Emotion Recognition based on EEG signals using Deep CNN model

Keywords: Emotion, Valence-Arousal, EEG, CNN, ICA, DWT

I. Motivation

Emotions states are highly correlated with human behaviors and thoughts. Thus, study on emotion recognition using electroencephalogram (EEG) signals is a significant factor in the brain-computer interface system (BCI). In addition, understanding and knowing emotions could be helpful during treatment of psychological disorders such as attention deficit hyperactivity disorder (ADHD).

In this study, we developed an emotion recognition system based on the valence-arousal model. Independent component analysis (ICA) was applied in order to remove the ocular movement effect. Afterward, we applied discrete wavelet transform (DWT) on the processed EEG signals which was separated to gamma, beta, alpha and theta bands. Shannon's entropy and signal energy were computed with a temporal window from these 4 channels. Deep convolutional neural network (CNN) model is trained to classify the signal into valence-arousal space.

II. Related Works

Many researchers had tried to construct reliable emotion recognition system through various machine learning and feature extraction techniques. Bazgir [1] constructed SVM, KNN and ANN as classifiers, processed EEG signals with DWT and extracted significant features with PCA. Their model reached 91.3% accuracy on DEAP dataset. CNN models were widely used due to its ability of extracting important features from signals automatically. Keelawat [2] use deep CNN extracted features from standardized and processed signals. Their model reached 73.06% accuracy on their music stimulate dataset. Besides EEG, CNN could be used in extracting features from electrocardiogram (ECG) and galvanic skin response (GSR) [3]. The CNN model reached 81% accuracy of arousal and 71% accuracy of valence on AMIGOS dataset [4].

III. Contributions

In this study, we proposed an 2D deep CNN based model extracting spatial and temporal features for emotion recognition on physiological signal dataset,

AMIGOS [4], with 71.27% and 80.48% accuracy on arousal and valence classifications.

IV. Teamwork

I am the only one team member in this project. However, I often got advices during group meeting such as applying ICA to remove ocular movement effect from EEG signals.

V. Design

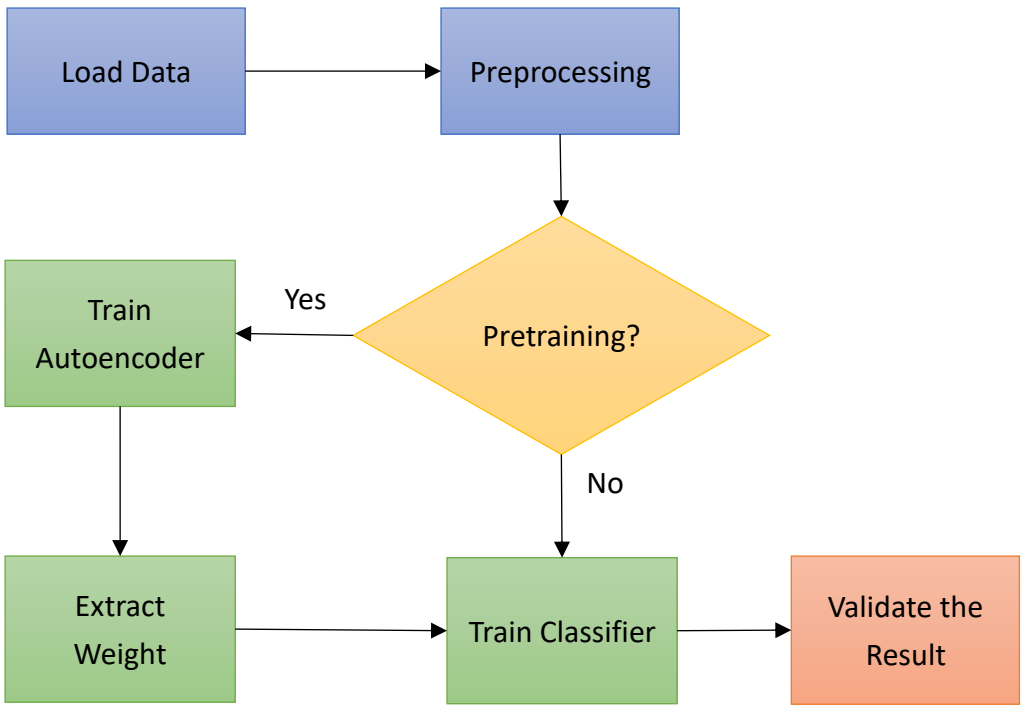


Figure 1. The flow chart of our emotion recognition experiment design.

