

HUMAN EMOTION DETECTION USING PHYSIOLOGICAL SIGNALS

A report submitted in partial fulfillment of the requirements Of

Mini Project

In
Sixth
Semester By

1MS18IS083	Saaniya Afreen
1MS18IS106	Subodh P Rane
1MS18IS109	Sukanya Singh

Under the guidance of

Dr Sumana M

Associate Professor

Dept. of ISE, RIT



RAMAIAH

Institute of Technology

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

RAMAIAH INSTITUTE OF TECHNOLOGY

(AUTONOMOUS INSTITUTE AFFILIATED TO VTU)

M. S. RAMAIAH NAGAR, M. S. R. I. T. POST, BANGALORE – 560054
2020-2021

Acknowledgement

We have taken extreme efforts in this project. However, it would not have been possible without the kind support and help of many individuals.

We would like to extend our sincere thanks to all of them. We are highly indebted to our college Principal Dr. N.V.R. Naidu and Head of Department Dr. Yogish H K for giving us a platform/opportunities to do this project as well as for constant supervision and timely support in completing the project .

We are highly indebted to Dr Sumana Maridithaya for her guidance and constant supervision as well as for providing necessary information regarding the project & also for her support in completing the project.

We would like to express our gratitude towards fellow mates and other faculties of Ramaiah Institute of Technology for their kind cooperation and encouragement which helped us in completion of this project.

Our thanks and appreciations also go to our colleagues in developing the project and people who have willingly helped us out with their abilities.

Abstract

Recommender systems have been based on context and content, and now the technological challenge of making personalized recommendations based on the user's emotional state arises through physiological signals that are obtained from devices or sensors. Novel trends in affective computing are based on reliable sources of physiological signals such as Electroencephalogram (EEG), Electrocardiogram (ECG), and Galvanic Skin Response (GSR). The use of these signals provides challenges of performance improvement within a broader set of emotion classes in a less constrained real-world environment. This study applies the deep learning approach using a deep convolutional neural network on a dataset of physiological signals (electrocardiogram and galvanic skin response), in this case, the AMIGOS dataset. The detection of emotions is done by correlating these physiological signals with the data of arousal and valence of this dataset, to classify the affective state of a person. In addition, an application for emotion recognition based on classic machine learning algorithms is proposed to extract the features of physiological signals in the domain of time, frequency, and nonlinear. The results outperform state-of-the-art approaches for classification into four classes, namely High Valence—High Arousal, High Valence—Low Arousal, Low Valence—High Arousal, and Low Valence—Low Arousal. This application uses a convolutional neural network for the automatic feature extraction of the physiological signals, and through fully connected network layers, the emotion prediction is made. The experimental results on the AMIGOS dataset show that the method proposed in this project achieves a better precision of the classification of the emotional states, in comparison with the originally obtained by the authors of this dataset.

Contents

1. Introduction	4
• Motivation	
• Scope	
• Objectives	
• Proposed model	
• Block Diagram of Proposed Model	
• Organisation of report	
2. Literature review	6
3. Problem statement	14
4. System analysis and design	15
• Use Case Diagram	
5. Modelling	17
• Valence-arousal model	
• Basic architecture of our model	
6. Implementation	18
• self-assessment score plotted in valence-arousal space	
• K-means clustering graph	
• ICA preprocessing	
• DWT decomposition	
7. Results and discussion	24
8. Testing	25
9. Conclusion	29

Introduction

Motivation:

Emotions states are highly correlated with human behaviors and thoughts. Thus, study on emotion recognition using physiological 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.

Scope:

Emotion plays an important role in our daily life and work. Real-time assessment and regulation of emotion will improve people's life and make it better. For example, in the communication of human-machine-interaction, emotion recognition will make the process more easy and natural. Another example, in the treatment of patients, especially those with expression problems, the real emotion state of patients will help doctors to provide more appropriate medical care. In recent years, emotion recognition from physiological signals has gained mass attention.

Proposed Model:

In this study, we proposed an 2D deep CNN based model extracting spatial and temporal features for emotion recognition on physiological signal dataset, AMIGOS , with 71.27% and 80.48% accuracy on arousal and valence classification. 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 the final stage.

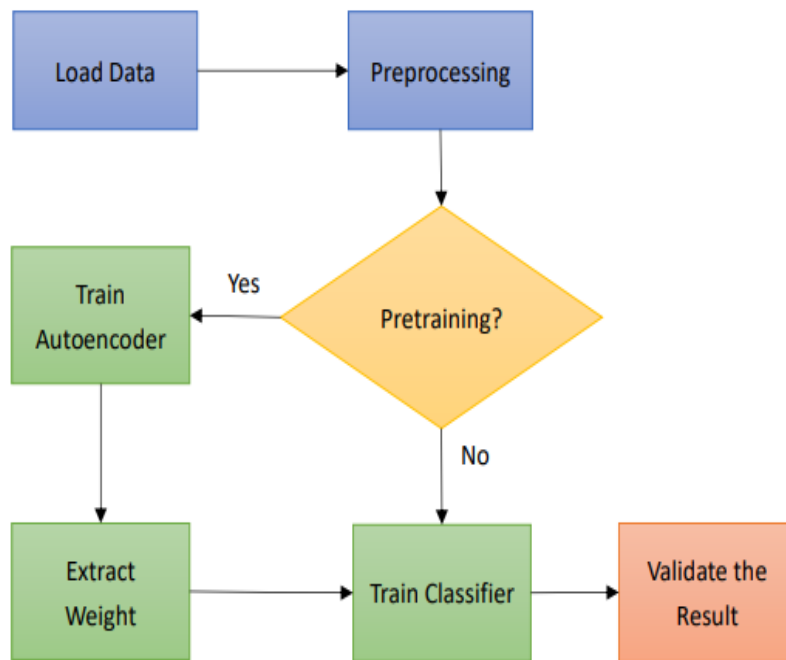


Figure 1:Block Diagram of Proposed Model

Organisation of the report :

In order to explain the developed system, the following sections are covered:

- Literature Review describes the study of the existing systems and techniques taken into account prior to development of the proposed system.
- System Analysis and Design provides a detailed walk through of the software engineering methodology adopted to implement the model, an overview of the system and the modules incorporated into the system
- Modelling and Implementation provides a deeper insight into the working of the model. The various modules and their interactions are depicted using relevant descriptive diagrams.
- Testing the model to ensure bug/error free model along with the Results obtained. Discussion then provides detailed analysis on quality assurance measures.
- Conclusion about the Results obtained after successfully running the model and Future Scope of the model is highlighted.

