System Identification Term Project

2102531 Semester 1/64 Year 2021

EEG-based valence and arousal estimation using wavelet scalogram and convolutional networks

Napat Samsow ID 6470173021

Totok Nugroho ID 6470032121

Advisor: Jitkomut Songsiri

Department of Electrical Engineering, Faculty of Engineering

Chulalongkorn University

November 29, 2021

Contents

1 Introduction

Emotion plays an important role in human lives and work because it is involved in a plethora of cognitive processes such as decision-making, perception, social interactions and intelligence [AF19]. Emotion recognition has become a prominent research area, because it is helpful to the diagnosis depression, schizophrenia, and other mental diseases. Emotion recognition can also bring human satisfactory user human-computer interaction experience [LZLK21].

A person's emotion state may become apparent by subjective experiences (person's feeling), behavioural, and physiological signal. Subjective self-reports about how the person is feeling can provide valuable information but there are issues with validity because participants may answer how they feel others would answer instead of how exactly they are feeling [AF19]. In line with subjective self-report, behavioural or non-physiological signal such as facial expressions, actions, voices, etc also cannot accurately identify human emotion in disguise. In contrast, physiological signal such as Electrocardiogram (ECG), Electromyogram (EMG), Galvanic Skin Response (GSR), Heart Rate Variability (HRV), and particularly Electroencephalogram (EEG) [GAF21] has better objectivity and difficult to hide in order to recognise emotion state [WW21].

Emotion can be represented in two different perspectives, which are discrete and dimensional scheme. In the discrete perspective, as proposed by Plutchik [GAF21], there are eight basic emotions: anger, fear, sadness, disgust, surprise, curiosity, acceptance, and joy. All the other emotions can be formed by those basic ones. In the other perspective, dimensional scheme, the most popular is the circumplex model of affect, which uses valence as abscissa and arousal as ordinate. Valence ranges from positive feelings (pleasure) to negative feelings (unpleasure) and arousal ranges from sleepy to excited [AF19].

Among other physiological signals, EEG signals are spontaneous and can show specific states of the brain that provide information about person's emotion [LZLK21]. There are two main areas of the brain correlated with emotional activity which are the amygdala (located near to the hippocampus, in the frontal part of the temporal lobe) and the pre-frontal cortex. Pre-frontal and parietal asymmetry in the alpha band and temporal asymmetry in gamma band are present for valence recognition, while pre-frontal asymmetry in alpha band and temporal asymmetry in the gamma band are observable for arousal recognition [AF19].

EEG is the electrical signal recording brain activity that is represented as voltage fluctuations resulting from ionic current flows within the neurons of the brain [AZAA15]. It has electrical amplitude ranging from 10 to 100 μV [AGA$^+$04] and frequency from 0.5 Hz to 40 Hz [CLR14]. According to the difference of frequency band, EEG signal can be divided into five types, delta (0.5 – 4 Hz), theta (4 – 8 Hz), alpha (8 – 13 Hz), beta (13 – 30 Hz), and gamma (more than 30 Hz) [CLR14]. Studies showed that alpha, beta, and gamma wave are more reliable [WW21] and most discriminative [AF19] in order to recognize emotion state.

Previous works in emotion recognition based on EEG signals identify emotion in only several state in dimensional or discrete perspective. For instance, emotions are differentiated as high/low valence, high/low arousal, or in quadrants of circumplex model of affect (HVHA, LVHA, LVLA, and HVLA where H, L, V, A stand for High, Low, Valence, and Arousal respectively). However, there has been only little attention on estimating exact value of valence and arousal [GAF21]. Studies that estimate exact value of valence and arousal use hand-extracted feature of EEG or traditional method such as statistical, Hilbert–Huang spectrum, differential entropy, approximate entropy, sample entropy, wavelet entropy, wavelet energy, Hjorth parameters, IMF energy and entropy [GAF21], power spectral density [KMS$^+$11].

Features used for emotion recognition can be categorized into two main types: traditional features which should be designed using prior knowledge and deep features which can be automatically learned from data. These features can be categorized into three main types: time-domain, frequency-domain, and time-frequency domain features. Time-domain features can be extracted from the time-domain information of the signal. The commonly used time-domain features are Hjorth parameters [Hjo70], which describe the statistical properties of the signal obtained by applying signal processing techniques in the time domain, and statistical measures such as mean, variance, power, and kurtosis of the signal. [AACP07] used synchronization measures, Hjorth parameters, and fractal dimension as features. These features were then used for classification using linear discriminant analysis (LDA). [PH09] proposed higher-order crossing features and evaluated them with four classifiers: quadratic discriminant analysis (QDA), k-nearest neighbors (k-NN), Mahalanobis distance, and support vector machine (SVM).

Frequency-domain features are also important for emotion recognition as the features extracted from different frequency bands have different degrees of relation to valence and arousal. [ZCZ+16] used powers in every specific frequency band as features for SVM to classify emotion. They also applied the ReliefF-based method to select the most few important channels. Many recent studies considered both time- and frequency-domain features simultaneously. [JYBM21] mainly used statistics of signal, Hjorth parameters, higher-order crossings, band power spectrum features, Hilbert-Huang spectrum (HHS), discrete wavelet transform (DWT), as well as features calculated from combinations of electrodes. These features were then selected using a regression-based Relief filter and used to estimate continuous values of valence and arousal using linear regression (LR), support vector regression (SVR), and multilayer perceptron (MLP). The results showed that MLP outperforms LR and SVR. [GAF21] used Hjorth parameters, spectral entropy, wavelet energy and entropy, and intrinsic mode function (IMF) energy and entropy as features. Continuous values of valence and arousal were then estimated using seven regressors: LR, additive regression, decision tree, k-NN, random forest (RF), and SVR with two kernels. However, the use of only frequency-domain features leads to a lack of time-domain information, which is also useful for emotion recognition.

Time-frequency domain is a representation of the signal frequency with localization in time. Due to the non-stationary properties of the EEG signal, the time-frequency domain is useful because both time- and frequency-domain information is considered simultaneously. [LL18] used CNN to extract features from spectrogram obtained by short-time Fourier transform (STFT) and then emotions were classified using MLP. [Mur11] used four different wavelets, namely, db4, db8, sym8, and coif5 to extract features. The emotions were then classified using k-NN. The spectrogram produced by STFT has a uniform length of time window over the signal. In contrast, continuous wavelet transform (CWT) has multiple temporal resolutions depending on the frequency and the choice of the mother wavelet. The CWT has a higher temporal resolution at a higher frequency and a higher frequency resolution at a lower frequency. [GAF21, ZCZ+16] showed that higher frequency bands are more useful for emotion recognition than lower ones. This shows the advantages of the CWT over STFT in the task of emotion recognition. There are three commonly used mother wavelets for CWT: Morse, Morlet, and Bump wavelets. The Bump wavelet has a lower resolution than the other two [BMSM]. Morse and Morlet wavelets have no considerable difference but the first one is more generalized, i.e., its parameters can be chosen [BMSM]. In addition, [PS21] showed that the Morse wavelet outperforms the other two.

