ECE247 Report

Xu Yan
705062008
yanxu2015@g.ucla.edu,

Yuedong Cui
805728143
cui54@g.ucla.edu

Yu Cao
705908390
yucao26@g.ucla.edu

Zeang Chen
506072941
shyzc4@g.ucla.edu

Abstract

In this project, we examined three hybrid deep learning structures on a EEG-based motor imagery task, namely CNN+LSTM, CNN+GRU, and CNN+Transformer. The best accuracy reaches 72% by CNN+LSTM. Justification was provided to validate our results as well as some thoughts on finetuning. With our best model, we also investigated some key factors that may influence the classification performance, including subject variability, time length, bandpass filtering, preprocessing, etc. and presented corresponding experimental results with visualization.

1. Introduction

Motor imagery (MI) decoding remains a difficult task due to the low signal-to-noise ratio and the high-dimensional structure of EEG signals. The occurrence of deep learning methods, ranging from convolutional neural networks, recurrent neural networks, to the Transformer, have allowed EEG-based MI to take a big step forward.

A popular deep learning model branch for extracting spatial patterns is the Convolutional Neural Network (CNN). The convolutional layer is a crucial component functioning as feature extraction. This suggests that CNN can not only serve as an independent classifier but also function as an effective feature extractor in more complex models. Over half of the studies have utilized CNN models to classify MI signals using various types of CNN, including vanilla CNN with both shallow and deep architectures [5, 9, 10], as well as other CNN variants, such as attention-based CNN [2, 11], residual-based CNN [13, 20], and inception-based CNN [3, 19], etc.

Compared with CNN, another commonly used type of deep learning architecture, recurrent neural network (RNN), is particularly useful for analyzing time-series data. It can detect patterns and extract temporal features, making it a valuable tool for analyzing signals like EEG. Among all variants of RNN, the long short-term memory networks (LSTM) and gated recurrent units (GRU) are two prominent RNN architectures that have been extensively used in the literature for analyzing time-series data. LSTM can effectively learn long-term temporal dependencies and alleviate the vanishing gradient problem encountered by traditional RNN with the help of input, forget, and output gates. In the context of MI tasks, LSTM models have been employed in several studies to achieve high classification accuracy [8, 17]. While LSTM has three gates to control the flow of information, GRU only has two gates, making it more lightweight. Examples such as [12] reveals that GRU achieves competitive accuracies with LSTM.

The thriving Transformer offers new possibilities to further improve the MI classification model. Instead of utilizing a convolutional or recurrent layer to extract information, Transformer employs multi-head attention. Different from the recurrent networks which process the sequence recursively element by element, Transformer takes the whole sequence as an input and has its unique multi-head attention mechanism optimized for parallelization. Moreover, compared to other deep learning models, Transformer is more interpretable. Recently, a few studies have attempted to adopt Transformer on MI tasks [7, 15].

EEG is typically regarded as having high temporal resolution in milliseconds but relatively low spatial resolution in centimeters [1]. The property of EEG signals motivated us to design models which can fully exploit both spatial and temporal information of EEG in MI tasks. Drawing on previous research, our hypothesis posits that CNN is effective in extracting spatial information, while alternative architectures may be more adept at handling temporal information extraction. Based on the characteristics of models mentioned above, we proposed three hybrid model structures, i.e. CNN+LSTM, CNN+GRU, CNN+Transformer, for this four-class MI classification on BCI-C IV-2a dataset [18] to see which deep learning structure is the most appropriate. We also provided our answers to some hot topics in the EEG-based MI field based on the results of our model, including subject variability, bandpass filtering, etc.

2. Results

In this section, we would report the experimental results and gave brief descriptive analysis. Note that **more detailed**

analysis is included in the Discussion section.

2.1. General Performance

Table 1 presents the average accuracies and standard deviations of all the models we proposed. In the three models we designed (row 3, 5, 7), CNN+LSTM achieves the highest accuracy 72% with the smallest standard deviation 0.93. We also ran the vanilla version of all basic structures we used to compare with the architectures we proposed. Generally speaking, our hybrid models greatly outperform the corresponding vanilla versions (row 2, 4, 6) in both accuracies (by 10% to 20%) and stability, which provide evidence that combining structures do help with classification.

