Evaluating Different Neural Network Structures for EEG Predictions

Chushu Shen
UCLA
chushu@g.ucla.edu

Zitian Wang
UCLA
zitiansunsh1ne@ucla.edu

Abstract

An electroencephalogram (EEG) dataset, containing around 3000 trails from 9 subjects, was adopted for evaluating the difference between various deep learning models. Several models, including CNN, CNN+LSTM, CNN+Transformer, were trained on the dataset and comprehensively compared. The impact of training with single subject and specific time bins were also extensively examined. Overall, CNN with LSTM and Transformer structures demonstrated better performance than CNN alone.

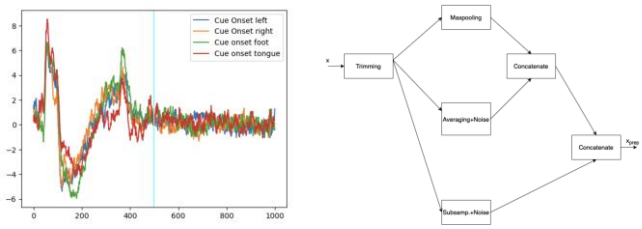
1. Introduction

1.1. Data Overview and Preprocessing

The whole EEG data was collected from 9 subjects. By observing the general trend of the training data, larger variations across four classes were observed between 0-500 time bins than that of 500-1000 time bins. Trimming was therefore applied to the dataset to obtain only the data from 0-500 time bins. To further effectively reduce the data dimensions and improve generalizability, three separate parallel operations were executed to each input sequence. First, maxpooling with stride=2 was applied to each input sequence, reducing the time bin dimensions from 500 to 250. Second, the value of the input sequence was averaged and the random noise was added to each input sequence. The resulting 4330 trails from maxpooling and averaging+noise (each operation produces a sequence with length = 2115) were then concatenated together. On the other hand, subsampling was applied to the whole dataset with adding random noise, resulting in another 4330 trails. Ultimately, the two 4330 trails were concatenated together, which served as the product from data preprocessing [1].

1.2. Model overview

Convolutional Neural Network (CNN) Inspired by how retina cells work in neurological systems, convolutional neural network (CNN), a neural network architecture that specializes in dealing with multi-dimensional arrays, has been widely applied in areas such as computer vision. CNN contains convolution layers which utilize kernel filters to process the input as its original tensor form. For each filter in one convolution layer, convolution operation will be conducted between the filter and the input tensor with the pre-specified stride and padding. A convolution layer is always followed by a pooling layer, which extracts useful information from the convolution output through operations such as Max (MaxPooling) and Average (Average Pooling). A deeper CNN with more hidden convolution layers can have large receptive fields, which always achieves superior performance than the Fully Connected Network because of less parameters by parameter sharing. In the current project, CNN was applied as the baseline structure for performance evaluation on the input EEG signal data.



(a) (b)

Figure 1: (a) Visualization of EEG data set. (b) Workflow of data preprocessing.

Recurrent Neural Network (RNN) As EEG signals are time-series data, Recurrent Neural Network (RNN) structure is considered as a potential option to perform classification tasks. LSTM can tackle the problem of gradient vanishing and explosion, and can store information over long-term temporal dependency [2].

Transformer Transformer, which contains self-attention layers of encoders and decoders, have been broadly applied for dealing with sequential data in areas such as natural language processing [3]. Specifically, self-attention layers are designed to capture the relative weight of a position in the overall sequence. In a self-attention layer, one input sequence will initially be separated into several pieces, which are called multi-heads. Three sets of parameter matrices are used for calculating the relative weights of each input parts through adding a series of dot products between the parameter matrices and different pieces. A Transformer Encoder contains a self-attention layer and two fully connected layers [3, 4].

1.3. CNN+LSTM

For the CNN+LSTM model, number of convolutional layers is fixed at 4, followed by 2 or 3 LSTM layers and a fully-connected output layer. For LSTM layer, hyperparameters such as dropout rate and layer units number were optimized using 5-fold cross validation [2].