Literature review

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 physiological 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]

A. Multimodal Dataset

[3] Emotion is the degree to which a person reacts to changes in the context as a response to the elicitation that manifests itself in their affective states[13] . People use the senses to express the emotion experienced through gestures, speech or physiological responses. The correlation between emotions and physiological data determines the multimodal affect recognition. The contents of images , movie clips and music videos have been used to induce emotions that users appraise with explicit measurements , in order to verify the arousal and valence levels. On the other hand, emotions elicited by multimedia content are implicitly recognized by means of physiological and brain signals, enabling the consolidation of a multimodal affective dataset that compares the affective response of people[13].

Precisely, the dataset ASCERTAIN[14] affects the personality and emotion recognition induced by 36 movie clips that have a duration of 58 to 128 seconds, with the registration of physiological signals (ECG and GSR), EEG and activity facial of 58 participants. AMIGOS dataset detects the mood, affect and personality of 40 participants with the registration of their EEG, ECG, and GSR signals, as a result of the stimulus caused during the viewing of short and long videos.

Abadi *et al.* for the affect detection analyzes the physiological response of the ECG, Electrooculogram (EOG) and trapezius-Electromyogram (EMG), and contrasts the brain signals (EEG and Magnetoencephalogram) of 30 participants who watched 36 movie clips from 80 seconds and 40 segments of one-minute music videos that are part of the DEAP dataset. In the emotional state's recognition of 32 participants, DEAP includes physiological signals (GSR, BVP, SKT, EOG, and EMG) and EEG. Similarly, the multimodal database MAHNOB-HCI contains physiological signals (ECG, GSR, SKT, and Respiration), eye gaze and EEG from 27 participants, who evaluated the emotion through various stimuli (20 emotional videos, 14 short videos, and 28 images).

Both DEAP and MANNOB-HCI demonstrate better EEG effectiveness in predicting arousal and physiological signals obtained a better outcome with valence. AMIGOS has the same behavior with EEG signals, but unlike and , it obtained better f1-score outcome with arousal. The physiological features in DECAF had a better arousal recognition in the movie clips and a better valence outcome in the music clips. In ASCERTAIN the multimodal results (ECG and GSR) had a better performance in contrast to the EEG.

B. Emotional States Detection

The publications related to the affective recognition from physiological data have the purpose of constructing reliable models supported by techniques and machine learning algorithms, to discover patterns of the emotional states that are hidden in the physiological signals. Various methodologies have been explored for the preprocessing of data, the extraction, and selection of physiological features, as stages prior to the classification of emotion.

Some studies for the affect recognition of have implemented supervised classification approaches such as k-Nearest Neighbor (k-NN) , and Support Vector Machine (SVM). The researchers defined keywords to validate the user's emotional responses through the valence and excitement model. The physiological signals are processed by sliding window technique and the process of reducing the dimensionality of the features is based on the Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) techniques.

On the other hand, the Deep Learning approach applies non-linear transformations to physiological signals for the detection of features of human emotional behavior. In this context, CNN techniques have been used for the automatic extraction of SCR and BVP features and 70 to 75% accuracy results have been obtained in the prediction of emotion (relaxation, anxiety, excitement, and fun).

Regarding to semi-supervised learning methodologies SAE was integrated with Deep Belief Network (DBN) using a Bayesian inference classification based decision fusion method , results of arousal were obtained in 73.1% and valence in 78.8%. In they defined a hybrid model composed of a CNN and a Recurrent Neural Network (RNN). As a requirement for the sequential processing in the CNN, the features were extracted and the prediction was made in the Long Short-Term Memory (LSTM) unit of the RNN. This model obtained an accuracy of 74.1% for arousal and 72.1% for valence. The models based on CCN and DNN showed better results in the affective classification when using the image domain of the EEG signals[15].

The related works deal with the trend of deep learning for the emotions detection related to heart disease, mental disorder, and stress. However, to validate the affective models there is a limitation in the access to small physiological datasets or there is a problem in obtaining correct data. Therefore, it is necessary to publish repositories of physiological datasets, so that researchers can test the classification models that can be used in the personalization of tourist services or any domain.

[6] Clustering analysis method is one of the main analytical methods in data mining, the method of clustering algorithm will influence the clustering results directly. This paper discusses the standard k-means clustering algorithm and analyzes the shortcomings of standard k-means algorithm, such as the k-means clustering algorithm has to calculate the distance between each data object and all cluster centers in each iteration, which makes the efficiency of clustering is not high. This paper proposes an improved k-means algorithm in order to solve this question, requiring a simple data structure to store some information in every iteration, which is to be used in the next iteration. The improved method avoids computing the distance of each data object to the cluster centers repeatedly, saving the running time. Experimental results show that the improved method can effectively improve

the speed of clustering and accuracy, reducing the computational complexity of the k-means.