2.2. Experiment on Subject Dependence

We trained our best model architecture CNN+LSTM on each subject's training data and got the results as shown in Table 2 except for row 1, where we trained on the whole training data and tested on subject 1's test data. By comparing row 1 and row 2, there's no benefit from training on subject 1's data specifically. Instead, training across 9 persons' data can help with the classification on subject 1 with 5% higher accuracy.

2.3. Experiment on Length of Time Window

We utilized different time windows on samples to see how the classification is affected. Since the original time length of data is 1000 ms, we took 5 different time windows as $\{200, 400, 600, 800, 1000\}$ and their corresponding results are plotted in Figure 1. As the time window increases, the performance will go up first and then down with the maximum point around 74% falling in range $\{400, 600\}$.

2.4. Experiment on Frequency Bands

We filtered our data to be in 8-30 Hz range and trained them with our CNN+LSTM model architecture. From row 3 and row 8, it is obvious that training with filtered data does not help with learning. Rather, the accuracy drops by 30% with increased standard deviation.

3. Discussion

3.1. Performance Interpretation

3.1.1 Choice of Hyperparameters

A general hyperparameter is learning rate which controls the speed of the process. If it is too small, the model may take longer time to converge or stuck in a local minimum. Otherwise, the model may overshoot resulting in oscillation reflected in the error graph.

CNN: The number of convolution layer decides the ability that how complex feature could be learned by the model. More layers could increase the accuracy but also make the model overfitting easily. Increasing the number of filters may cause overfitting since the number of neurons increases which means the model becomes more complex. Similarly, increasing kernel size results in a larger receptive field of current layer, i.e. capturing more general features of EEG data. Positively thinking, we could gain more information and it may be useful during the training process but it also increases the risk of overfitting and computational time. To avoid overfitting, we added max pooling layers where downsampling or reducing the number of parameters, batch normalization layers, and dropout layers. However, adding too many layers above or setting pool size too big or dropout probability too high could cause underfitting and a lower training accuracy since input information is less.

LSTM/GRU: The number of units in LSTM/GRU function determines the number of LSTM/GRU cells in the layer. More units could capture more features in the data but also be more prone to overfitting.

Transformer: The 'num_heads' parameter determines the number of parallel self-attention heads. Each head could capture multiple features of the data simultaneously. More heads means more complexity and may cause overfitting. The 'key_dim' parameter controls the dimensionality of the key vectors used to compute its attention scores, i.e. the importance of different parts of the data. It enables the model to capture complex patterns if the number increases but could also overfit the model. Large 'key_dim' corresponds to high scores assigned by attention mechanism to some specific positions in the data and potentially makes the model to just concentrate on these parts and finally memorize them. Another parameter named 'value_dim' is the dimensionality of the value vectors used to compute the output of the attention mechanism which represents a weighted sum of the values. If this number increases, it could allow model to detect more fine-grained details of the input data with the incremental chance of overfitting with the same reason as 'key_dim'.

3.1.2 Model Comparison

Theoretically, GRU should run faster than LSTM as it has a simpler architecture with only two gates (reset and update) whereas LSTM has more parameters. On the other hand, GRU doesn't need a memory unit which might save more time. However, in reality, they both had similar operation time around 13s for each epoch. The reason could be before LSTM/GRU, we used three CNN layers to extract features and this took up most of time. Another reason might be that our input data size was too small for GRU to prove its ability. However, the performance of two models were pretty close. When it came to Transformer, the speed was much faster since Transformer is highly parallelizable and well-matched with GPU. Meanwhile, Transformer processes data in parallel which is more efficient than LSTM/GRU who sequentially proceed data. Another interesting point was found that sometimes even the loss between training and validation error was very small, the difference between training accuracy and validation accuracy was still large. A possible explanation could be the existence of 'key_dim' and 'value_dim' so that the model just memorized part of the training data instead of really learning features which was revealed on the validation accuracy as it converged earlier.

To further evaluate how our CNN+LSTM model performs, we also provided the confusion matrix in Figure 2 to present the classification results. The lighter the color, the higher the recognition rate between classes. It is straightforward that our CNN+LSTM model has difficulty in classifying the Down (foot) class, which is easy to be misclassified as either Left or Right. Besides, we realized that the Right class always has a high probability when misclassifying.

