Decoding of Electroencephalography (EEG) Signals using Neural Networks

Rushi Bhatt
University of California, Los Angeles
rushibhatt@g.ucla.edu

Shivani Ganti
University of California, Los Angeles
sganti@g.ucla.edu

Ronak Kaoshik
University of California, Los Angeles
ronak42@g.ucla.edu

Subhiksha Mani
University of California, Los Angeles
subhiksha@g.ucla.edu

Abstract

This report aims to provide insights on Brain Computer Interaction (BCI) Competition's electroencephalography (EEG) data [1] by employing various deep learning techniques and models. Using this temporal EEG data, the classification task is to predict one of four actions performed by a subject.

The models explored include Convolutional Neural Networks (CNN), Long Short-Term Memory Networks (LSTM), Recurrent Neural Networks (RNN), and ensemble approaches combining select models. These models were further iterated upon with pre-processing steps such as adding random noise, stratified sampling, and data augmentation.

This paper will target a performance comparison for each of these models, while also providing additional experiments to better understand performance across all subjects and various time periods.

1. Introduction

This report constructs a CNN, RNN, LSTM, CNN+LSTM, CNN+RNN, with an emphasis on CNN and CNN+LSTM as they proved to be the two best models after initial experimentation. CNN was the first model that was experimented because it is known to be good at classifying temporal data.

1.1. CNN

The proposed CNN model consists of three convolutional blocks with each block consisting of a 2-dimensional convolutional layer, and a 2-D max pooling layer. Following each convolution layer, a dropout of 0.5 as well as batch normalization layer is added to avoid overfitting.The Exponential Linear Unit(ELU) was used as the activation for the 2-D convolutional layers. The three blocks are followed by an output fully-connected layer using softmax activation. This was the best performing model. Reference the detailed architecture diagram in **Figure 4 and 5**.

1.2. RNN

The RNN model uses 3 GRU layers with a dropout of 0.5 and relu activation.It is followed by a batch normalization and dropout layer to reduce overfitting. The final output layer using the softmax activation.

1.3. LSTM

The Basic LSTM model makes use of two LSTM layers using the elu activation followed by batch normalization layers. The final connected layer makes use of the softmax function.

1.4. CNN + LSTM

The CNN+LSTM model was developed to have the CNN layers used to extract features from the input data and the LSTM layers are included to help with sequence prediction. The hybrid model consists of 3 convolutional blocks with each block consisting of a 2-dimensional convolutional layer, and a 2-D max pooling layer. Two LSTM layers with a dropout of 0.6 are also added. The output layer still makes use of the softmax activation.

1.5. CNN + RNN

The CNN+RNN model makes use of two CNN blocks, with each block including a 2-D convolution layer, a 2-D max pooling layer, batch normalization layer and dropout layer. It is followed by two RNN layers and an output layer.

2. Results

The best performing model was the 3-layer CNN with a best test accuracy of 75.76%. Reference the confusion matrix for this model in **Figure 6**.

2.1. Model Comparisons

Table 1 displays the best test accuracy for all 5 models.

2.2. Accuracy vs. Preprocessing Techniques

Table 2 displays the test accuracy for models without any preprocessing, with preprocessing and random sampling, and with preprocessing and stratification. These are outputted for the top two models: CNN and CNN+LSTM. Graphs for the CNN training and validation accuracy can be found in **Figure 1** and **Figure 2**, respectively representing with and without stratification preprocessing.

2.3. Accuracy vs. Subject Comparison

The test accuracy for each subject using CNN and CNN+LSTM models can be found on **Table 3**.

2.4. Accuracy vs. Time Period Comparison

Table 4 displays the test accuracy over various time samples for CNN and CNN+LSTM models.

3. Discussion

3.1. Tuning Hyperparameters

The parameters that were tuned during the experiments were batch size, learning rate, number of epochs, number of layers, filter size, kernel size, type of activation layers and type of optimizer.