In our experiment design, we loaded the data first and conducted preprocessing process. Afterwards, processed data were inputted into a convolutional autoencoder or a classifier depending on whether the user wants pretraining or not, and we could validate the results for final stage.

VI. Implementations

A. Load Data

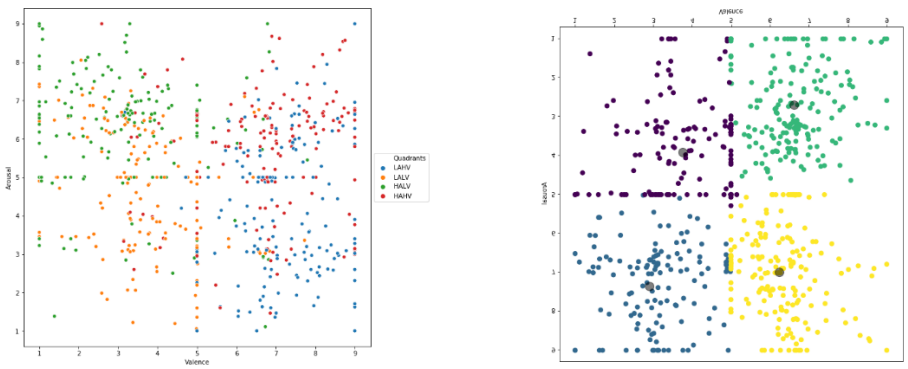
We use AMIGOS [4] as our training and validation dataset. This dataset

conducted two experiments. First, 40 participants watched 16 short videos which are less than 250 seconds individually. Second, five group of four participants and 17 people individually watched four long movies. After both of experiments, self-assessment including valence and arousal scaled from 1 to 9 were applied.

In our experiment, we used EEG data recorded during short videos experiments. The EEG data were preprocessed with a sampling frequency of 128Hz. Because pytorch [5] was used to construct deep CNN models and the original data from AMIGOS dataset [4] was store in matlab file, we used scipy.io.loadmat function to load matlab file to python object.

B. Preprocessing

First, we needed to determine the classification ground truth for each EEG signal. In our experiments, if the self-assessment value was greater than 5 we regarded as high, on the other hand, if the value was less than 5 we regarded as low. However, some participants assessed their emotion states with value equal to 5. In this case, K-means [6] was applied to determine which cluster this instance belongs to.

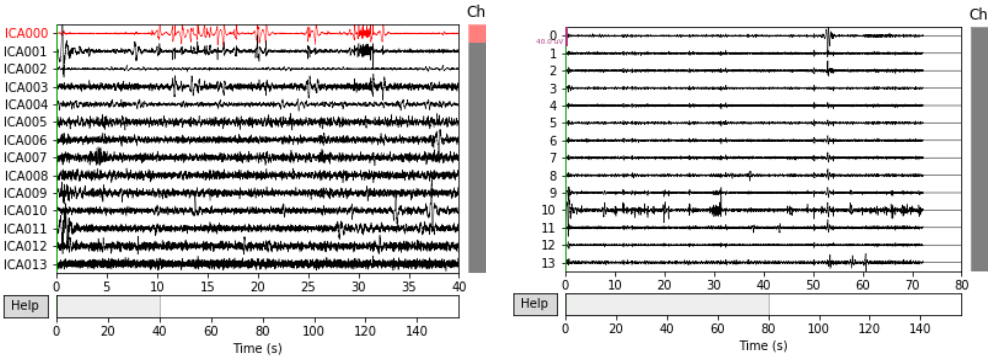


(a) (b)

Figure 2. (a) The self-assessment score plotted in valence-arousal space, the color represents emotion sates that experiment conductors of AMIGOS dataset [4] wanted to stimulate. (b) The ground truth label after K-means clustering [6]