3.2. Experiments Interpretation

3.2.1 Experiment on Subject Dependence

There are two ways to train models in EEG classification tasks [6]. One is subject-independent, ignoring the huge variations in EEG signals between different subjects and mix the data from all subjects to train and test one general classifier. The other one is subject-dependent, representing training a specific model for each subject to reduce the effect of individual variations on the results. The first way is more cost-effective while requires the model to have more generalization ability. Our hypothesis here was that the model training on a particular subject's data can work better than that training on all subjects' data. To verify, we conducted an experiment and the results are listed as row 1 and 2 in Table 2. Specifically, row 1 is what we got by training on the whole training set. We surprisingly found that though the customized model for subject 1 can learn more information from this person during training, it does not excel the performance of the general classifier. A possible reason might be that since the size of training set is much smaller (around 1/9), there is not enough data for the model to learn, i.e. easy to overfit on this high-dimensional EEG dataset. Row 3-10, results from other subjects with the same trend, also validate our inference. Note that for fair comparison, we did not try to simplify our model structure here. Rather, we only finetuned it to optimize the classification accuracy for subject 1. This result indicates that training across all subject does help with improving the accuracy for a specific subject due to sufficiency of training data to learn.

3.2.2 Experiment on Length of Time Window

Our hypothesis on this question was that only the first 400-500 samples of each trial are helpful (refering to the data visualization figure from *CNN with data preprocessing.ipynb*) since the latter parts of four classes do not have any distinguishable patterns and the amplitude changes are also very subtle. The trend in Figure 1 is consistent with our hypothesis. This finding has guided our data preprocessing to only focus on the first half data of each trial.

3.2.3 Experiment on Frequency Bands

The frequency range of EEG oscillations can be separated into several bands, i.e. α (8-12 Hz), β (13-30 Hz), and γ (>30 Hz) [14]. It also has been found that for EEG-based MI signals, the event-related desynchronization (ERD) and event-related synchronization (ERS) of sensorimotor rhythms mainly occurs 8-30 Hz frequency. Therefore, to make the signal more informative to the model, we applied fifth-order Butterworth filter with range 8-30 Hz during preprocessing and we expected the model to perform better with bandpass filtered data. However, the model performance goes against our assumption. To explore the reason behind, we plotted the topographic maps of power spectral density of both α and β band for different trials with MNE-Python package [4] shown in Figure 4. Our thoughts were that if the data is separable, there should be some clear patterns as indicators. Unfortunately, by looking at Figure 4, there is no distinguishable patterns for each class. Particularly, in β band, the activation areas are even very similar for all four classes in the central parietal lobe area and occipital lobe (electrode CP3, CP1, CP2, CP4, P1, Pz, and P2). This finding proves that the data itself is hard to classify with these two bands if no other feature extraction technique is applied, which explains why our accuracy is low.

3.2.4 Experiment on Data Preprocessing

The outcome mentioned in Section 3.2.3 enlightened us to check the training data quality after preprocessing. We visualized the whole training set before and after preprocessing with t-SNE [16] via a scatter plot as displayed in Figure 3. We splitted them into two groups based on labels for better visualization. It is apparent that after preprocessing, shown in Figure 3(b) and Figure 3(d), samples are more organized with less chaos. This testifies that the prepocessing works and accounts for the effectiveness of MI classification. However, we did not easily observe any separate cluster in these plots. It inspires us to design a more complex model to better learn the information. Also, another possible reason is that the spatial distribution of samples in a much higher dimensional space is hard to be displayed in 2D.

References

[1] Boris Burle, Laure Spieser, Clémence Roger, Laurence Casini, Thierry Hasbroucq, and Franck Vidal. Spatial and temporal resolutions of eeg: Is it really black and white? a scalp current density view. *International Journal Of Psychophysiology*, 97(3):210–220, 2015. 1

[2] Chen-Chen Fan, Hongjun Yang, Zeng-Guang Hou, Zhen-Liang Ni, Sheng Chen, and Zhijie Fang. Bilinear neural network with 3-d attention for brain decoding of motor imagery movements from the human eeg. *Cognitive Neurodynamics*, 15:181–189, 2021. 1

[3] Álvaro Fernández-Rodríguez, Francisco Velasco-Álvarez, and Ricardo Ron-Angevin. Review of real brain-controlled wheelchairs. *Journal of Neural Engineering*, 13(6):061001, 2016. 1