After numerous experiments, incremental filter sizes for each of the layers proved to be more efficient than repeating the same filter size as data varies from layer to layer. Increasing epochs also allows for a longer time for the model to learn and accurately train the data. Accuracy of some models began stagnating or decreasing after experimenting with too large of epochs, so a nominal value of 50 was chosen as it was also faster to run. Increasing the batch size to 200 allowed for more samples to be worked through and a higher accuracy rate. Despite starting out with a four-layer model, a three-layer CNN model was more promising. Since the model makes use of batch normalization, it allowed for higher learning rates to be implemented and for the accuracy to converge faster.

3.2. Pre-processing

After conducting multiple experiments it is observed that using pre-processing techniques like trimming, max pooling, averaging, and adding noise gave a significant improvement in the model performance. As can be seen from the results stated in **Table 2**, CNNs have an improvement in test accuracy from 55% to 75.73% and similarly for CNN+LSTMs. A plot of the 22 channel-EEG signal from the raw dataset showcase that the majority of the information is stored in the initial half of the time-series sample and the later half consists of noise. So the very first preprocessing step is trimming half of the series. It is followed by max pooling operation in a temporal sense which leads to further halving of the series. Similarly, another series is obtained using averaging in a temporal sense and adding white noise. These two series are concatenated to form an even larger training set. A final step involves sub-sampling to further augment the dataset. The operations of max-pooling, averaging and sub-sampling helped to preserve significant information and this is ascertained from the results. The deep-networks are able to learn well from such a well pre-processed and augmented dataset as opposed to the raw dataset.

3.3. Comparison Across Subjects

This experiment compares accuracy using two different preprocessing techniques across all nine subjects. Results can be found in **Table 3**.

The first preprocessing technique randomly shuffles the data, where the validation set is specified to be of size 1500 for the entire dataset. This same ratio of approximately 18 percent is then applied to each subject's partitioned validation set for consistency.

The second preprocessing technique uses stratified sampling, the data is now confined to that of each subject's. The stratification is done by creating equal splits across all 36 labels (4 classes and 9 subjects) and then including them in train-validation sets by splitting them using the aforementioned ratio of 18 percent.

Regardless of the choice of model or preprocessing approach, some subjects consistently seem to perform either extremely well or poorly. For instance, subjects 0, 4, and 8 routinely score amongst the highest of all subjects, whereas subjects 1 and 5 regularly score amongst the lowest. The results perhaps suggest that the variability amongst subjects remains the same across CNN-based models with similar sampling techniques.

3.4. Comparison Across Time Periods

In this experiment, we focus on exploring how many time samples are required to get a required to get a reasonable classification accuracy. All the tests are performed on preprocessed EEG data, which consists of 250 time samples for each EEG channel for each subject. Also, the experiments were conducted for both random sampling as well as stratified sampling.

From the results in **Table 4 and Figure 3** , we observe that for the proposed 3-layer CNN model, obtains a fairly high accuracy only with 125 time samples, which is about 50 % of the entire time sequence. For the proposed CNN + LSTM model, the accuracy saturates after 150 time samples, which is 60 % of the entire time sequence.

Moreover, from the experiments with and without stratification, we observe that the stratification test accuracy almost always outperforms the random selection accuracy, regardless of the model. From this, we can further conclude that equally stratifying data (over 36 classes), and then preparing train/ validation splits boosts the overall classification performance.

3.5. Comparison Across All Models

The four models experimented with are CNN, CNN+LSTM, CNN+RNN, LSTM and RNN.

Out of these, the 3-layer CNN model performed the best with an accuracy of 75.61 % while LSTM performed the worst with an accuracy of 25.06 %.The basic models of CNN and RNN performed poorly, but through observation, the CNN model provided a larger accuracy between the two. Therefore, the hybrid models of CNN+LSTM and CNN+RNN were explored since LSTMs and RNNs help with sequence detection. The ensemble approaches featuring CNNs seem to be much more accurate than their basic model counterparts (LSTM, RNN) with accuracies of 70.9 % and 63.4 %.