[7][8] Magnetoencephalography and electroencephalography (M/EEG) measure the weak electromagnetic signals originating from neural currents in the brain. Using these signals to characterize and locate brain activity is a challenging task, as evidenced by several decades of methodological contributions. MNE, whose name stems from its capability to compute cortically-constrained minimum-norm current estimates from M/EEG data, is a software package that provides comprehensive analysis tools and workflows including preprocessing, source estimation, time–frequency analysis, statistical analysis, and several methods to estimate functional connectivity between distributed brain regions. The present paper gives detailed information about the MNE package and describes typical use cases while also warning about potential caveats in analysis. The MNE package is a collaborative effort of multiple institutes striving to implement and share best methods and to facilitate distribution of analysis pipelines to advance reproducibility of research.

[9] Convolutional Neural Networks (CNNs), which have been used in computer vision and speech recognition, have successfully been applied to EEG-based BCIs; however, they have mainly been applied to single BCI paradigms and thus it remains unclear how these architectures generalize to other paradigms. Here, we ask if we can design a single CNN architecture to accurately classify EEG signals from different BCI paradigms, while simultaneously being as compact as possible. In this work we introduce EEGNet, a compact convolutional network for EEG-based BCIs. We introduce the use of depthwise and separable convolutions to construct an EEG-specific model which encapsulates well-known EEG feature extraction concepts for BCI. We compare EEGNet to current state-of-the-art approaches across four BCI paradigms: P300 visual-evoked potentials, error-related negativity responses (ERN), movement-related cortical potentials (MRCP), and sensory motor rhythms (SMR). We show that EEGNet generalizes across paradigms better than the reference algorithms when only limited training data is available. We demonstrate three different approaches to visualize the contents of a trained EEGNet model to enable interpretation of the learned features. Our results suggest that EEGNet is robust enough to learn a wide variety of interpretable features over a range of BCI tasks, suggesting that the observed performances were not due to artifact or noise sources in the data.

[10] We introduce Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyper-parameters have intuitive interpretations and typically require little tuning. Some connections to related algorithms, on which Adam was inspired, are discussed. We also analyze the theoretical convergence properties of the algorithm and provide a regret bound on the convergence rate that is comparable to the best known results under the online convex optimization framework. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods. Finally, we discuss AdaMax, a variant of Adam based on the infinity norm.

Problem statement

Human emotion detection using physiological signals (EEG, ECG and GSR). In this study, we developed an emotion recognition system based on the valence arousal model. Emotions play an important role in everyone's life. The brain waves can tell us the difference in the emotions the person is going through and hence help in emotion detection.

System analysis

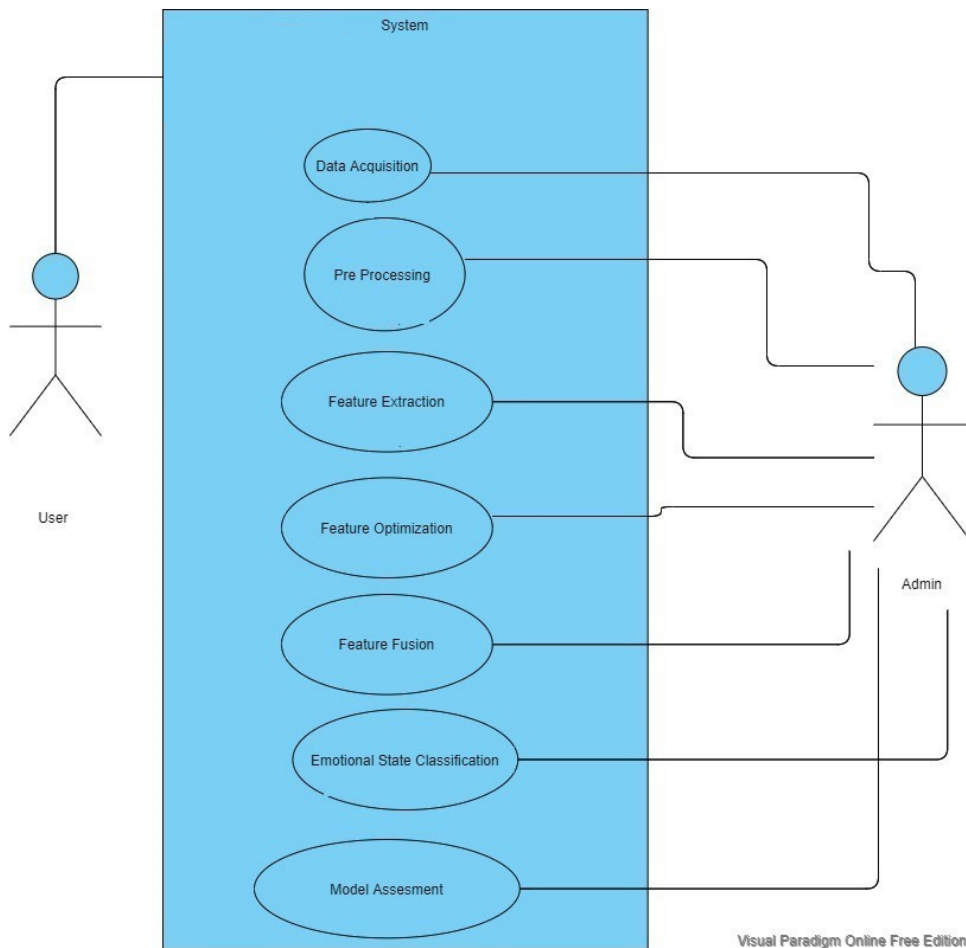


Figure 2 : Use Case Diagram

Figure 2 above explains how the physiological signals are obtained from the user and used to train and test our model.

- The data set is obtained from the author of the AMIGOS data set.
- After obtaining, we convert it to csv and drop the NaN values.
- Features like ocular movements and alpha,beta,theta and gamma bands are extracted.
- After removal of ocular movement, our data is well optimised.
- The data is finally ready for model assessment.
- A model is created for training and testing the data.