In this paper, we conduct a valence and arousal estimation using EEG signals. The preprocessed EEG signals are transformed into wavelet scalograms using CWT with Morse wavelet. Convolutional neural network (CNN) automatically extracts features from the scalogram. The features are then used for regression using multilayer perceptron (MLP). We evaluate our method on the DEAP dataset using root-mean-square error (RMSE) and the sign agreement metric (SAGR). We investigate the effect on the performance when varying the number of layers in the networks. Our main contributions are:

- We propose a wavelet CNN-based valence and arousal estimation method, which is the first deep learning method for estimating continuous values of valence and arousal.

- We also investigate the effect on the performance when varying the number of layers in the networks based on ResNet architecture.

Project description The main objectives are to design a deep learning framework for estimating valence and arousal and study the effect on the performance when varying the number of layers. Scope is to use wavelet-based CNN as feature extractor, multilayer perceptron (MLP) as the regression model, and use the dataset from [KMS+11].

2 System modelling

In this study, we use 32-channel EEG signal data to estimate valence and arousal. Time-frequency representation of wavelet or wavelet scalogram of the EEG produced based on the Morse wavelet is used as input of CNN. The CNN then extracts feature from the scalogram. This feature is used as input for regression model to estimate valence and arousal. In this section, we describes convolutional neural network and multilayer perceptron.

2.1 Convolutional neural network

Convolutional neural networks (CNNs) are a type of neural network that exploit convolution operation for processing data that has grid-like structure. Let the indices i, j, c indicate the positions along the height, width, and depth. Convolution is an operation on two real-valued functions. For three-dimensional tensor such as an RGB image and the wavelet scalogram of multi-channel EEG signal, convolution from the lth layer to the $(l+1)$th layer is defined as:

$$h^{(l+1)}(i,j,c) = \sum_{r=1}^{N_{H_K}^{(l)}} \sum_{s=1}^{N_{W_K}^{(l)}} \sum_{k=1}^{N_C^{(l)}} w_{rsk}^{(c,l)} h^{(l)}(i+r-1, j+s-1, k) \tag{1}$$

where $i \in \{1, \ldots, N_{H_I}^{(l)} - N_{H_K}^{(l)} + 1\}$, $j \in \{1, \ldots, N_{W_I}^{(l)} - N_{W_K}^{(l)} + 1\}$, $c \in \{1, \ldots, N_C^{(l+1)}\}$, and $N_H^{(l)}, N_W^{(l)}, N_C^{(l)}$ denote the height, width, and the number of channels of the kernel or feature map at the lth layer, respectively. For example, $N_{H_K}^{(l)}$ is the height of the kernel at the lth layer. The cth kernel for the lth layer has parameters to be estimated denoted by 3-dimensional tensor $W^{(c,l)} = [w_{ijc}^{(c,l)}]$. The resulted feature map in the lth layer is denoted by 3-dimensional tensor $H^{(l)} = [h^{(l)}(i,j,c)]$.

After convolution, the activation function is typically applied to introduce nonlinearity. The commonly used activation function is Rectified Linear Unit (ReLU).

In practice, CNNs contain several layers, each layer consists of convolution operation and activation function. This means the feature map of the current layer is used as the input for the next layer. In this way, we can extract more complex/abstract feature as the number of layers increases. At the end, the feature map resulted from the last layer is vectorized into a feature vector for regression.

2.2 Multilayer perceptron

Multilayer perceptron (MLP) is a class of feedforward neural network consisting of three types of layers which are input layer, hidden layer, and output layer. An MLP defines a mapping $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ where $\boldsymbol{x}$ is data or feature and $\boldsymbol{\theta}$ is model parameters. At the transition from lth layer to $(l+1)$layer, the MLP do affine transformation and then add nonlinearity:

$$\boldsymbol{h}^{(l+1)} = f(\boldsymbol{W}^{(l)\top} \boldsymbol{h}^{(l)} + \boldsymbol{b}^{(l)}) \tag{2}$$

where $\boldsymbol{h}^{(l)}$ is the activation at the lth layer, $\boldsymbol{W}^{(l)}$ is called weights at the lth layer, $\boldsymbol{b}^{(l)}$ is called a bias at the lth layer, and $f(\cdot)$ denotes the activation function. These weights and bias are the parameters to be estimated. The commonly used activation function is rectified linear unit (ReLU) due to its performance and computational efficiency. The commonly used loss function for regression is ℓ_2 loss because it penalize large errors.

The process of estimating parameters can be done by using gradient descent algorithm. Hyperparameters in our method are mother wavelet and its parameters, CNN architecture, the numbers of layers and filters, input dimension, activation function, the number of layers and unit in MLP, learning rate, regularization weight, the number of epochs for training.

3 Methodology

3.1 Overview

Our method has two main steps: feature extraction and regression modelling. To estimate valence and arousal using EEG signals, the feature has to be extracted from preprocessed EEG signal. To extract features, we first transform the signal into time-frequency domain using CWT with Morse wavelet. CNN is then used to automatically extract feature from the scalogram. In this study, we use ResNet as base CNN architecture. Features extracted by the CNN is used to estimate valence and arousal using MLP. A diagram of our framwork is shown in *Figure 1*. The results are then evaluated on the DEAP dataset using root-mean-square error (RMSE) and sign agreement metric (SAGR).

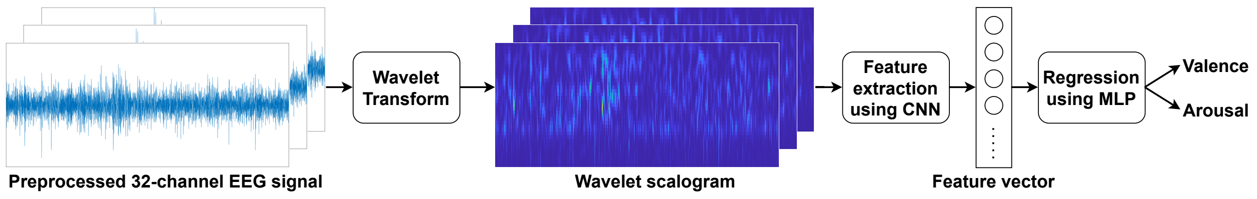


Figure 1: Our wavelet-based CNN framework.

3.2 Continuous wavelet transform

Wavelet transform is a signal processing technique that transform the time-domain signal into the time-frequency domain called wavelet scalogram. Compared with short-time Fourier transform (STFT) which has a fixed window length, the wavelet scalogram provide better temporal resolution, i.e., give higher temporal resolution at a higher frequency and higher frequency resolution at a lower frequency. [GAF21, ZCZ+16] showed that higher frequency bands are more useful for emotion recognition than lower ones. This shows the strength of wavelet scalogram over the STFT. The scalogram is also useful when dealing with non-stationary data.