Since CNNs are faster and easier to train, most of the focus was towards improving the basic CNN and CNN hybrid models. Basic RNNs have the problem of exploding and vanishing gradients and thus aren't as stable and performed poorly despite adding batch normalization and dropout. Meanwhile, LSTMs are able to maintain information in memory for long periods of time. The better performance in LSTMs over RNNs in the hybrid models can be attributed to this fact.

References

[1] Organizers of BCI Competition IV (2008) *https://www.bbci.de/competition/iv/*, BCI Competition IV.

[2] Robin Tibor Schirrmeister et al (2018) *https://arxiv.org/pdf/1703.05051.pdf*, Deep learning with convolutional neural networks for brain mapping and decoding of movement-related information from the human EEG.

4. Tables/ Figures

Model	Best Test Accuracy
CNN	75.76 %
RNN/ GRU	39.48 %
LSTM	25.06 %
CNN + RNN	64.9 %
CNN + LSTM	70.08%

Table 1. Model comparison

Pre-processing	Train/ Val split	CNN	CNN + LSTM
Without Preproc.	Random Sampling	55 %	49 %
With Preproc.	Random Sampling	72 %	70.8 %
	Stratification	75.73 %	72.12 %

Table 2. Accuracy with v/s without Preprocessing Techniques

Subject	CNN		CNN + LSTM	
	Random	**Stratify**	**Random**	**Stratify**
0	70%	63.5%	63.5%	55%
1	49.5%	52%	49.5%	46.5%
2	68%	68%	62%	70%
3	73,5%	65%	64.4%	59%
4	78%	80%	75%	72%
5	48%	56%	39%	57%
6	72.5%	68.5%	50%	56%
7	61%	63%	54.5%	53.5%
8	73%	84%	73%	79%

Table 3. Accuracy vs. Subject Comparison

Time Samples	CNN		CNN+LSTM	
	Random	**Stratify**	**Random**	**Stratify**
25	38.6 %	40.6 %	38.4 %	41.4 %
50	53.7 %	54.6 %	50.1 %	51.1 %
75	58.4 %	58.6 %	54.7 %	54.1 %
100	65.6 %	62.9 %	57.9 %	59.7 %
125	70.0 %	69.9 %	60.1 %	63.0 %
150	70.2 %	72.6 %	65.7 %	65.8 %
175	68.4 %	67.9 %	65.1 %	66.6 %
200	69.8 %	70.7 %	66.2 %	64.7 %
225	68.3 %	69.2 %	64.6 %	64.7 %
250	70.6 %	70.7 %	65.9 %	65.8 %