1.4. CNN+Transformer

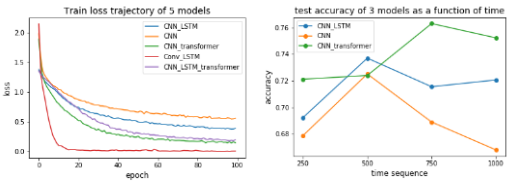
In this project, a TransformerEncoder + CNN structure was experimented for application in EEG data [3, 4].

1.5. TransformerEncoder + CNN + LSTM

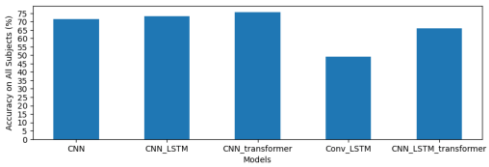
Given the robustness of TransformerEncoder, CNN, and LSTM, we further proposed a neural network architecture, T-CRNN, that contains all three structures [3, 4].

1.6. ConvLSTM

A 3-layer convolutional LSTM followed by 2 dense layers was also experimented. Convolutional LSTM uses convolution for input-to-state and state-to-state connections, as opposed to dense connections in LSTM [5]. Due to the limitation of GPU on Google Colab, no hyperparameter tuning was conducted for the T-CRNN and ConvLSTM structures.



(a) (b)



(c)

Figure 2: (a) Train loss of all models on all training subjects. (b) Test accuracy of CNN, CNN+LSTM and CNN+Transformer as a function of time. (c) Test accuracy of all models.

2. Results

2.1. Hyperparameter Tuning

For our CNN+LSTM and CNN+Transformer structures, 5-fold cross validation was performed to find the optimal parameters on the train_valid dataset. Specifically, the train_valid dataset was first randomly shuffled and then equally split into 5 folds. During each cross-validation iteration, one fold was used for validation while all the other four folds were used for training the model with the specified hyperparameters, and the accuracy of the trained model was calculated based on the validation set. The cross validation process iteratively repeated 5 times until all 5 folds had been used as the validation set, which generates five accuracy values. The performance of the model with the specified hyperparameters was evaluated based on the average of all the five accuracies. The model with the best performance was selected for further evaluation.

To determine the optimal CNN+LSTM architecture, CNN+LSTM models with different drop-out rate, units and number of LSTM layers were evaluated using 5-fold cross validation, as shown in Appendix. Table 2(a) . Results suggested that the model with 2 LSTM layers, 150 units and 0.5 dropout rate achieved the highest average validation accuracy of 73.72%. Therefore, we selected this CNN+LSTM model to perform the rest of the experiments.

For TransformerEncoder, there are two hyperparameters: intermediate_dimensions and number_of_heads. Intermediate_dimensions represent the size of the two hidden fully connected layers, and the number_of_heads represent how much pieces the input sequence will be split into before feeding into the self-attention layer. Totally, 12 CNN+Transformer models with different sets of hyperparameters were evaluated through 5-fold CV as shown in Table 2(a) and 2(b) from Appendix. Ultimately, the CNN+Transformer model with intermediate_dims=6, number_of_heads=2, and 4 convolution layers was selected as the best-performing model.

To better optimize the CNN+Transformer structure, a variant of Adam, AdamW, was used for optimizing all 12 CNN+Transformer structures [6]. AdamW includes a term of weight decay in the updating equation, and can thus achieve faster convergence comparing to vanilla Adam [6].

2.2. Training and Testing with all subjects

All five models were trained and tested using separate training and test sets. Figure 2(a) shows the loss trajectory during the training process. Figure 2(c) shows the test accuracy of each model on test set. From the loss trajectory it can be noticed that Conv_LSTM rapidly overfitted within 20 epochs, which resulted in the lowest test accuracy of only 49.09%. Even though the loss of CNN+LSTM+Transformer was only slightly higher than the best-performed CNN+Transformer model, the test accuracy was only 65.97%. Therefore, in the following sections we gave up these two models and only kept the other three: CNN, CNN_LSTM and CNN+Transformer, the test accuracy of which were 71.67%,74.21% and 75.79% respectively.