Tools and technologies used:

- python == 3.7.4
- numpy == 1.16.5
- pandas == 0.25.1
- pytorch == 1.0.0
- matplotlib == 3.1.2
- pickle == 4.0
- mne == 0.19.2
- scipy == 1.3.1
- sklearn == 0.20.0
- pywt == 1.0.3

MODULES :

The proposed work is classified into 4 modules :

- **Load Data**

Load Data is the process of copying and loading data or data sets from a source file, folder or application to a database or similar application. It is usually implemented by copying digital data from a source and pasting or loading the data to a data storage or processing utility.

- **Preprocess Data**

Data preprocessing refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training models. It transforms raw data into an understandable and readable format.

- **Train Data**

The training data is an initial set of data used to help a program understand how to apply technologies like neural networks to learn and produce sophisticated results. It may be complemented by subsequent sets of data called validation and testing sets.

- **Validate the results**

Validation of results is an important phase of search. Some of the overall goals of this phase are the following. Ensure in a cost-effective way whether a set of searches performed are satisfying a production request.

Modelling

Our project is based on valence-arousal model. This model suggests that emotions are distributed in a two-dimensional circular space, containing arousal and valence dimensions. Arousal represents the vertical axis and valence represents the horizontal axis, while the center of the circle represents a neutral valence and a medium level of arousal. In this model, emotional states can be represented at any level of valence and arousal, or at a neutral level of one or both of these factors.

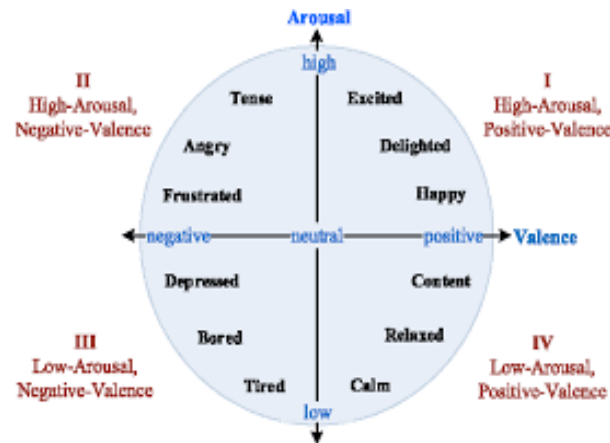


Figure 3 : Valence-arousal model

BASIC ARCHITECTURE

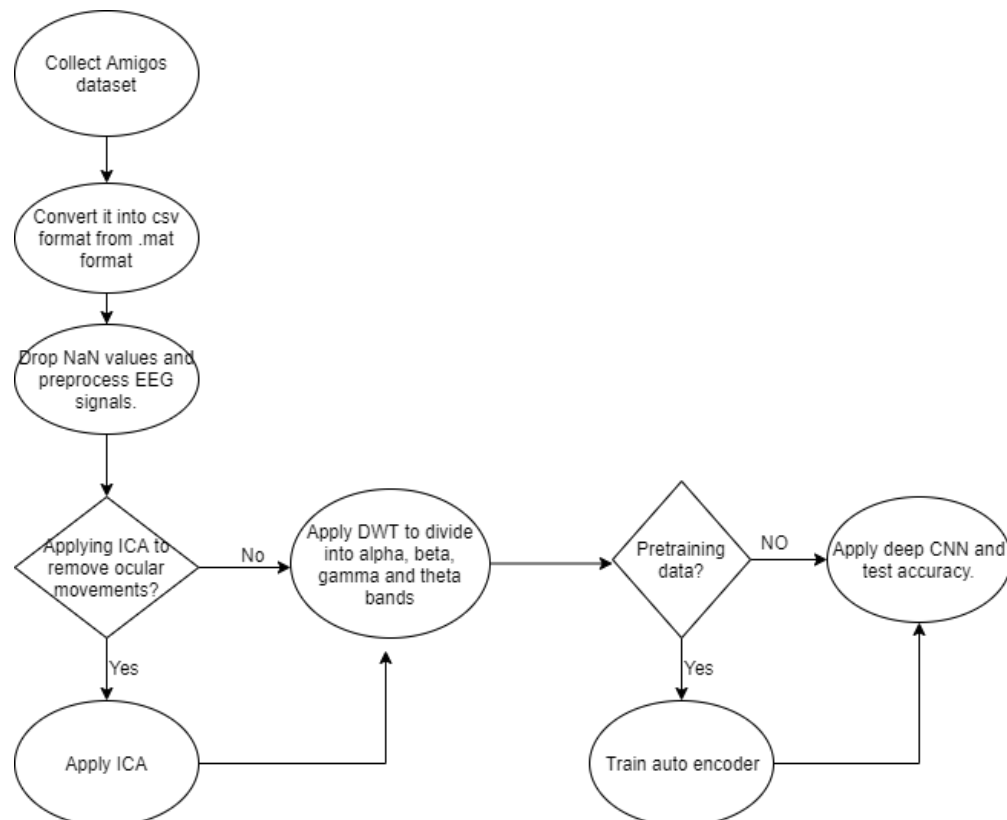


Fig 4: Basic architecture of our model.

The above figure shows the basic architecture which is followed in the project. We used CNN to train our model and tested the accuracy of it.

Auto-encoder:

Autoencoder is an unsupervised algorithm based on the BP algorithm, which contains an input layer, one or more hidden layers and an output layer (as can be seen in Figure 11b). The dimension of the input layer is equal to that of the output layer, so that it was called 'encoder network' (EN) from input layer to the hidden layer and 'decoder network' (DN) from hidden network to output layer. The autoencoder works as below: At first the weights of the EN and the DN are initiated. Then the autoencoder is trained according to the principle that minimizes the error between the original data and the reconstructed data. It is easy to get the desired gradient value by passing the chaining method of the DN and passing the EN using the backward propagation error derivation and adjusting the weighted value of the autoencoder to the optimal one.