After determined the ground truth for each instance. We could start to process the input EEG signals. First, we needed to remove NaN instances in AMIGOS dataset [4]. After the remove process, there were 614 instances in total. Second, we dropped recording data during the first three seconds due to noise and less influence of the emotion state. Before applying DWT, we need to remove the

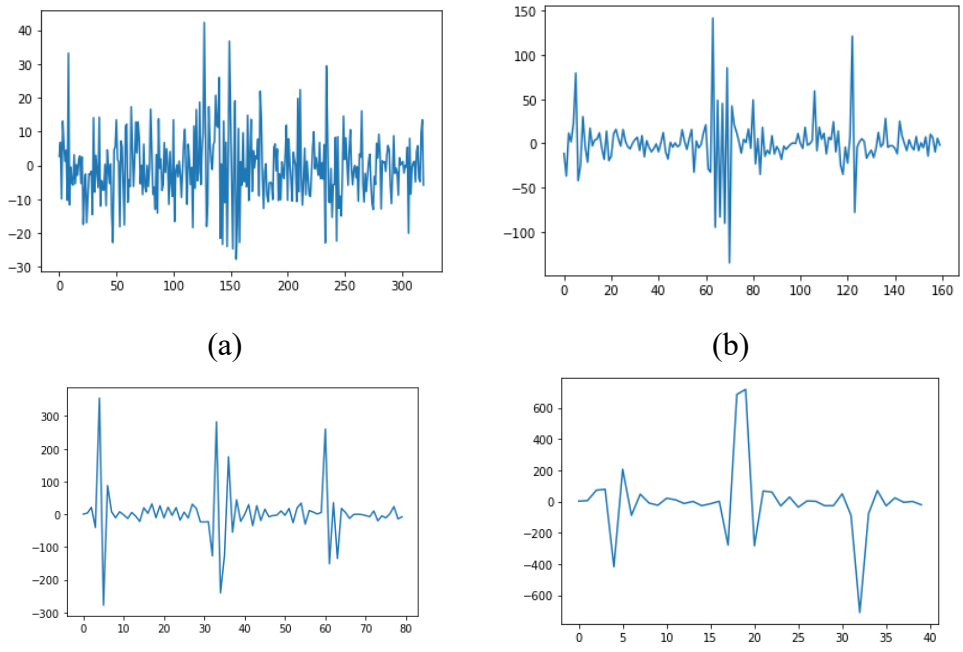
ocular movement effect on EEG signals. Thus, ICA was applied to find the ocular movement component and drop the signal. In our implementation, mne library [7] [8] was used in removing ocular movement.



(a) (b)

Figure 3. (a) The signal I regarded as ocular movement signal (b) EEG signals after removing component in ICA channel 0 [7] [8]

When all EEG signals were processed, DWT was applied to decompose signal into four frequency bands: gamma (32-64Hz), beta(16-32Hz), alpha(8-16Hz), and theta(4-8Hz) by Daubechies 4 as the mother wavelet.



(a) (b)

(c) (d)

Figure 4. (a) Gamma band signal (b) Beta band signal (c) Alpha band signal (d) theta band signal

Before computing entropy or energy in each temporal window, all coefficients in frequency bands were normalized in interval [0, 1]. In this study,

the temporal window was set to 4 seconds with overlapping of 2 seconds.

The entropy and energy of signal over a temporal window of a frequency band is computed as follows:

$$Entropy_j = -\sum_{i=0}^{n} C_j^i log C_j^i \quad (1)$$

$$Energy_j = \sum_{i=0}^{n} \left(C_j^i\right)^2 \quad (2)$$

Where C_j^i is the ith wavelet coefficient within j temporal window containing n coefficients.