2.3. Training and Testing with Subject 1

CNN, CNN+LSTM, and CNN+Transformer were all examined for their performance on the whole testing set when trained only with EEG data from subject 1. Each model's performance on the testing set based exclusively on each subject were listed in Table 3. All three models' performance on subject 1 were much higher (around 65%) than that of all other subjects. Comparing to the scenario when trained with EEG data from all subjects, testing accuracy for all 3 models were much lower across subject

2-9 when trained only with data from subject 1.

2.4. Training and Testing with Different Times

Figure 2(b) shows the classification accuracy on test set as a function of time. Here we used data of 250, 500, 750 and 1000 time bins to train and test the CNN, CNN+LSTM and CNN+Transformer models. Both CNN and CNN_LSTM achieved the highest accuracy on data of 500 time bins, while the test accuracy of CNN+Transformer peaked at 750 time bins. To investigate at least how much time is required for each model to achieve a reasonable accuracy, we used time sequence varying from 250-500 time bins and from 500-750 time bins to evaluate the performance of CNN/CNN+LSTM and CNN+Transformer respectively. The results suggested that CNN and CNN_LSTM could achieve a reasonable accuracy of 72.5% and 73.7% respectively, on time sequence of 350 time bins. Although CNN+Transformer can already achieve an accuracy of over 72% using only 250 time bins, we tend to think 750 time bins is the shortest time required for achieving a testing accuracy comparable to the model's overall performance.

3. Discussions

3.1. Comparison of training on subject 1 and all subjects

Table 3 in Appendix reported the test accuracy of CNN, CNN+LSTM and CNN+Transformer on each of the 9 subjects, using subject 1 and all subjects for training respectively. When using subject 1 for training, all three models achieved highest accuracy on the dataset from subject 1, while the test accuracy on other datasets was significantly lower. The performance of CNN and CNN_LSTM on test data from subject 4-7, and these of transformer on data set 4 and 8 were close to random guess(25%), which suggested that the models fail to apply features learned from subject 1 to these dataset. The test accuracy on other 8 subjects were significantly improved when the training data from these subjects were also included. In the meantime, the test accuracy of CNN and CNN+Transformer on subject 1 were also improved when trained across all subjects, increasing by 3.5% and 6.5% respectively. The result indicated the generalization ability of the models to improve the classification performance on subject 1 by learning from other datasets. Therefore, we concluded that increasing the number of variability of training data would improve the performance of all three models.

3.2. Transformer Performance and Model Comparison

For the performance of all 5 models, both CNN+Transformer+LSTM and ConvLSTM are the most sophisticated models, which achieved the lowest performance due to overfitting on the train_valid dataset. For example, the ConvLSTM model achieved nearly 0 loss after 100 epoch of training, which suggested that ConvLSTM model overfitted on the train_valid dataset and therefore made more errors in predicting testing datasets due to lack of generalizability.

For models with less complexity, CNN+LSTM achieves better performance than that of the baseline CNN model. Comparing to vanilla CNN models, CNN+LSTM has several advantages such that LSTM will memorize the important portion of the input long sequence and will utilize these memorized portions in making the next predictions [2]. Such structure can be more feasible in capturing the common features from all the input sequences and also be helpful in solving problems such as exploding gradients during the optimization. Thus, CNN+LSTM is more capable in applications for sequential data than that of baseline CNN.

Among all the models, CNN+Transformer achieves the highest performance. The key difference between CNN+Transformer and all other models is that TransformerEncoder was placed in front of the convolution layers, while all other models start from the convolution layers. TransformerEncoder directly takes the input original EEG sequence, and calculates the relative weight of each position in terms of the whole sequence by its self-attention layer [3, 4]. Such operation can consider the non-neighboring relationships between different parts of the sequence, which could be crucially important for the training [3, 4]. Besides, the TransformerEncoder's parameters are optimizable through backpropagation and gradient descent. These advantages enable the input sequence to contain relative spatial information on each position before taken by the following convolution layers, which could result in better performance.