Deep CNN:

Convolutional Neural Networks (CNNs), a class of deep, feed-forward artificial neural networks based on their shared-weights architecture and translation invariance characteristics, have achieved great success in the image domain. They have been introduced to process physiological signals, such as EEG, EMG and ECG in recent years. Martinez et al. trained an efficient deep convolutional neural network (CNN) to classify four cognitive states (relaxation, anxiety, excitement and fun) using skin conductance and blood volume pulse signals.

Implementation

Load Data:

We use AMIGOS as our training and validation dataset. This dataset has two experiments. First, 40 participants watched 16 short videos which were less than 250 seconds individually. Second, five groups of four participants and 17 people individually watched four long movies. After both of the experiments, self-assessment including valence and arousal scaled from 1 to 9 were applied. In our experiment, we used EEG data recorded during short video experiments. The EEG data were preprocessed with a sampling frequency of 128Hz. Because pytorch was used to construct deep CNN models and the original data from AMIGOS dataset was stored in matlab file, we used `scipy.io.loadmat` function to load matlab file to python object.

```
def LoadMatData(path):
    mats=[]
    for i in range(40):
        if i+1 < 10:
            dataName="Data_Preprocessed_P0"+str(i+1)
            src=path+dataName+"/" +dataName+".mat"
            mat=scipy.io.loadmat(src)
            mats.append(mat)
        else:
            dataName="Data_Preprocessed_P"+str(i+1)
            src=path+dataName+"/" +dataName+".mat"
            mat=scipy.io.loadmat(src)
            mats.append(mat)
    return mats

#organising data
def OrganisingData(mats):
    data_list=[]
    selfassessment_list=[]

    for mat in mats:
        #short videos only
        data=mat['joined_data'][0][0:16]

        #Instaces * Channel -> Channel * Instances
        for i in range(16):
            data[i]=np.transpose(data[i],(1,0))
        data_list.append(data)
```

Fig 5 : Loading data and converting it into csv

Pre-processing:

First, we needed to determine the classification ground truth for each physiological 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 a value equal to 5. In this case, K-means[6] was applied to determine which cluster this instance belongs to.

After determining the ground truth for each instance, We could start to process the input physiological signals. First, we needed to remove NaN instances in the AMIGOS dataset. After the removal 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 physiological signals. Thus, ICA was applied to find the ocular movement component and drop the signal. In our implementation, mne library was used in removing ocular movement.

```
def PreprocessData():
    with open("./data_list.pkl", "rb") as file:
        data_list=pickle.load(file)

    EEG_list=list()

    #removing first 3secs
    for tester in data_list:
        for instance in tester:
            EEG_list.append(instance[0:14][:,384:])

    #record the indecies of nan array
    nan_set=set()

    #Preprocess EEG
    for i in range(len(EEG_list)):
        if(np.all(np.isnan(EEG_list[i]))):
            nan_set.add(i)
            EEG_list[i]=np.nan

    #Remove the data with Nan values
    EEG_list=np.delete(EEG_list,list(nan_set)).tolist()
    pd.read_csv("./groundtruth.csv").drop(list(nan_set)).to_csv("./groundtruth_dropnan_EEG.csv", index=False)
    return EEG_list
```

Fig 6 : dropping the first 3 seconds and dropping NaN values.

```

for eeg in EEG_list:
    info = mne.create_info(14, 128, ch_types=['eeg']*14)
    eeg = scaler.fit_transform(eeg.T).T * 10e-5

    raw = mne.io.RawArray(eeg, info)
    raw_tmp = raw.copy()
    raw_tmp.filter(1, None)

    ica=mne.preprocessing.ICA(method="infomax",
                              fit_params={"extended": True},
                              random_state=1)

    ica.fit(raw_tmp)
    ica.plot_sources(inst=raw_tmp,start=0, stop=40)

    raw_corrected=raw.copy()
    hannel_exclude=list(map(int,
                           input("Input the channel index you want to exclude: ").split()))
    #channel_exclude = [1]
    #print(channel_exclude)

    ica.exclude=channel_exclude
    ica.apply(raw_corrected)
    raw_corrected.plot(n_channels=14, start=0, duration=40)

    eeg_corrected=scaler.fit_transform(raw_corrected[:,0].T).T

```

Fig 7 : Applying ICA to remove ocular movements.

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

```
if feature == "entropy":
    for i in range((samples), (n-samples), samples):
        lowpass_signal, noise=pywt.dwt(signal[:,i-samples:i+samples], 'db4')
        for j in range(4):
            lowpass_signal, highpass_signal=pywt.dwt(lowpass_signal, 'db4')
            highpass_signal=scaler.fit_transform(highpass_signal.T).T
            if j==0:
                if i == samples:
                    gamma=entropy(highpass_signal.T).reshape(1,-1).T
                else:
                    gamma=np.concatenate((gamma,
                                           entropy(highpass_signal.T).reshape(1,-1).T), axis=1)

            if j==1:
                if i == samples:
                    beta=entropy(highpass_signal.T).reshape(1,-1).T
                else:
                    beta=np.concatenate((beta,
                                         entropy(highpass_signal.T).reshape(1,-1).T), axis=1)

            if j==2:
                if i == samples:
                    alpha=entropy(highpass_signal.T).reshape(1,-1).T
                else:
                    alpha=np.concatenate((alpha,
                                           entropy(highpass_signal.T).reshape(1,-1).T), axis=1)
```

Fig 8 : Applying DWT to decompose signals.

Models:

Our deep CNN model was based on EEG Net , 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 9: Classifier CNN model

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

```

def __init__(self):
    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 10:Convolutional autoencoder used for pretraining

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 with weight decay $1e-5$ and binary cross entropy was the loss function for both training processes.

After the 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 is loaded, we may start to train our classifier.