Continuous wavelet transform (CWT) allows the parameters of translation and scale of the wavelets to be varied continuously. The CWT of a signal $x(t)$ at a scale a and translation b can be expressed as

$$X(a,b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t)\psi^* \left(\frac{t-b}{a}\right) dt \tag{3}$$

where $\psi(t)$ is a mother wavelet which is a continuous function in both time and frequency domain, and ψ^* denotes a complex conjugate of the mother wavelet. The scalogram is defined as

$$S(a,b) = \frac{1}{|a|} \left| \int_{-\infty}^{\infty} x(t)\psi^* \left(\frac{t-b}{a}\right) dt \right|^2 \tag{4}$$

The Fourier transform of the generalized Morse wavelet can be expressed as

$$\Psi_{P,\gamma}(\omega) = U(\omega) a_{P,\gamma} \omega^{P^2/\gamma} e^{-\omega^\gamma} \tag{5}$$

where $U(\omega)$ is the unit step function, $a_{P,\gamma}$ is a normalizing constant, P^2 is time-bandwidth product, and γ is a parameter controlling the symmetry of the Morse wavelet.

Figure 2 shows an example of scalogram of the EEG signal at electrode Fp1 of the participant 29 when watching video 3 (joy video). *Figure 3* shows an example of scalogram of the EEG signal at electrode Fp1 of the participant 29 when watching video 23 (video of how to fight loneliness). It can be seen through the scalogram that *Figure 2* has greater activation (greater magnitude in more vast area) than *Figure 3*. The greater activation in the left frontal (e.g. Fp1) associate with happiness in this case represent by high valence. This is an accordance with previous research that stated a relatively greater left frontal activation is associated with a positive emotions, such as joy or happiness [AF19].

3.3 Convolutional neural network

A convolutional neural network (CNN) is a kind of neural network exploiting convolution operation to extract features from two-dimensional or grid-like data. The advantages of CNN are that it can extract more complex features than traditional methods and the process of designing features is no longer required.

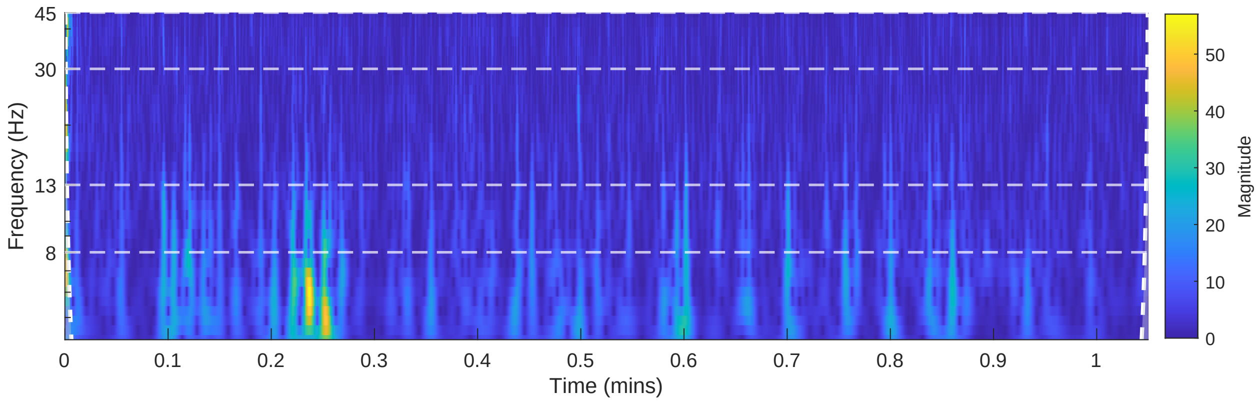


Figure 2: Scalogram of the EEG signal of electrode Fp1 of the participant 29 when watching video 3. This example has a valence of 9.00 and an arousal of 7.08.

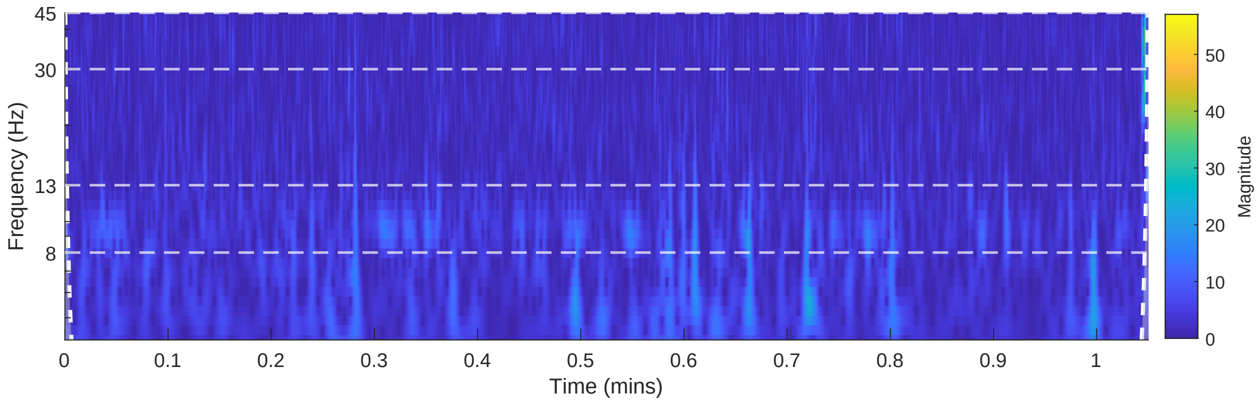


Figure 3: Scalogram of the EEG signal at electrode Fp1 of the participant 29 when watching video 23. This example has a valence of 2.05 and an arousal of 1.00.

3.4 Multilayer perceptron

The multilayer perceptron is a class of feedforward neural networks consisting of multiple layers and units. Each unit does affine transformation and typically adds nonlinearity. Due to the number of units and nonlinearity, one can learn complex patterns from data for the purpose of machine learning. The process of estimating model parameters can be done by using a gradient descent algorithm. In this study, we implement our method using PyTorch library [PGM+19].

3.5 Model evaluation

To evaluate the models, we use the root-mean-square error (RMSE) and sign agreement metric (SAGR). The RMSE measures the average magnitude of the errors having penalty on large errors. The SAGR measures agreement of signs between ground truths and predictions. In addition, the SAGR can provide another aspect in evaluation because the polarities of valence and arousal are also important. The SAGR metric is defined as follows:

$$\mathrm{SAGR} = \frac{1}{N}\sum_{i=1}^{N} \delta(\mathrm{sign}(\hat{y}_i), \mathrm{sign}(y_i)) \tag{6}$$

where $y \in \mathbb{R}^2$ represents ground truth, $\hat{y}$ represents prediction, N represents the number of samples, and δ is the Kronecker delta function, defined as:

$$\delta(a,b) = \begin{cases} 1, & a = b \\ 0, & a \neq b \end{cases} \tag{7}$$

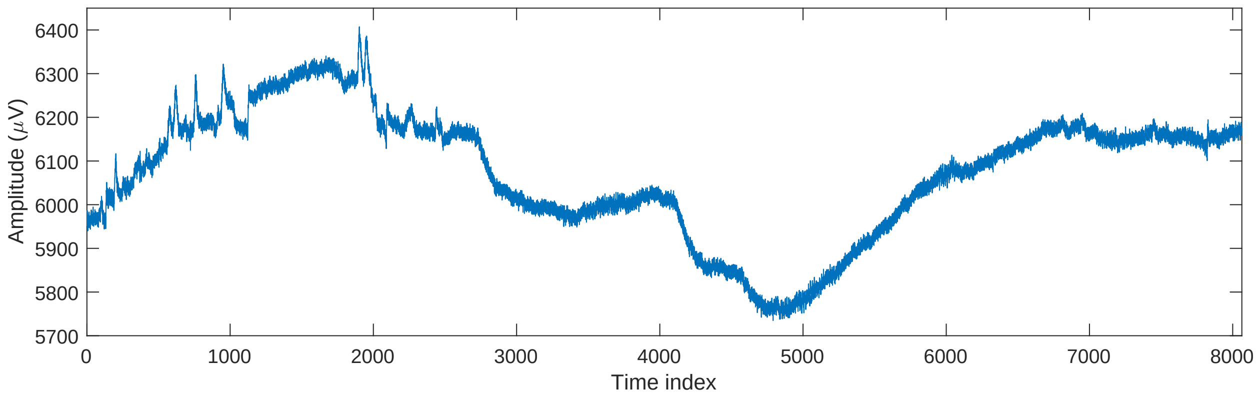


Figure 4: A plot of raw EEG signal at electrode Fp1 of participant 1 when watching video 1.

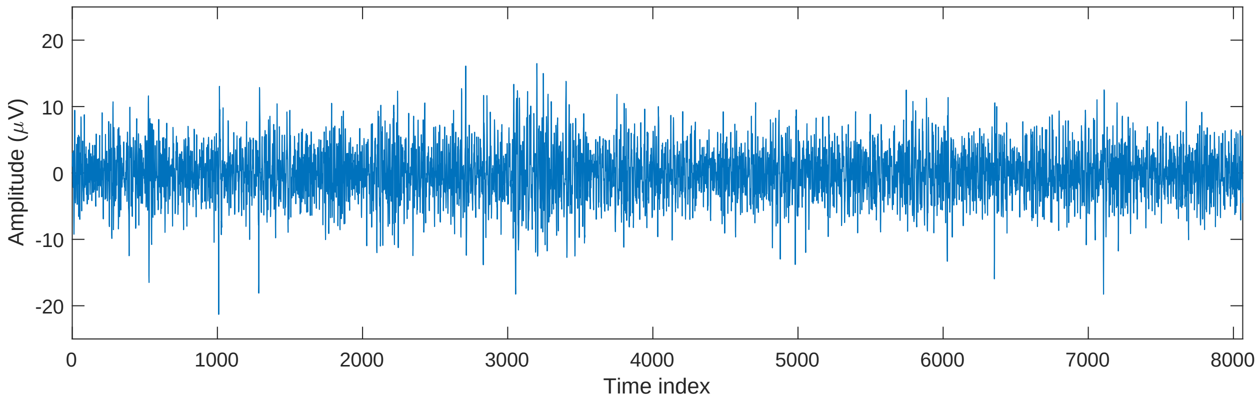


Figure 5: A plot of preprocessed EEG signal at electrode Fp1 of participant 1 when watching video 1.

4 Data description

DEAP, A Database for Emotion Analysis using Physiological Signals, is a multimodal dataset for analysing human affective state [KMS+11]. The dataset was collected from 32 subjects (50%male), aged between 19 and 37 (mean 26.9). It used music video stimulus to evoke different emotion state. The experiment was performed in laboratory experiment with controlled illumination. Every subject followed the experimental setup for the data collection that can be seen in *Figure 6* 2 minutes baseline recording, first 20 trials, break, and second 20 trials. Each trial contains a 2 second displaying current trial number, a 5 second baseline recording, 1 minute music video stimulus, and fill self-assessment of valence, arousal, dominance, and liking in a continuous scale ranging from 1 to 9 and familiarity in a discrete scale ranging from 1 to 5. During watching the music video stimuli, subject's physiological signal such as EEG, Electromyography (EMG), Electrooculography (EOG), Galvanic Skin Response (GSR), Blood Volume Pressure (BVP), temperature, and respiration were recorded. As main data in this paper, we use preprocessed data of the EEG in DEAP which has specification as already down sampled to 128 Hz, EOG artefacts were removed, filtered with bandpass filter (4.0 Hz to 45.0 Hz).

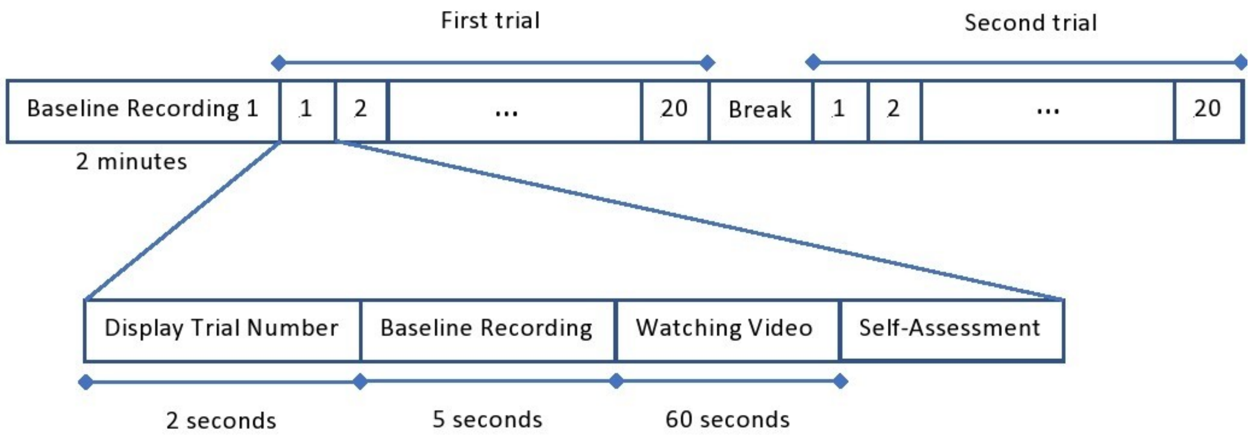


Figure 6: Experimental setup for the data collection

In this research, we use wavelet transform instead of other transforms is because the merit of wavelet transforms dealing with non-stationarity of EEG signal. The raw data EEG which is before pre-processing and the preprocessed data of EEG can be seen in *Figure 4* and *Figure 5*. From *Figure 4* it seems that the raw data EEG is non-stationary because it has sinusoidal trend which makes the mean and variance is not constant over time. In contrast, from *Figure 5* seems that the preprocessed data EEG is stationary. To show that those pre-assumption by looking the graph are correct, we did stationary test using ADF-test and KPSS-test command in MATLAB.