In terms of the testing accuracy based on different time bins, CNN+Transformer's testing accuracy is the highest with 750 time bins, which is different from all other models with the best accuracy on fewer time bins. This could be explained by that: The larger variation part (250-500 time bins) of the input sequence is further magnified and emphasized while the low variation portion is downweighted after the TransformerEncoder processes the input sequence [3, 4]. With magnifying the larger variation, the processed input sequence is more suitable for increasing the generalizability of the model, resulting in better performance.

4. Appendix

Table 1 Architecture of models.

Details of Layers for CNN+LSTM					
Layer	Activation Function	Dropout	Output Shape	Specific Parameters	Total Parameters
Convolution1D	ELU	0.5	(N, 250, 25)	Filters=25; Kernel_size=10	5525
MaxPooling1D	N/A	N/A	(N, 84,25)	Pooling_size=(3,1);Padding="Same"	0
BatchNormalization	N/A	0.5	(N,84,25)	N/A	100
Convolution1D	ELU	0.5	(N,84,25)	Filters=50; Kernel_size=(10,1); Padding="Same"	12250
MaxPooling1D	N/A	N/A	(N,28,50)	Pooling_size=(3,1); Padding="Same"	0
BatchNormalization	N/A	0.5	(N, 28, 50)	N/A	200
Convolution1D	ELU	0.5	(N,28,100)	Filters=100; Kernel_size=(10,1); Padding="Same"	50100
MaxPooling1D	N/A	N/A	(N,10,100)	Pooling_size=(3,1); Padding="Same"	0
BatchNormalization	N/A	0.5	(N,10,100)	N/A	400
Convolution1D	ELU	0.5	(N,10,200)	Filters=200; Kernel_size=(10,1); Padding="Same"	200200
MaxPooling1D	N/A	N/A	(N,4,200)	Pooling_size=(3,1); Padding="Same"	0
BatchNormalization	N/A	0.5	(N,4,200)	N/A	800
LSTM	ELU	0.5	(N,4,150)	Recurrent_dropout=0; Return_sequence="True"	210600
LSTM	ELU	0.5	(N,10)	Recurrent_dropout=0; Return_sequence="True"	6440
Dense	Softmax	N/A	(N, 4)	Units=4	44
Optimizer: Adam;Learning rate=10^{-3}					

Details of Layers for CNN+Transformer					
Layer	Activation Function	Dropout	Output Shape	Specific Parameters	Total Parameters
Transformer_Encoder	ELU	0.5	(N, 250, 22)	Intermediate_Dims=6; Num_heads=2	2404
Reshape	N/A	N/A	(N, 250, 1, 22)	N/A	0
Convolution2D	ELU	0.5	(N, 250, 1, 50)	Filters=50; Kernel_size=(10,1); Padding="Same"	11050
MaxPooling2D	N/A	N/A	(N, 84,1, 50)	Pooling_size=(3,1); Padding="Same"	0
BatchNormalization	N/A	0.5	(N, 84,1, 50)	N/A	200
Convolution2D	ELU	0.5	(N, 84, 1, 100)	Filters=100; Kernel_size=(10,1); Padding="Same"	50100
MaxPooling2D	N/A	N/A	(N, 28,1, 100)	Pooling_size=(3,1); Padding="Same"	0
BatchNormalization	N/A	0.5	(N, 28,1, 100)	N/A	400
Convolution2D	ELU	0.5	(N, 28, 1, 50)	Filters=50; Kernel_size=(10,1); Padding="Same"	50050
MaxPooling2D	N/A	N/A	(N, 10,1, 50)	Pooling_size=(3,1); Padding="Same"	0
BatchNormalization	N/A	0.5	(N, 10,1, 50)	N/A	200
Convolution2D	ELU	0.5	(N, 10, 1, 100)	Filters=100; Kernel_size=(10,1); Padding="Same"	50100
MaxPooling2D	N/A	N/A	(N, 4,1, 100)	Pooling_size=(3,1); Padding="Same"	0
BatchNormalization	N/A	0.5	(N, 4,1, 100)	N/A	400
Flatten	N/A	N/A	(N, 400)	N/A	0
Dense	Softmax	N/A	(N, 4)	Units=4	1604
Optimizer: AdamW, learning rate = 1e-3, other parameters are used as the default.					