[4] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A. Engemann, Daniel Strohmeier, Christian Brodbeck, Roman Goj, Mainak Jas, Teon Brooks, Lauri Parkkonen, and Matti S. Hämäläinen. MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7(267):1–13, 2013. 3

[5] Yimin Hou, Lu Zhou, Shuyue Jia, and Xiangmin Lun. A novel approach of decoding eeg four-class motor imagery tasks via scout esi and cnn. *Journal of Neural Engineering*, 17(1):016048, 2020. 1

[6] Sunhee Hwang, Pilhyeon Lee, Sungho Park, and Hyeran Byun. Learning subject-independent representation for eeg-based drowsy driving detection. In *2021 9th International Winter Conference on Brain-Computer Interface (BCI)*, pages 1–3. IEEE, 2021. 3

[7] Demetres Kostas, Stephane Aroca-Ouellette, and Frank Rudzicz. Bendr: Using transformers and a contrastive self-supervised learning task to learn from massive amounts of eeg data. *Frontiers in Human Neuroscience*, 15:653659, 2021. 1

[8] Shiu Kumar, Alok Sharma, and Tatsuhiko Tsunoda. Brain wave classification using long short-term memory network based optical predictor. *Scientific Reports*, 9(1):9153, 2019. 1

[9] Vernon J Lawhern, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance. Eegnet: A compact convolutional neural network for eeg-based brain–computer interfaces. *Journal Of Neural Engineering*, 15(5):056013, 2018. 1

[10] Ming-Ai Li, Jian-Fu Han, and Li-Juan Duan. A novel mi-eeg imaging with the location information of electrodes. *IEEE Access*, 8:3197–3211, 2019. 1

[11] Xiuling Liu, Yonglong Shen, Jing Liu, Jianli Yang, Peng Xiong, and Feng Lin. Parallel spatial–temporal self-attention cnn-based motor imagery classification for bci. *Frontiers in Neuroscience*, 14:587520, 2020. 1

[12] Tian-jian Luo, Chang-le Zhou, and Fei Chao. Exploring spatial-frequency-sequential relationships for motor imagery classification with recurrent neural network. *BMC Bioinformatics*, 19(1):1–18, 2018. 1

[13] Yazeed K Musallam, Nasser I AlFassam, Ghulam Muhammad, Syed Umar Amin, Mansour Alsulaiman, Wadood Abdul, Hamdi Altaheri, Mohamed A Bencherif, and Mohammed Algabri. Electroencephalography-based motor imagery classification using temporal convolutional network fusion. *Biomedical Signal Processing and Control*, 69:102826, 2021. 1

[14] Ernst Niedermeyer and FH Lopes da Silva. *Electroencephalography: Basic principles, clinical applications, and related fields*. Lippincott Williams & Wilkins, 2005. 3

[15] Yonghao Song, Xueyu Jia, Lie Yang, and Longhan Xie. Transformer-based spatial-temporal feature learning for eeg decoding. *ArXiv Preprint ArXiv:2106.11170*, 2021. 1

[16] David Vazquez, Antonio M Lopez, Javier Marin, Daniel Ponsa, and David Geronimo. Virtual and real world adaptation for pedestrian detection. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 36(4):797–809, 2013. 3

[17] Ping Wang, Aimin Jiang, Xiaofeng Liu, Jing Shang, and Li Zhang. Lstm-based eeg classification in motor imagery tasks. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 26(11):2086–2095, 2018. 1

[18] Yijun Wang, Zhiguang Zhang, Yong Li, Xiaorong Gao, Shangkai Gao, and Fusheng Yang. Bci competition 2003-data set iv: an algorithm based on cssd and fda for classifying single-trial eeg. *IEEE Transactions on Biomedical Engineering*, 51(6):1081–1086, 2004. 1

[19] Ce Zhang, Young-Keun Kim, and Azim Eskandarian. Eeg-inception: an accurate and robust end-to-end neural network for eeg-based motor imagery classification. *Journal of Neural Engineering*, 18(4):046014, 2021. 1

[20] Xuyang Zhu, Peiyang Li, Cunbo Li, Dezhong Yao, Rui Zhang, and Peng Xu. Separated channel convolutional neural network to realize the training free motor imagery bci systems. *Biomedical Signal Processing And Control*, 49:396–403, 2019. 1