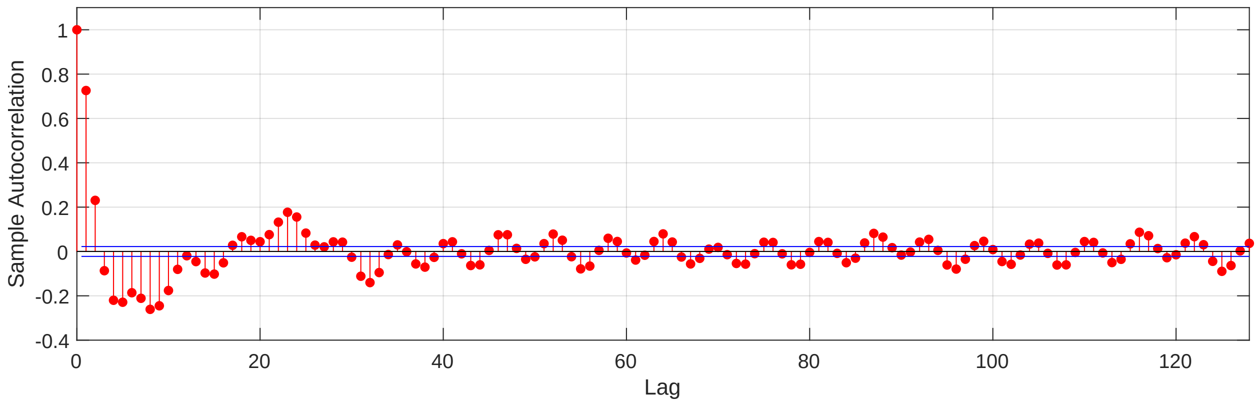


Figure 7: A plot of sample autocorrelation function of EEG of electrode Fp1 of participant 1 when watching video 1.

A time series data is said to be strictly stationary if the joint distribution of $X(t_1), \dots, X(t_n)$ is the same as the joint distribution of $X(t_1+\tau), \dots, X(t_n+\tau)$ for all $t_1, \dots, t_n, \tau$ where $X(t)$ is the random variable at time t [YS05]. Unfortunately, the strictly stationary definition is too strong for most application. Therefore, in practical application, the weakly (wide sense) stationary definition is often used [KKSK11]. Wide sense stationary is based on the first and second moment of X(t), which is the mean and the autocovariance. A time series data is said to be wide sense stationary if:

1. The mean of $X(t)$ that can be formulated as $E[X(t)] = \mu$ is constant over time. The $E[\cdot]$ is an expectation operator
2. The auto-covariance function depends only on the lag, τ, $Cov[X(t), X(t+\tau)] = \gamma(\tau)$

If $\tau = 0$, the wide sense stationary implies that both the variance and the mean are constant. To analyze stationarity of a time series data, we can formulate a time series data, $x(t)$, as below [Ziv06]:

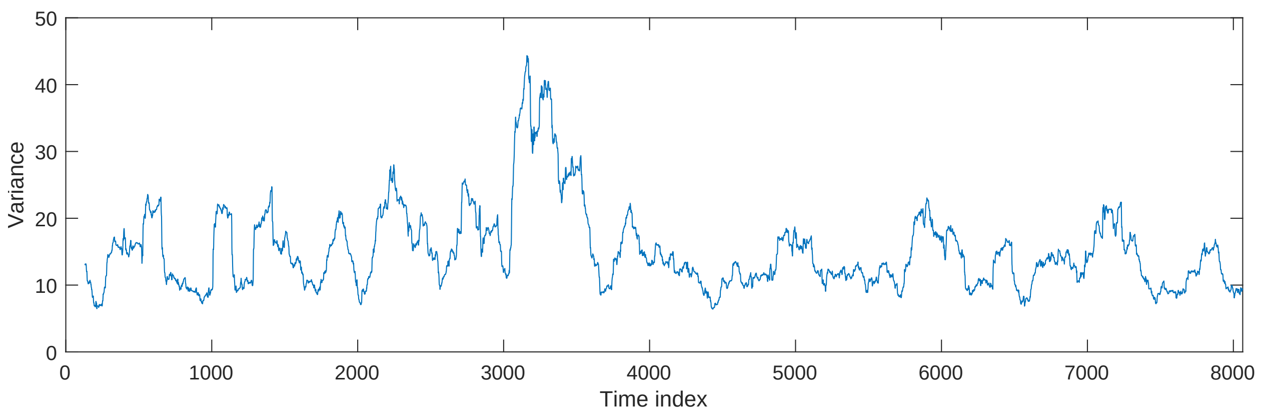


Figure 8: A plot of sample variance (calculated with a window length of 128) of EEG of electrode Fp1 of participant 1 when watching video 1.

$$x(t) = TD(t) + z(t) \tag{8}$$

$$TD(t) = \kappa + \delta t \tag{9}$$

$$z(t) = \phi z(t-1) + e(t) \tag{10}$$

Where $TD(t)$ is a deterministic linear trend, $z(t)$ is an AR(1) process, $e(t)$ is the white noise Gaussian process with zero mean and variance σ^2.

Non-stationarity can be found in time series data in 3 ways [KKSK11], which are:

1. The process has a deterministic trend, which is $\delta \neq 0$. It can be analysed using Kwiatkowski-Phillips-Schmidt-Schin (KPSS) test. KPSS-test command in MATLAB is a stationary test which rely on the mean of a time series data. It has null hypothesis that a time series data has constant mean over time, it will return 0. The alternative hypothesis is that the time series data has deterministic trend, it will return 1.

