I: CNN Model Architecture

1) Experiments and Hyperparameter Tuning::

a) Activation: We used the ELU activation in all the convolutional layers because it has all the advantages of a RELU unit but is faster than the RELU in convergence (despite it being slower to compute ELU). This gives a better generalization using the ELU and given that we have limited dataset which makes us prone to overfit, ELU was a better choice for activations.

b) Learning Rate: We tried different fixed learning rates between 10^{-1} to 10^{-4} with different multiplication factors and found that 10^{-3} worked the best.

c) Depth of the CNN (number of stacked CNN layers): We experimented with various stacked convolutional layers ranging from 1 to 6 layers and found that 4 layers gave the best validation performance than others. Lesser layers than this gave worse training accuracy due to underfitting while more than 4 showed worse validation accuracy due to overfitting.

d) Regularization: We chose a Dropout of 0.5 to reduce the amount of overfitting. We found that a lower value than 0.4 showed worse generalization performance while values more than 0.6 caused underfitting.

B. CNN - LSTM

To better utilize the long term dependency inside the EEG signals, we wanted to add a Recurrent Neural Network which are best suited for handling a time series data. We experimented with using two forms of RNN – LSTM and GRU. Since we already had an optimized model architecture of CNN, we proceeded with adding the RNN just after the CNN layers. Both LSTM and GRU showed similar initial results, so we went ahead with optimizing just the LSTM for this report. We used 20 LSTM units after the CNN layers, and introduced a fully connected dense layer in between. This FC Dense layer helped learn a better linear mapping between the feature maps generated by the CNN layers and the LSTM modules. The architecture of this CNN-LSTM model is described in Table II.

Idx	Layer - Parameters
1	CNN layers (same as in Model 1)
2	Flatten - Dense Layer with 100 neurons - Activation: None(Linear)
3	Reshape - LSTM with 20 units - Dropout 0.6 and Recurrent Dropout 0.1
4	Dense with 4 neurons - Activation: Softmax

TABLE II: CNN-LSTM Model Architecture

1) Experiments and Hyperparameter Tuning::

a) Learning Rate: We tried different fixed learning rates between 10^{-1} to 10^{-4} with different multiplication factors and found that $5 * 10^{-4}$ worked the best.

b) Dense Layer with 100 neurons: We saw that having this fully connected layer between the CNN and LSTM showed better train and validation performance. We tried having multiple such layers and different numbers of neurons with different activations, but anything other than a simple dense layer led to overfitting and worse generalization.

c) LSTM: We started with 10 units of LSTM and were already able to see a minor improvement in performance over the CNN-only model. We saw an even better improvement in performance when we increased to 20 units of LSTM. We experimented with more stacked layers of LSTMs and even Bidirectional LSTMs, but they were all leading to more overfitting, which seemed more and more difficult to regularize.

d) Regularization: In this model, we observed that removing the Dropout after some of the convolutional layers sometimes gave a better performance. The intuition behind this could be that more information can be passed onto the LSTM layer with lesser dropout.

e) Optimizer: We tried two different optimizers: Adam and RMSProp with different configurations and found Adam to be marginally faster and more stable in attaining convergence.

C. SkipNet

In this model, we perform a preprocessing to each EEG signal (trimmed to the first 500 samples) using an anchored STFT approach. We perform STFT using 5 window lengths - [16, 32, 64, 128, 256] and a fixed stride of 8 to get 5 spectral images for each signal. We then extract the spectrum data from 2 frequency bands (which have been discussed in the Reference paper of this model to be containing the most information) namely *mu* band (4-15Hz) and *beta* band (19-30Hz). We then concatenate images generated from different electrodes with the same anchor length into a single image. Thus, we end up with 5 image samples per subject's EEG recordings from all the electrodes. We apply max voting on this batch of 5 samples to decide the final estimated class during test time. These train samples are then fed to the model for training, after which we generate adversarial samples that look visually similar to the correctly classified training samples but is wrongly classified by the existing model. The visualization of this step is shown in the main report where an adversarial image is shown to be the additive sum of a training image and generated noise. The model is retrained using these newly generated samples which is found to further improve the performance by at least 1% additionally. The architecture of the SkipNet model is shown in Table III.