```
for epoch in range(epochs):
    model.train()
    total_loss=0
    for step, (batch_x, batch_y) in enumerate(train_loader):
        X_hat=model(batch_x)
        loss = loss_fct(X_hat, batch_y)
        total_loss += loss.item() * batch_x.size(0)

        loss.backward()

        optimizer.step()
        optimizer.zero_grad()

    train_loss.append(total_loss/len(train_loader.dataset))
    test_loss.append(TestAutoEncoder(test_loader, model, loss_fct).item())

    if epoch != 0 and min(train_loss) == (total_loss/len(train_loader.dataset)):
        torch.save(model.state_dict(), './AutoEncoder_lowestloss_ica_energy')

    if (epoch+1) % 10 == 0:
        print("Epoch: ", epoch+1, "| Loss: ", train_loss[-1])
        print("Test Loss: ", test_loss[-1])

    if (epoch+1) % 100 ==0:
        torch.save(model.state_dict(), './AutoEncoder_'+str(epoch+1)+"_ica_energy")
return train_loss, test_loss
```

Fig 11 : Training autoencoder.

Testing and Results

All the processes were tested thoroughly and the outputs and results were noted.

Loading the dataset

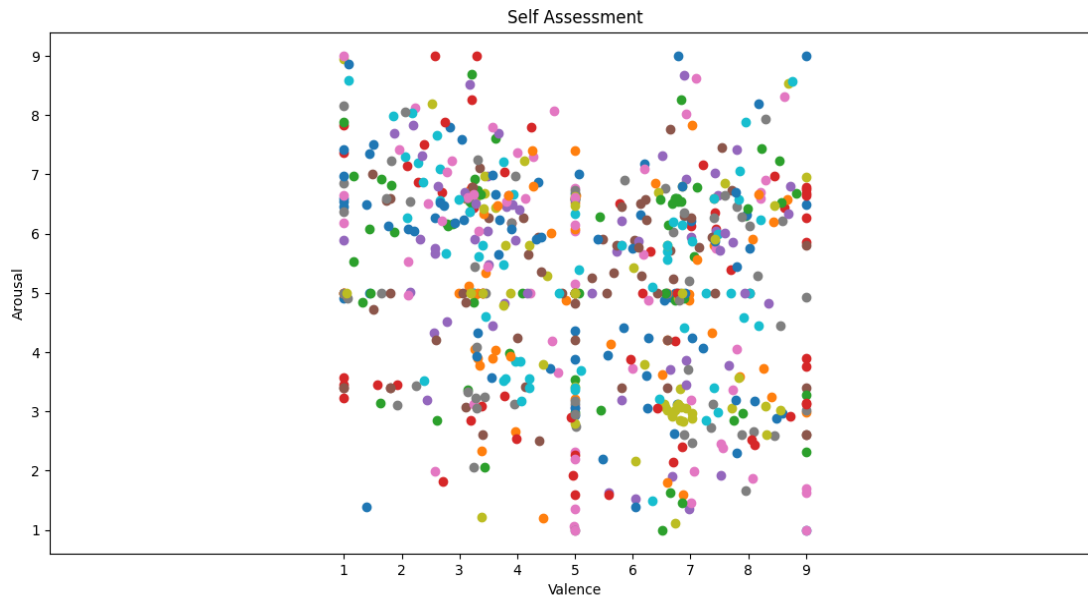


Figure 12:The self-assessment score plotted in valence-arousal space, the color represents emotion states that experiment conductors of AMIGOS dataset wanted to stimulate.

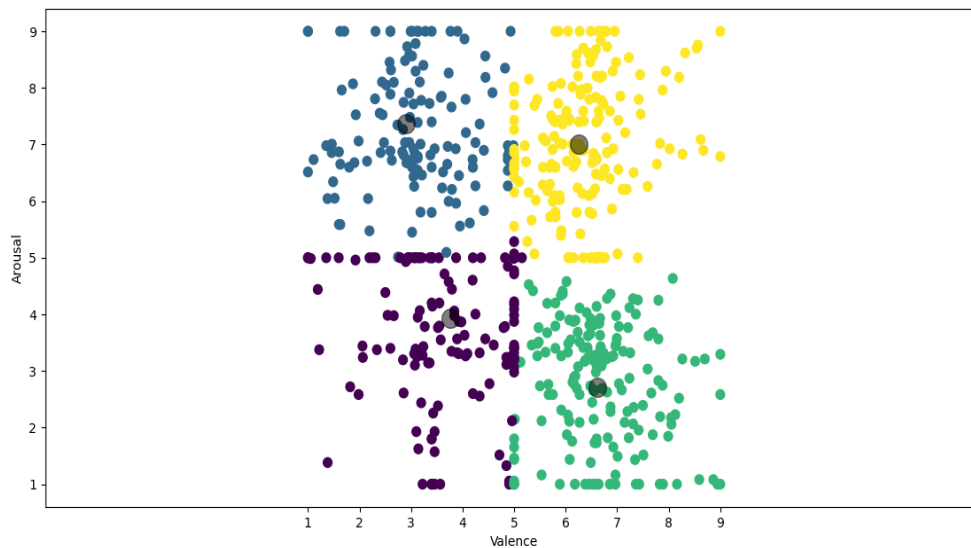


Figure 13: The ground truth label after K-means clustering

Preprocessing

ICA preprocessing -

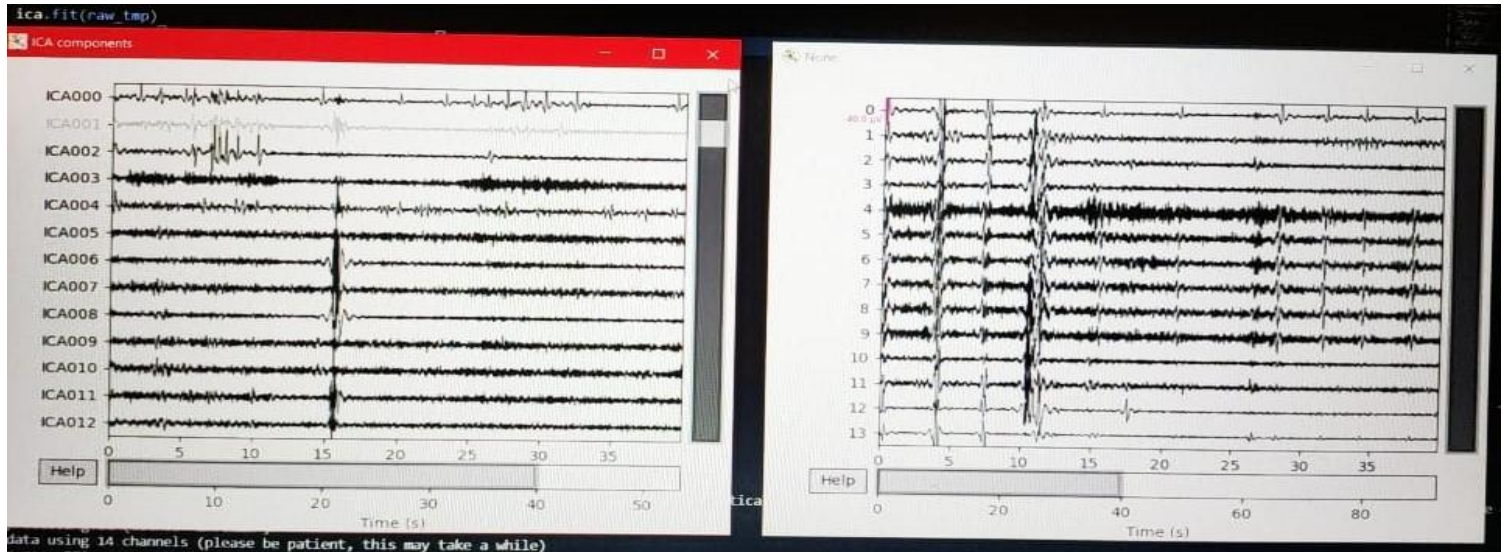


Figure 14 (a) EEG signals after removing component in ICA channel 0 [7] [8] (b) The signal I regarded as ocular movement signal

DWT decomposition into different bands -

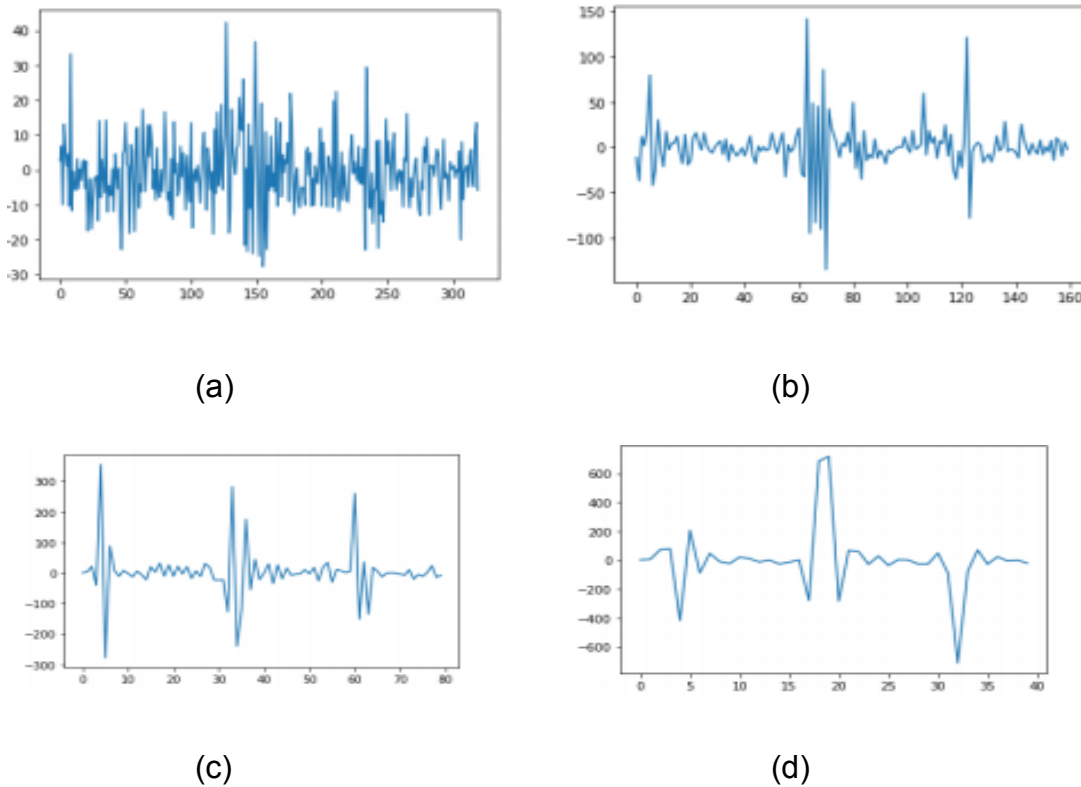
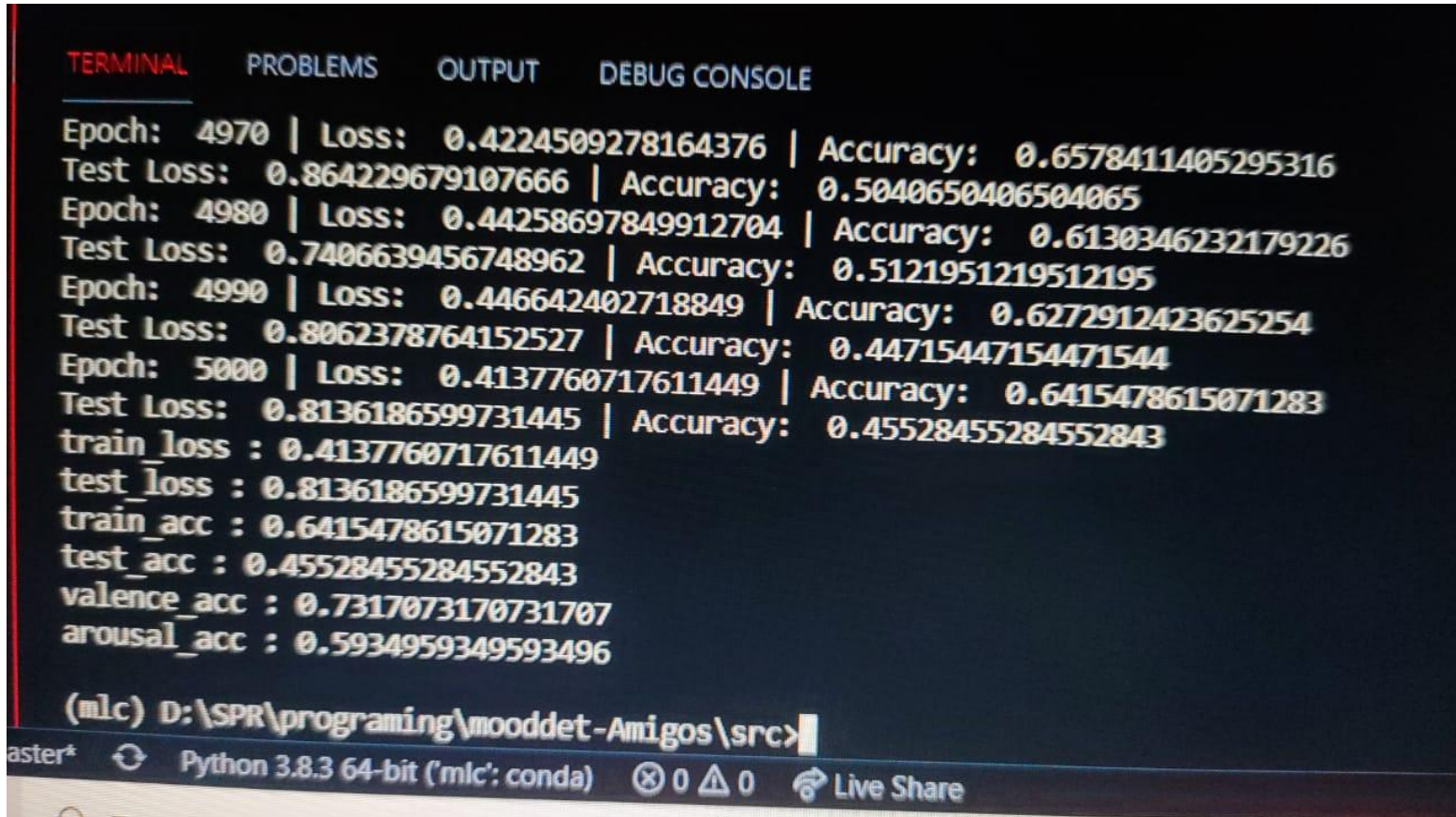


Figure 15. (a) Gamma band signal (b) Beta band signal (c) Alpha band signal (d) theta band signal

Final result

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.