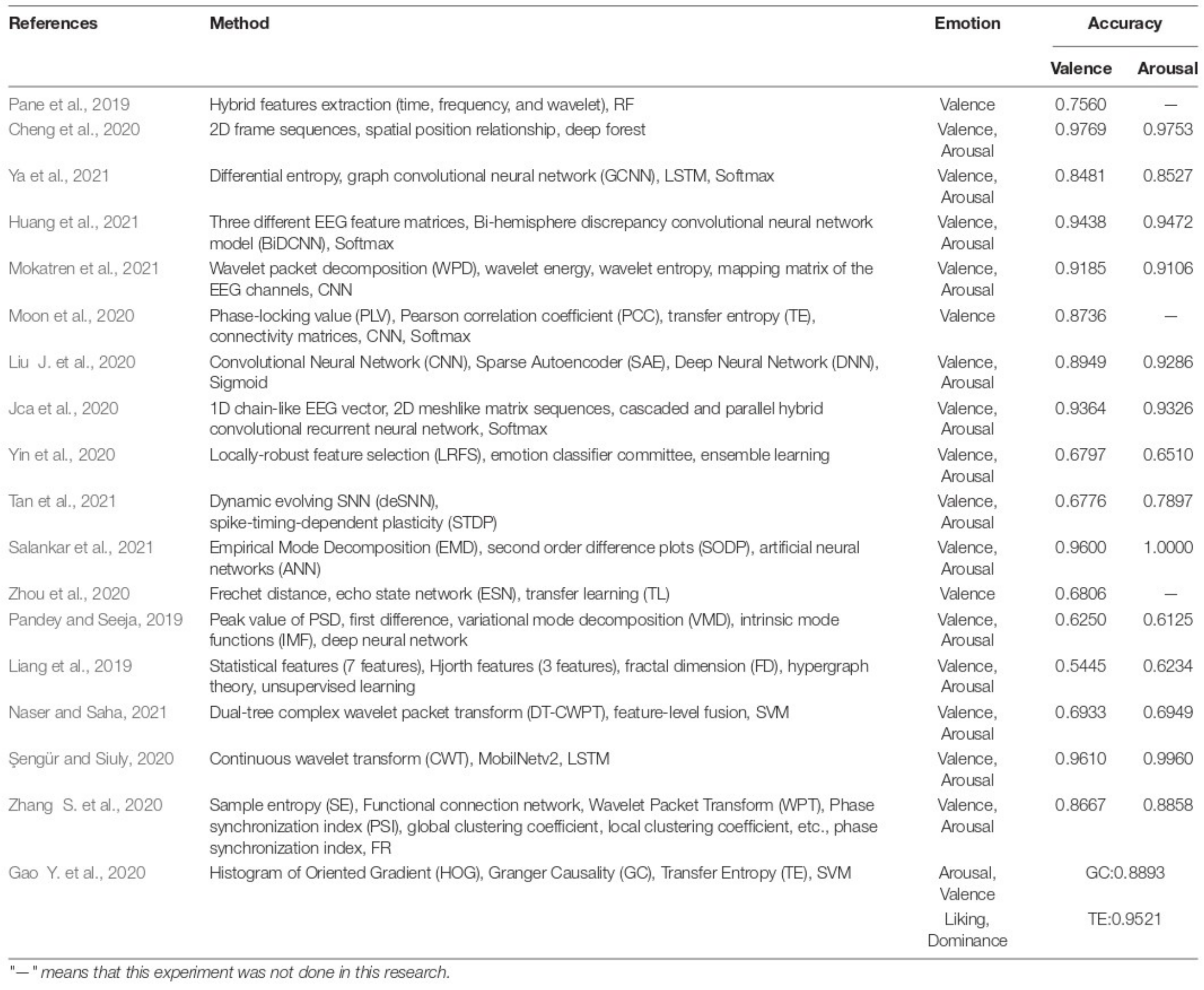
2. The process has a unit root, which is $\phi = 1$. It can be analysed using Augmented Dickey-Fuller (ADF) test. ADF-test command in MATLAB is a stationary test which rely on the presence of unit root in a time series data. It has null hypothesis that a time series data has unit root which means the data is non-stationary, it will return 0. The alternative hypothesis is that a time series data has absence of unit root which can be interpreted as stationary, it will return 1.

3. The process's variance changes over time.

The problem of evaluating the stationarity of time series data can be done using KPSS test for detecting the trend, ADF test for the presence of unit root, and detecting change of variance over time [KKSK11]. Here is an example of stationarity test using time series data from subject 1 at the FP1 channel. The time series data can be seen in *Figure 5* while the plot of sample autocorrelation and variance can be seen in *Figure 7* and *Figure 8*. From the analysis using ADF-test, it returned h=1 and p=0.001 which mean it has no unit root (it also can be seen from magnitude of the first lag of sample autocorrelation graph that is less than 1). From the KPSS-test, it returned h=0 and p=0.1 which means that the mean is almost constant at zero. However, this time series data is categorized as non-stationary because the variance change over time that can be seen in *Figure 8*. This result agrees with the previous work by [KKSK11], which stated that time series data of Magnetoenchepalography (MEG) and EEG usually had a constant mean but were categorized as non-stationary due to the changing of variance over time.

Previous research that also use DEAP dataset to build EEG based emotion recognition model can be seen in the *Table 1*.

Table 1: Previous research of emotion recognition that used DEAP Dataset [LZLK21]



References	Method	Emotion	Accuracy	
			Valence	**Arousal**
Pane et al., 2019	Hybrid features extraction (time, frequency, and wavelet), RF	Valence	0.7560	–
Cheng et al., 2020	2D frame sequences, spatial position relationship, deep forest	Valence, Arousal	0.9769	0.9753
Ya et al., 2021	Differential entropy, graph convolutional neural network (GCNN), LSTM, Softmax	Valence, Arousal	0.8481	0.8527
Huang et al., 2021	Three different EEG feature matrices, Bi-hemisphere discrepancy convolutional neural network model (BiDCNN), Softmax	Valence, Arousal	0.9438	0.9472
Mokatren et al., 2021	Wavelet packet decomposition (WPD), wavelet energy, wavelet entropy, mapping matrix of the EEG channels, CNN	Valence, Arousal	0.9185	0.9106
Moon et al., 2020	Phase-locking value (PLV), Pearson correlation coefficient (PCC), transfer entropy (TE), connectivity matrices, CNN, Softmax	Valence	0.8736	–
Liu J. et al., 2020	Convolutional Neural Network (CNN), Sparse Autoencoder (SAE), Deep Neural Network (DNN), Sigmoid	Valence, Arousal	0.8949	0.9286
Jca et al., 2020	1D chain-like EEG vector, 2D meshlike matrix sequences, cascaded and parallel hybrid convolutional recurrent neural network, Softmax	Valence, Arousal	0.9364	0.9326
Yin et al., 2020	Locally-robust feature selection (LRFS), emotion classifier committee, ensemble learning	Valence, Arousal	0.6797	0.6510
Tan et al., 2021	Dynamic evolving SNN (deSNN), spike-timing-dependent plasticity (STDP)	Valence, Arousal	0.6776	0.7897
Salankar et al., 2021	Empirical Mode Decomposition (EMD), second order difference plots (SODP), artificial neural networks (ANN)	Valence, Arousal	0.9600	1.0000
Zhou et al., 2020	Frechet distance, echo state network (ESN), transfer learning (TL)	Valence	0.6806	–
Pandey and Seeja, 2019	Peak value of PSD, first difference, variational mode decomposition (VMD), intrinsic mode functions (IMF), deep neural network	Valence, Arousal	0.6250	0.6125
Liang et al., 2019	Statistical features (7 features), Hjorth features (3 features), fractal dimension (FD), hypergraph theory, unsupervised learning	Valence, Arousal	0.5445	0.6234
Naser and Saha, 2021	Dual-tree complex wavelet packet transform (DT-CWPT), feature-level fusion, SVM	Valence, Arousal	0.6933	0.6949
Şengür and Siuly, 2020	Continuous wavelet transform (CWT), MobilNetv2, LSTM	Valence, Arousal	0.9610	0.9960
Zhang S. et al., 2020	Sample entropy (SE), Functional connection network, Wavelet Packet Transform (WPT), Phase synchronization index (PSI), global clustering coefficient, local clustering coefficient, etc., phase synchronization index, FR	Valence, Arousal	0.8667	0.8858
Gao Y. et al., 2020	Histogram of Oriented Gradient (HOG), Granger Causality (GC), Transfer Entropy (TE), SVM	Arousal, Valence	GC:0.8893	
		Liking, Dominance	TE:0.9521	

"–" means that this experiment was not done in this research.

5 Experimental results

This section describes experimental settings and discusses the results. In this project, DEAP dataset [KMS+11] is used in training and evaluating the models. The information about this dataset is described in *Section 4*.