Appendix

A. Experimental Results

Name	Train Acc.	Test Acc.	Test Std.
LSTM	0.7205	0.5073	1.2767
CNN+LSTM	0.7774	**0.7201**	**0.9278**
GRU	0.7198	0.5372	1.2294
CNN+GRU	0.8011	0.7173	0.9969
Transformer	0.6691	0.5767	1.1747
CNN+Transformer	0.7693	0.6716	1.0460
Bandpass	0.5612	0.4953	1.3256

Table 1. Performance of different methods on the whole dataset.

	Subject ID	Train Acc.	Test Acc.
1	1	0.7774	0.6650
2	1	0.8663	0.6150
3	2	0.8416	0.4600
4	3	0.8738	0.4300
5	4	0.7871	0.4605
6	5	0.8589	0.5112
7	6	0.8540	0.4949
8	7	0.8911	0.6095
9	8	0.9158	0.5382
10	9	0.9233	0.5479

Table 2. Performance of CNN+LSTM on each subdataset. Column 1 is the index of each row.

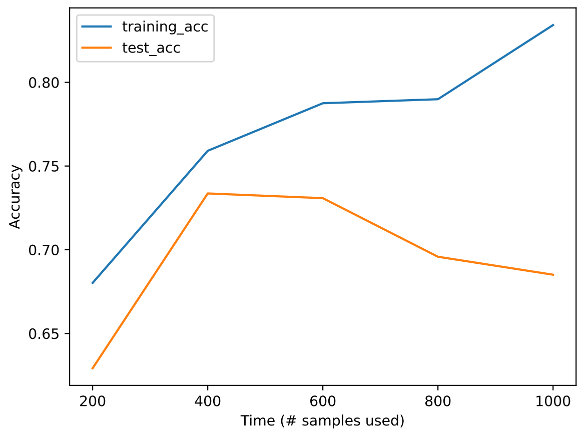


Figure 1. Classification accuracy as a function of time.

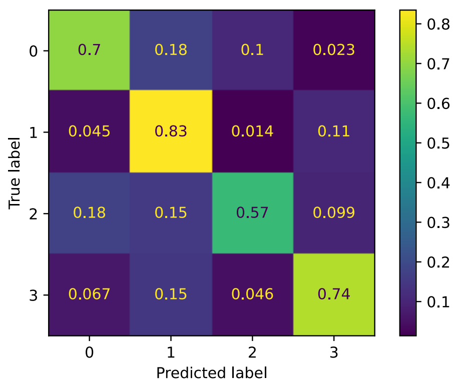
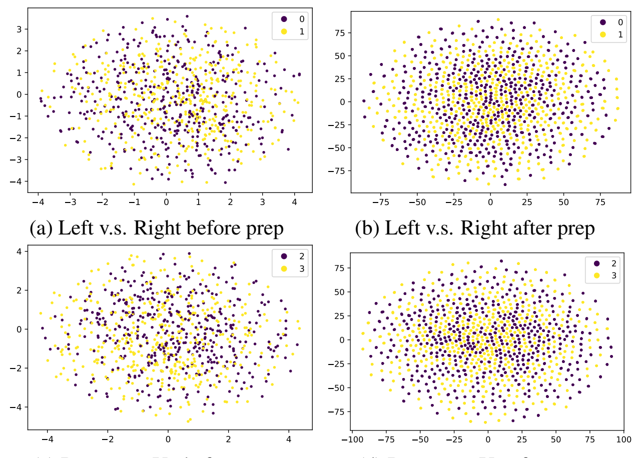


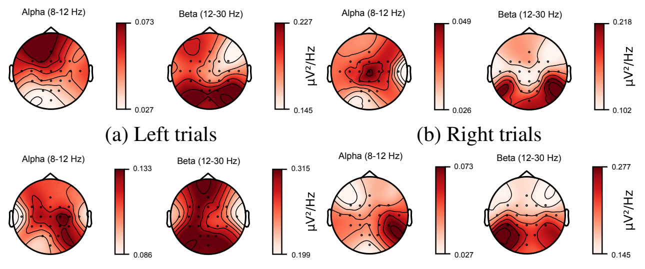
Figure 2. Confusion matrix of CNN+LSTM on the whole dataset.