Table 4. Accuracy vs. Time Period Comparison

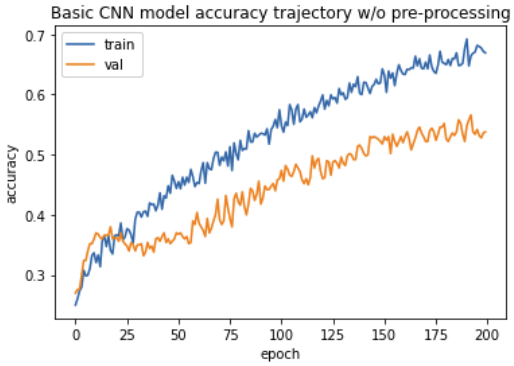

Figure 1. CNN model accuracy without preprocessing

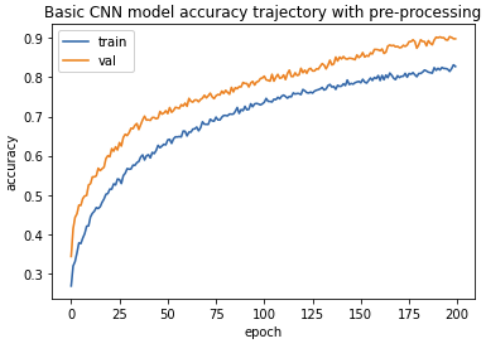

Figure 2. CNN model accuracy with preprocessing

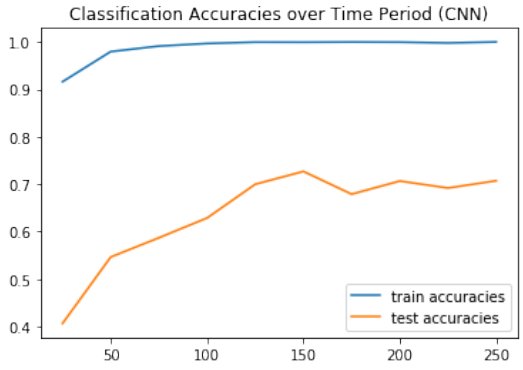

Figure 3. Accuracies over time samples for CNN with stratification

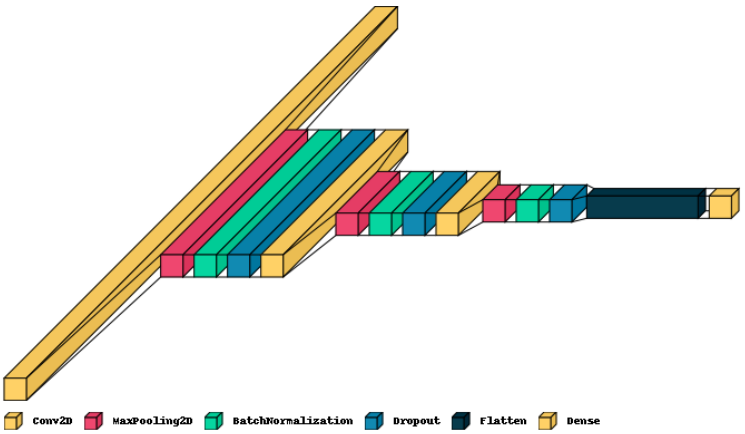

Figure 4. Architecture Overview of 3 layer CNN

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 250, 1, 25)	5525
max_pooling2d_3 (MaxPooling 2D)	(None, 84, 1, 25)	0
batch_normalization_3 (Batc hNormalization)	(None, 84, 1, 25)	100
dropout_3 (Dropout)	(None, 84, 1, 25)	0
conv2d_4 (Conv2D)	(None, 84, 1, 50)	12550
max_pooling2d_4 (MaxPooling 2D)	(None, 28, 1, 50)	0
batch_normalization_4 (Batc hNormalization)	(None, 28, 1, 50)	200
dropout_4 (Dropout)	(None, 28, 1, 50)	0
conv2d_5 (Conv2D)	(None, 28, 1, 100)	50100
max_pooling2d_5 (MaxPooling 2D)	(None, 10, 1, 100)	0
batch_normalization_5 (Batc hNormalization)	(None, 10, 1, 100)	400
dropout_5 (Dropout)	(None, 10, 1, 100)	0
flatten_1 (Flatten)	(None, 1000)	0
dense_1 (Dense)	(None, 4)	4004

Total params: 72,879
Trainable params: 72,529
Non-trainable params: 350

Figure 5. Details of CNN Model Architecture

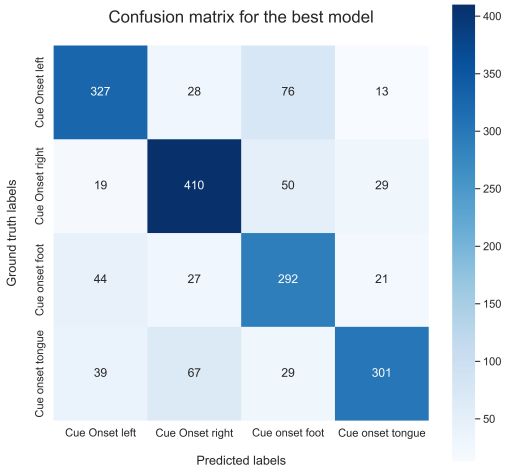

Figure 6. Confusion Matrix for 3-layer CNN