Raw EEG signals are preprocessed following [KMS+11]. The raw EEG signals are downsampled to 128 Hz and EOG artefacts are removed. A bandpass filter is applied in a frequency range from 4.0 to 45.0 Hz and then the data is averaged to the common reference. Plots of raw and preprocessed EEG signals are shown in *Figure 4* and *Figure 5*. The preprocessed EEG signals are transformed into wavelet scalograms using CWT with Morse wavelet. Each scalogram is then resized to 224×224, which is a preferable size of ResNet architectures. Because we use 32-channel EEG, there are 32 scalograms for each sample. These scalograms are stacked up to a dimension of 224×224×32 and used as the input of CNN. The CNN and MLP are used to extract features, and estimate valence and arousal. Before training, we split the data into training, validation, and test sets with portion ratios of 80%, 10%, and 10%, respectively. For training, Adam optimizer with a learning rate of 0.0001 and an ℓ_2 penalty weight of 0.0001 is applied to learn the model parameters. The number of epoch and batch size are 30 and 64, respectively.

In this study, we aims to investigate the effect on performance when varying the number of layers in the networks. We use ResNet as the base architecture and vary the number of layers. ResNet architectures contain five convolutional blocks. A basic block of the ResNet is shown in *Figure 9*. The

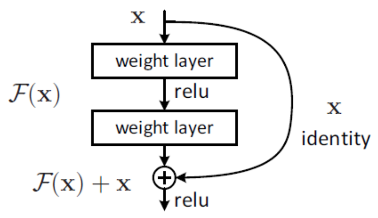

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 9: A building block of ResNet.

differences among variants of ResNet are the numbers of convolutional layers (CLs) and filters. In ResNet-18, each block has four CLs with a kernel size of 3 x 3, except the first block that has only a CL with a kernel size of 7 x 7 and a stride of 2. Max pooling operation with a size of 3 x 3 and a stride of 2 is performed after the first block. Downsampling is performed after the first CL of the third, fourth, and fifth convolutional blocks by convolution with a stride of 2. The convolutional network ends with a global average pooling. The FC layer has originally 1000 units. We change it to 2 units for valence and arousal. There is no dropout applied in ResNet. Every convolutional block, except the first block, in ResNet-18 has equal number of basic blocks. To vary the number of layers, we set the number of basic blocks in each convolutional block, except the first block, to 1, 2, 3, 4, 6, and 8 for simplicity. The configurations of architectures used are shown in *Table 2*.

Table 2: ResNet configurations. Building blocks (See *Figure 9*) are shown in brackets with the number of blocks stacked. Downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2 [HZRS16]. Only ResNet-18 is original. Other variants are our modifications.

layer name	output size	6-layer	10-layer	14-layer	18-layer	26-layer	34-layer	50-layer	66-layer
conv1	112×112	7×7, 64, stride 2							
conv2_x	56×56	3×3 max pool, stride 2							
		$\begin{bmatrix}3\times3, 64\end{bmatrix} \times 1$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix} \times 1$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix} \times 1$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix} \times 2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix} \times 3$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix} \times 4$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix} \times 6$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix} \times 8$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\end{bmatrix} \times 1$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix} \times 1$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix} \times 1$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix} \times 2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix} \times 3$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix} \times 4$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix} \times 6$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\end{bmatrix} \times 1$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix} \times 1$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix} \times 2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix} \times 2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix} \times 3$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix} \times 4$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix} \times 6$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix} \times 8$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\end{bmatrix} \times 1$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix} \times 1$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix} \times 2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix} \times 2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix} \times 3$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix} \times 4$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix} \times 6$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix} \times 8$
	1×1	average pool, 2-d fc							

The results of varying the number of layers based on ResNet structure are shown in *Table 3*. The model with minimum validation loss is ResNet-18. Its loss is 0.2462. The RMSE for valence and arousal are 0.2460 and 0.2465, respectively. The SAGR for valence and arousal are 0.6563 and 0.6094, respectively. With the number of layers is greater than 18, there is no performance improvement. ResNet-18 is also better at estimating valence than the shallower models. On the other hands, ResNet-6 estimate arousal more accurate than the deeper ones.

Table 4 shows the comparison of our method with previous study in terms of RMSE and SAGR. [GAF21] used Hjorth parameters, spectral entropy, wavelet energy and entropy, and intrinsic mode function (IMF) energy and entropy as features. They estimated valence and arousal using these features with k-NN and random forest. Their models were evaluated using 10-fold cross-validation. The results are shown in *Table 4*. The test RMSE of our method was 0.299 for valence and 0.259 for arousal. The test SAGR values for valence and arousal were 0.477 and 0.531, respectively.

6 Conclusion

The purpose of this study is to investigate the effect on the performance of our wavelet-based CNN valence and arousal estimation method when varying the number of layers in the networks. The experiment

Table 3: Validation loss, RMSE, and SAGR of valence and arousal estimation models using ResNet with different number of layers. Bold text denotes better performance.

#Layers	Loss	RMSE		SAGR	
		Valence	Arousal	Valence	Arousal
6	0.2476	0.2624	**0.2319**	0.5156	**0.6641**
10	0.2477	0.2591	0.2356	0.5391	**0.6641**
14	0.2477	0.2602	0.2345	0.5469	0.5625
18	**0.2462**	**0.2460**	0.2465	**0.6563**	0.6094
26	0.2497	0.2557	0.2435	0.5781	0.5391
34	0.2482	0.2614	0.2343	0.5313	0.5781
50	0.2563	0.2713	0.2403	0.5781	0.6016
66	0.2608	0.2739	0.2470	0.5234	0.5234

Table 4: Performance comparison with previous studies on exact valence and arousal estimation.

Authors	Regression Model	RMSE		SAGR	
		Valence	Arousal	Valence	Arousal
[GAF21]	k-NN	0.172	0.163	-	-
	RF	0.153	0.174	-	-
Ours	MLP	0.299	0.259	0.477	0.531

was conducted using ResNet architecture with different number of layers. The results showed that the ResNet-18 performed best and was better at estimating valence than the shallower models. On the other hand, ResNet-6 estimated arousal more accurate than the deeper ones. With the number of layers is greater than 18, there is no performance improvement.

From the results that the deep models are useful for estimating valence and the shallow models are beneficial for arousal estimation. In our view, valence and arousal estimation method can be improved by using multi-level features, i.e., using features from bottom, middle, and top layers, or using the ensemble of CNNs with different depth. In addition, using the well-known CNN architecture requires specific dimension of input, there is information loss in resizing the scalogram, and this may consequently decrease the model performance. In short, we suggest that multi-level features or ensemble of CNN should be applied and the CNN architecture should be designed for scalogram data.

References

[AACP07] Karim Ansari-Asl, Guillaume Chanel, and Thierry Pun. A channel selection method for EEG classification in emotion assessment based on synchronization likelihood. In *2007 15th European Signal Processing Conference*, pages 1241–1245. IEEE, 2007.

[AF19] Soraia M. Alarcão and Manuel J. Fonseca. Emotions recognition using EEG signals: A survey. *IEEE Transactions on Affective Computing*, 10(3):374–393, 2019.