Idx	Layer - Parameters				
0	Spectrogram Generation from EEG Signal				
1	Convolutional Layer	16 filters	Size: 22*44 x 1	'Valid' padding	RELU act.
2	Batch Normalization				
3	Convolutional Layer	16 filters	Size: 22*44 x 1	'Valid' padding	RELU act.
4	Batch Normalization				
5	Additive Layer : CNN1 (layer 2) activation + CNN2 (layer) 4 activation				
6	Fully Connected Dense Layer - 128 neurons - RELU activation				
7	Dense Layer with 4 neurons - Softmax activation				

TABLE III: Skipnet Model Architecture

II. Performance of each Model on each Dataset

A. Different Models trained and evaluated on all subjects

We saw the best performance from the SkipNet model which gave a test accuracy of 75.95%. We noticed that both LSTM and GRU initially gave similar performance trends, so we optimized just the LSTM model's hyperparameters which is why it showed much better performance than the CNN+GRU model in the Table below. We also observed that using a bidirectional LSTM caused overfitting which proved difficult to regularize. The accuracies obtained for each model are listed in Table IV.

Model	**Test Accuracy (after max voting)**
CNN only	70.03%
CNN + LSTM	71.11%
Skipnet	75.95%
CNN + GRU	65.46%
CNN + Bidirectional LSTM	65.24%

TABLE IV: Different Models Trained and Evaluated on All Subjects

B. Best 2 Models trained on all subjects, evaluated on each individual subject

We used the best 2 models - the CNN+LSTM model and the SkipNet model - pretrained on all the subjects combined and observed their performance on each subject individually. The accuracies seem to be distributed across each subject evenly which means that both the models perform well on each of the subjects. The accuracies are listed in Table V.

Model	**Sub 1**	**Sub 2**	**Sub 3**	**Sub 4**	**Sub 5**	**Sub 6**	**Sub 7**	**Sub 8**	**Sub 9**
CNN-LSTM	67.50%	64.99%	75.99%	72.50%	76.06%	65.31%	77.50%	70.50%	69.68%
SkipNet	72.1%	70.56%	79.31%	80.5%	77.28%	69.31%	83.06%	77.25%	72.37%

TABLE V: Best 2 Models Trained on All Subjects and Evaluated on Individual Subjects

C. Best 2 Models (optimized on all subjects together) trained and evaluated on each individual subject

We then trained the CNN+LSTM model and the SkipNet model - on each subject individually and evaluated them on that same subject to see how well the same model would perform if it was trained on one subject only and evaluated on the same subject. It can be observed from the results in Table VI that the CNN-LSTM model struggled to perform well with most of the subjects, while the SkipNet model is able to train efficiently and show better performance on each subject. This shows that the SkipNet model generalizes very well without needing to individually optimize it for any subject. This could be intuitively reasoned with the simplicity of the model as well as the augmented sample generation of this model that makes SkipNet learn efficiently even for lesser number of training samples. This could mean that the preprocessing (spectral image using STFT) done on the EEG signals in SkipNet helps the model become calibration free and subject agnostic.

Model	**Sub 1**	**Sub 2**	**Sub 3**	**Sub 4**	**Sub 5**	**Sub 6**	**Sub 7**	**Sub 8**	**Sub 9**
CNN-LSTM	50.0%	46.0%	42.0%	56.0%	76.61%	40.81%	70.0%	42.0%	68.08%
SkipNet	67.3%	70.18%	67.55%	62%	69.4%	63.33%	72.97%	68.81%	61.32%

TABLE VI: Best 2 Models Trained and Evaluated on Individual Subjects

D. CNN - LSTM Model (optimized on subject 1) trained and evaluated on each individual subject

Since the CNN+LSTM model that was optimized on all the subjects together did not generalize well when we trained and tested it on individual subjects, we had to optimize the CNN+LSTM model further individually for subject 1, and re-trained and evaluated the model on each individual subject. The accuracies are listed in Table VII. While optimizing for subject 1,

Model	**Sub 1**	**Sub 2**	**Sub 3**	**Sub 4**	**Sub 5**	**Sub 6**	**Sub 7**	**Sub 8**	**Sub 9**
CNN-LSTM	72.11%	48.76%	51.23%	59.9%	74.1%	42.06%	70.64%	43.71%	71.31%

TABLE VII: CNN-LSTM Model Optimized for Subject 1, Trained and Evaluated on Individual Subjects

we had to modify the architecture in terms of some of the modules and their hyperparameters. In the modified architecture, the number of filters in the CNN layers was changed to 16, 32, 64 and 128. The padding was changed from 'same' to 'valid' and an L2 regularixation of 0.01 was added to each layer. The kernel size of the second layer was changed to 21×1. Dropout was removed from all the layers and only added as 0.4 at the last layer. The single LSTM layer was changed to stacked bidirectional LSTM layers with 128, 64 and 32 LSTM units.

It can be seen that the CNN-LSTM model that was optimized on subject 1 performed well on that subject, but could not show consistant performance on other subjects using the same model. This shows that the CNN-LSTM model will need to be optimized or trained for every new subject. This reason, along with the overall better accuracy of the SkipNet model makes the SkipNet a better choice for EEG signals classification task.