(a) Left v.s. Right before prep (b) Left v.s. Right after prep

(c) Down v.s. Up before prep (d) Down v.s. Up after prep

Figure 3. Feature visualization before and after preprocessing for two pairs of labels. Legends 0, 1, 2, and 3 represent for left, right, down (foot), and up (tongue), respectively.



(a) Left trials (b) Right trials

(c) Down trials (d) Up trials

Figure 4. Visualization of PSD in 8-30 Hz band of the whole dataset (first 500 ms).

B. Model Architectures

Layer (type)	Output Shape	Param #
Conv2D	(None, 250, 1, 16)	3184
MaxPooling2D	(None, 84, 1, 16)	0
BatchNormalization	(None, 84, 1, 16)	64
Dropout	(None, 84, 1, 16)	0
Conv2D	(None, 84, 1, 32)	4640
MaxPooling2D	(None, 28, 1, 32)	0
BatchNormalization	(None, 28, 1, 32)	128
Dropout	(None, 28, 1, 32)	0
Conv2D	(None, 28, 1, 32)	9248
MaxPooling2D	(None, 10, 1, 32)	0
BatchNormalization	(None, 10, 1, 32)	128
Dropout	(None, 10, 1, 32)	0
Flatten	(None, 320)	0
Dense	(None, 32)	10272
Reshape	(None, 32, 1)	0
LSTM	(None, 32)	4352
Dense	(None, 4)	132
Total params	32,148	
Trainable params	31,988	
Non-trainable params	160	

Table 3. Details of CNN+LSTM Structure

B.1. Details of Training

Training set (2115 trials) and test set (443 trials) were load from original dataset. Then, the training set was further split to training data (1740 trials) and validation data (375 trial) by random selection. After this, all three data sets were trimmed to 500 time bins as mention in Section 3.2.2 and were separated to sections with equal length which was the maximum value of each section so that data length was shortened but key information still remained. Then we averaged values of each section together with addition of Gaussian noise in order to make our data more robust. Subsequently, training, validation and test data were subsampled to 6960 trials, 1500 trials and 1772 trials to increase the number of data points. Finally, we added the width of the segment with 1 as the preparation for the following 2D convolution layer. Finally, data were input to different models listed in Table 1. The activation function, classifier, optimizer and loss function were chosen as ELU, softmax, Adam and categorical cross entropy respectively.

Layer (type)	Output Shape	Param #
Conv2D	(None, 250, 1, 25)	5525
MaxPooling2D	(None, 84, 1, 25)	0
BatchNormalization	(None, 84, 1, 25)	100
Dropout	(None, 84, 1, 25)	0
Conv2D	(None, 84, 1, 30)	7530
MaxPooling2D	(None, 28, 1, 30)	0
BatchNormalization	(None, 28, 1, 30)	120
Dropout	(None, 28, 1, 30)	0
Conv2D	(None, 28, 1, 30)	9030
MaxPooling2D	(None, 10, 1, 30)	0
BatchNormalization	(None, 10, 1, 30)	120
Dropout	(None, 10, 1, 30)	0
Flatten	(None, 300)	0
Dense	(None, 100)	30100
Reshape	(None, 100, 1)	0
GRU	(None, 10)	390
Dense	(None, 4)	44
Total params	52,959	
Trainable params	52,789	
Non-trainable params	170	

Table 4. Details of CNN+GRU Structure

Layer (type)	Output Shape	Param #
InputLayer	[(None, 250, 1, 22)]	0
Conv2D	(None, 250, 1, 30)	2010
MaxPooling2D	(None, 125, 1, 30)	0
BatchNormalization	(None, 125, 1, 30)	120
Dropout	(None, 125, 1, 30)	0
Conv2D	(None, 125, 1, 30)	2730
MaxPooling2D	(None, 63, 1, 30)	0
LayerNormalization	(None, 63, 1, 30)	60
MultiHeadAttention	(None, 63, 1, 30)	399
Dropout	(None, 63, 1, 30)	0
Flatten	(None, 1890)	0
Dense	(None, 4)	7564
Total params	12,883	
Trainable params	12,823	
Non-trainable params	60	

Table 5. Details of CNN+Transformer Structure