C. Models

Our deep CNN model was based on EEG Net [9], which is a 2D CNN model considered both spatial relationship and temporal relationship.

```
ClassifierEEGDWT2D(
  (encoder): EncoderEEGDWT2D(
    (conv1): Sequential(
      (0): Conv2d(4, 16, kernel_size=(1, 5), stride=(1, 1), padding=(0, 2), bias=False)
      (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (conv2): Sequential(
      (0): Conv2d(16, 32, kernel_size=(2, 1), stride=(1, 1), groups=16, bias=False)
      (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace)
    )
    (pool1): MaxPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0, dilation=1,
ceil_mode=False)
    (dropout1): Dropout2d(p=0.5)
    (sep_conv): Sequential(
      (0): Conv2d(32, 32, kernel_size=(1, 15), stride=(1, 1), padding=(0, 7), bias=False)
      (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU(inplace)
    )
    (pool2): MaxPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0, dilation=1,
ceil_mode=False)
    (dropout2): Dropout(p=0.5)
  )
  (clf): Sequential(
    (0): Linear(in_features=2080, out_features=512, bias=True)
    (1): ReLU(inplace)
    (2): Dropout(p=0.5)
    (3): Linear(in_features=512, out_features=64, bias=True)
    (4): ReLU(inplace)
    (5): Dropout(p=0.5)
    (6): Linear(in_features=64, out_features=2, bias=True)
    (7): Sigmoid()
  )
)
```

Figure 5. Classifier CNN model

In addition to CNN classifier, in order to pretrain the classifier, a convolutional autoencoder was implemented which is shown in Figure 6.

```
AutoEncoderEEGDWT2D(
  (encoder): EncoderEEGDWT2D(
    (conv1): Sequential(
      (0): Conv2d(4, 16, kernel_size=(1, 5), stride=(1, 1), padding=(0, 2), bias=False)
      (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
    )
    (conv2): Sequential(
      (0): Conv2d(16, 32, kernel_size=(2, 1), stride=(1, 1), groups=16, bias=False)
      (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
      (2): ReLU(inplace)
    )
    (pool1): MaxPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0, dilation=1,
ceil_mode=False)
    (dropout1): Dropout2d(p=0.5)
    (sep_conv): Sequential(
      (0): Conv2d(32, 32, kernel_size=(1, 15), stride=(1, 1), padding=(0, 7), bias=False)
      (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
      (2): ReLU(inplace)
    )
    (pool2): MaxPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0, dilation=1,
ceil_mode=False)
    (dropout2): Dropout(p=0.5)
  )
  (decoder): DecoderEEGDWT2D(
    (deconv1): Sequential(
      (0): ConvTranspose2d(32, 32, kernel_size=(1, 15), stride=(1, 1), padding=(0, 7),
bias=False)
      (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
      (2): ReLU(inplace)
    )
    (unpool1): MaxUnpool2d(kernel_size=(1, 4), stride=(1, 4), padding=(0, 0))
    (dropout1): Dropout(p=0.5)
    (deconv2): Sequential(
      (0): ConvTranspose2d(32, 16, kernel_size=(2, 1), stride=(1, 1), groups=16,
bias=False)
      (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True
      (2): ReLU(inplace)
    )
    (unpool2): MaxUnpool2d(kernel_size=(1, 4), stride=(1, 4), padding=(0, 0))
    (dropout2): Dropout(p=0.5)
    (deconv3): Sequential(
      (0): ConvTranspose2d(16, 4, kernel_size=(1, 5), stride=(1, 1), padding=(0, 2),
bias=False)
      (1): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): Sigmoid()
    )
  )
)
```

Figure 6. Convolutional autoencoder used for pretraining

D. Training

There are 491 training data and 123 validation data. All the data were zero padded before inputting to any CNN model. Convolutional autoencoder and the classifier CNN were trained with 5000 epochs in total, batch size 16, optimized by Adam algorithm [10] with weight decay 1e-5 and binary cross entropy was the loss function for both training process.

After autoencoder was trained, we load the weight of the encoder with the lowest loss to the CNN part of the CNN classifier as the initial weight. When the weight was loaded, we may start to train our classifier.