```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE
Epoch: 4970 | Loss: 0.4224509278164376 | Accuracy: 0.6578411405295316
Test Loss: 0.864229679107666 | Accuracy: 0.5040650406504065
Epoch: 4980 | Loss: 0.44258697849912704 | Accuracy: 0.6130346232179226
Test Loss: 0.7406639456748962 | Accuracy: 0.5121951219512195
Epoch: 4990 | Loss: 0.446642402718849 | Accuracy: 0.6272912423625254
Test Loss: 0.8062378764152527 | Accuracy: 0.44715447154471544
Epoch: 5000 | Loss: 0.4137760717611449 | Accuracy: 0.6415478615071283
Test Loss: 0.8136186599731445 | Accuracy: 0.45528455284552843
train_loss : 0.4137760717611449
test_loss : 0.8136186599731445
train_acc : 0.6415478615071283
test_acc : 0.45528455284552843
valence_acc : 0.7317073170731707
arousal_acc : 0.5934959349593496

(mlc) D:\SPR\programing\mooddet-Amigos\src>
```

Figure 16: Result of CNN without ICA

Table 1. Validated accuracy of classifiers with different preprocessing methods

Models	Valence Accuracy (%)	Arousal Accuracy (%)
CNN with ICA	80.48	69.11
CNN without ICA	76.42	71.27
CNN without pretraining	61.78	56.91

According to Table 1., removing ICA improved the accuracy of valence while dropping 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.

Conclusion and Future Work

In this project, we proposed an emotion recognition model based on CNN. EEG signals from the 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 features in a temporal window of 4 seconds with overlapping 2 seconds were extracted as the input data of CNN. Afterwards, CNN was pre trained 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 valence and 71.27% of arousal.

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

References

- [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] Shi Na , Liu Xumin, Guan Yong, Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm
- [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.
- [12] A. Greco, G. Valenza and E. P. Scilingo, *Advances in Electrodermal Activity Processing with Applications for Mental Health: From Heuristic Methods to Convex Optimization*, Springer, 2016.
- [13] S. Katsigiannis and N. Ramzan, "DREAMER: A database for emotion recognition through EEG and ECG signals from wireless low-cost off-the-shelf devices", *IEEE J. Biomed. Health Informat.*, vol. 22, no. 1, pp. 98-107, Jan. 2018.
- [14] R. Subramanian, J. Wache, M. K. Abadi, R. L. Vieriu, S. Winkler and N. Sebe, "ASCERTAIN: Emotion and personality recognition using commercial sensors", *IEEE Trans. Affect. Comput.*, vol. 9, no. 2, pp. 147-160, Apr./Jun. 2018.
- [15] L. Wenqian, C. Li and S. Sun, "Deep convolutional neural network for emotion recognition using EEG and peripheral physiological signal"