Table 2 Hyperparameter tuning. (a) CNN+Transformer (b) CNN+LSTM

CNN_Transformer Model	Intermediate _Dimensions	Number_Of_ Heads	CNN_Layers	5-Fold CV Accuracy %
1	6	2	3	72.66
2	6	11	3	72.03
3	6	22	3	72.85
4	8	2	3	72.52
5	8	11	3	72.78
6	8	22	3	72.19
7	**6**	**2**	**4**	**73.82**
8	6	11	4	72.49
9	6	22	4	72.85
10	8	2	4	73.35
11	8	11	4	72.94
12	8	22	4	72.98

(a)

CNN_LSTM Model	Units	Dropout_Rate	LSTM_Layers	5-Fold CV Accuracy %
1	50	0.3	2	72.27
2	50	0.4	2	71.47
3	50	0.5	2	71.93
4	100	0.3	2	71.74
5	100	0.4	2	71.91
6	100	0.5	2	73.30
7	150	0.3	2	70.14
8	150	0.4	2	72.29
9	**150**	**0.5**	**2**	**73.72**
10	50	0.3	3	71.17
11	50	0.4	3	72.28
12	50	0.5	3	71.73
13	100	0.3	3	71.49
14	100	0.4	3	72.39
15	100	0.5	3	71.69
16	150	0.3	3	71.60
17	150	0.4	3	71.91
18	150	0.5	3	71.73

(b)

Table 3 Comparison of test performance using subject 1 and all subject for training.

	Train on subject 1			Train on all subjects		
Subject ID	CNN %	CNN_ LSTM %	CNN_ Transformer %	CNN %	CNN_ LSTM %	CNN_ Transformer %
1	61.50	68.00	69.00	65.00	63.50	75.50
2	36.00	32.00	35.50	50.50	54.50	64.50
3	47.50	59.00	67.00	79.50	79.50	77.50
4	29.50	22.50	28.50	64.00	76.50	73.00
5	29.79	26.06	46.81	78.19	77.66	78.72
6	35.20	30.61	45.92	78.06	76.02	69.39
7	42.00	38.50	42.00	72.00	76.50	80.00
8	42.50	46.50	29.50	70.50	71.50	78.50
9	35.64	38.30	44.15	73.40	83.51	85.64

5. Reference

1. Monsoor, Tonmoy, et al. "CNN with data preprocessing.ipynb." BruinLearn, https://bruinlearn.ucla.edu/courses/154235/files/12794459/download?download_frd=1.
2. Manaswi, N. RNN and LSTM. In Deep Learning with Applications Using Python; Apress: Berkeley, CA, USA, 2018.
3. Vaswani, Ashish, et al. “Attention Is All You Need.” ArXiv.org, 6 Dec. 2017, https://arxiv.org/abs/1706.03762.
4. Keras Team. “Keras Documentation: Transformerencoder Layer.” Keras, https://keras.io/api/keras_nlp/layers/transformer_encoder/.
5. L. Feng, C. Cheng, M. Zhao, H. Deng and Y. Zhang, "EEG-Based Emotion Recognition Using Spatial-Temporal Graph Convolutional LSTM With Attention Mechanism," in IEEE Journal of Biomedical and Health Informatics, vol. 26, no. 11, pp. 5406-5417, Nov. 2022, doi: 10.1109/JBHI.2022.3198688.
6. Loshchilov, Ilya, and Frank Hutter. “Decoupled Weight Decay Regularization.” ArXiv.org, 4 Jan. 2019, https://arxiv.org/abs/1711.05101.