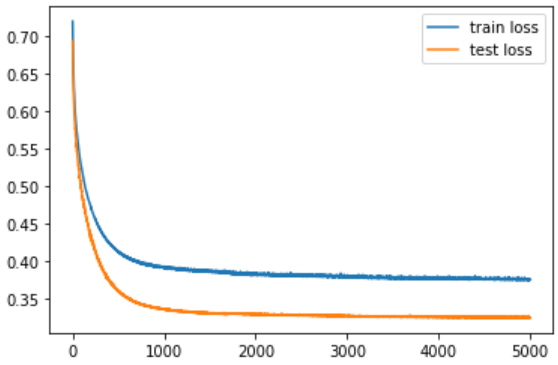


Figure 7. The loss curve of training the convolutional autoencoder

VII. Performance Evaluation

Table 1. Validated accuracy of classifiers with different preprocessing methods

Models	Valence Accuracy (%)	Arousal Accuracy (%)
Entropy-CNN with ICA	**80.48**	69.11
Energy-CNN with ICA	74.79	65.04
Entropy-CNN without ICA	76.42	**71.27**
Energy-CNN without ICA	66.67	69.92
Entropy-CNN without pretraining	61.78	56.91

We trained CNN using entropy or energy as features. In addition, we compared the results with or without applying ICA for removing ocular movement, and the model with or without using convolutional autoencoder as the pretraining method.

According to Table 1., using entropy as feature leads to better results than energy, and using convolutional autoencoder proved to be essential by increasing the accuracy up to 15%. However, removing ICA improved the accuracy of valence while dropped the accuracy of arousal. There are two factors that may cause the dropping accuracy. One of the factors is the decision of whether a channel contains ocular movement was done by a non-professional member, me, which could lead to errors. Another factor is that ocular movements contain information related to arousal. The influence is obvious when the experiment conductor stimulated HALV emotions using horror movies such as Silent Hill.

VIII. Conclusion

In this study, we proposed an emotion recognition model based on CNN. EEG signals from dataset were applied ICA to remove ocular movement effect and DWT to decompose processed EEG signals to gamma, beta, alpha and theta frequency bands. The entropy or energy features in temporal window of 4 seconds with overlapping 2 seconds were extracted as the input data of CNN. Afterwards, CNN was pretrained by constructing convolutional autoencoder and the weight of the encoder was extracted as the convolutional part of the classifier. Our classifier reached 80.48% accuracy of and 71.27%.

For future work, other entropy features such as multiscale entropy can be computed as features in temporal windows. ICA process for removing ocular movement effect can be improved by reduce human labor since it took three days to process all 614 data.

IX. Reference Paper

[1] O. Bazgir, Z. Mohammadi and H. A. H. Seyed, "Emotion Recognition with Machine Learning Using EEG signals," *Conference on Biomedical Engineering (ICBME). IEEE,* 2018.

[2] P. Keelawat, N. Thammasan, M. Numao and B. Kijsirkul, "Saptiotemporal Emotion Recognition using Deep CNN Based on EEG during Music Listening," *arXiv preprint arXiv,* 2019.

[3] L. Santamaria-Granados , M. Munoz-Organero , G. Ramirez-González , E. Abdulhay and N. ARUNKUMAR, "Using deep convolutional neural network for emotion detection on a physiological signals dataset (AMIGOS)," *IEEE Access,* pp. 57-67, 2018.

[4] J. A. Miranda-Correa, M. K. Abadi, N. Sebe and I. Patras, "AMIGOS: A Dataset for Affect, Personality and Mood Research on Individuals and Groups," *ArXiv e-prints,* Feb. 2017.

[5] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, "Automatic differentiation in PyTorch," *NIPS 2017 Workshop Autodiff Submission,* Oct. 2017.

[6] S. P. Lloyd, "Least squares quantization in PCM.," *information Theory, IEEE Transactions,* pp. 129-137, 1982.

[7] A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen and M. Hämäläinen, "MNE software for

processing MEG and EEG data," *NeuroImage,* pp. 446-460, Feb. 2014.

[8] A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, R. Goj, M. Jas, T. Brooks, L. Parkkonen and M. Hämäläinen, "MEG and EEG data analysis with MNE-Python," *Frontiers in Neuroscience,* 2013.

[9] V. J. Lawhern, A. J. Solon and N. R. Waytowich, "EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces.," *Journal of neural engineering,* 2018.

[10] J. B. Diederik P. Kingma, "Adam: A Method for Stochastic Optimization," *arXiv preprint,* 2014.

[11] N. Srivastava, G. Hinton, A. Krizhevsky , I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* pp. 1929-1958, 2014.