[AGA$^+$04] H Aurlien, IO Gjerde, JH Aarseth, G Eldøen, B Karlsen, H Skeidsvoll, and NE Gilhus. EEG background activity described by a large computerized database. *Clinical Neurophysiology*, 115(3):665–673, 2004.

[AZAA15] M Abo-Zahhad, Sabah M Ahmed, and Sherif N Abbas. A new EEG acquisition protocol for biometric identification using eye blinking signals. *International Journal of Intelligent Systems and Applications*, 7(6):48–54, 2015.

[BMSM] Sara Bagherzadeh, Keivan Maghooli, Ahmad Shalbaf, and Arash Maghsoudi. A hybrid EEG-based emotion recognition approach using wavelet convolutional neural networks (WCNN) and support vector machine. *Basic and Clinical Neuroscience*, pages 0–0.

[CLR14] Patrizio Campisi and Daria La Rocca. Brain waves for automatic biometric-based user recognition. *IEEE transactions on information forensics and security*, 9(5):782–800, 2014.

[GAF21] Filipe Galvão, Soraia M Alarcão, and Manuel J Fonseca. Predicting exact valence and arousal values from EEG. *Sensors*, 21(10):1–21, 2021.

[Hjo70] Bo Hjorth. EEG analysis based on time domain properties. *Electroencephalography and clinical neurophysiology*, 29(3):306–310, 1970.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[JYBM21] Mahshad Javidan, Mohammadreza Yazdchi, Zahra Baharlouei, and Amin Mahnam. Feature and channel selection for designing a regression-based continuous-variable emotion recognition system with two EEG channels. *Biomedical Signal Processing and Control*, 70:102979, 2021.

[KKSK11] Lech Kipiński, Reinhard König, Cezary Sielużycki, and Wojciech Kordecki. Application of modern tests for stationarity to single-trial MEG data. *Biological Cybernetics*, 105(3):183–195, October 2011.

[KMS$^+$11] Sander Koelstra, Christian Muhl, Mohammad Soleymani, Jong-Seok Lee, Ashkan Yazdani, Touradj Ebrahimi, Thierry Pun, Anton Nijholt, and Ioannis Patras. DEAP: A database for emotion analysis; using physiological signals. *IEEE transactions on affective computing*, 3(1):18–31, 2011.

[LL18] H-J Lee and S-G Lee. Arousal-valence recognition using CNN with STFT feature-combined image. *Electronics Letters*, 54(3):134–136, 2018.

[LZLK21] Haoran Liu, Ying Zhang, Yujun Li, and Xiangyi Kong. Review on emotion recognition based on electroencephalography. *Frontiers in Computational Neuroscience*, pages 1–15, 2021.

[Mur11] Muthusamy Murugappan. Human emotion classification using wavelet transform and KNN. In *2011 International Conference on Pattern Analysis and Intelligence Robotics*, volume 1, pages 148–153. IEEE, 2011.

[PGM$^+$19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[PH09] Panagiotis C Petrantonakis and Leontios J Hadjileontiadis. Emotion recognition from EEG using higher order crossings. *IEEE Transactions on information Technology in Biomedicine*, 14(2):186–197, 2009.

[PS21] Pallavi Pandey and KR Seeja. Subject independent emotion recognition system for people with facial deformity: an EEG based approach. *Journal of Ambient Intelligence and Humanized Computing*, 12(2):2311–2320, 2021.

[WW21] Jiang Wang and Mei Wang. Review of the emotional feature extraction and classification using EEG signals. *Cognitive Robotics*, 1:29–40, 2021.

[YS05] Kiyoung Yang and Cyrus Shahabi. On the stationarity of multivariate time series for correlation-based data analysis. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4 pp.–, 2005.

[ZCZ+16] Jianhai Zhang, Ming Chen, Shaokai Zhao, Sanqing Hu, Zhiguo Shi, and Yu Cao. ReliefF-based EEG sensor selection methods for emotion recognition. *Sensors*, 16(10):1558, 2016.

[Ziv06] Eric Zivot. Unit root tests. pages 111–139, 2006.

7 Appendices

7.1 Python code

```
from google.colab import drive
drive.mount('/content/drive')

%cd drive/MyDrive/EE531/

import os
import random
import pandas as pd
import numpy as np
import scipy.io as sio
from tqdm.notebook import tqdm
import matplotlib.pyplot as plt
%matplotlib inline

import torch
import torchvision
import torchvision.transforms as transforms
import torch.nn as nn
import torchvision.models as models
import torch.nn.functional as F

from torchvision.models.resnet import ResNet, BasicBlock, _resnet
from torchvision.datasets import ImageFolder
from torch.utils.data import Dataset, DataLoader
from torchvision.io import read_image
from typing import Type, Any, Callable, Union, List, Optional
from torch import Tensor
from typing import Union, List, Dict, Any, cast

seed = 531
torch.manual_seed(seed)
random.seed(seed)
np.random.seed(seed)
```

7.1.1 Data Preparation

```
class CustomDataset(Dataset):
    def __init__(self, img_dir, annotations_file, transform=None):
        self.img_dir = img_dir
        self.img_labels = pd.read_csv(os.path.join(self.img_dir,annotations_file))
        self.transform = transform

    def __len__(self):
        assert len(self.img_labels) == len([name for name in os.listdir(self.img_dir) if
        ↪  name.split('.')[1] == 'mat'])
        return len(self.img_labels)
    def __getitem__(self, idx):
        label = labels.loc[(labels['Participant'] == idx//40 + 1) & \
                           (labels['Video'] == idx%40 + 1)][['Valence', 'Arousal']]
        label =  np.array(label).reshape((2,))
        label = ((label - 1) / 8).astype(np.float32)    # range from 1 to 9 -> from 0 to
        ↪  1
        image_name = f's{str(idx//40 + 1).zfill(2)}_v{str(idx%40 + 1).zfill(2)}.mat'
        img_path = os.path.join(self.img_dir, image_name)
        image = sio.loadmat(img_path)
        image = np.absolute(image['scalograms']).astype(np.float32)
        if self.transform:
            image = self.transform(image)
        return image, label

labels = pd.read_csv('DEAP/data/scalogram/labels.csv')
transform = transforms.ToTensor()
full_data = CustomDataset('DEAP/data/scalogram', 'labels.csv', transform=transform)
train_size = int(0.8 * len(full_data))
val_size = int(0.1 * len(full_data))
test_size = int(0.1 * len(full_data))
training_data, val_data, test_data = torch.utils.data.random_split(full_data,
↪  [train_size, val_size, test_size], generator=torch.Generator().manual_seed(seed))
```

7.1.2 Model

```
def custom_resnet(x):
    model = ResNet(BasicBlock, [x, x, x, x])    # x is Number of BasicBlocks in each conv
    ↪  block
    model.conv1 = nn.Conv2d(32, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3),
    ↪  bias=False)
    model.fc = nn.Linear(in_features=512, out_features=2, bias=True)
    for m in model.modules():
        if isinstance(m, nn.Conv2d):
            nn.init.kaiming_normal_(m.weight, mode='fan_out', nonlinearity='relu')
        elif isinstance(m, (nn.BatchNorm2d, nn.GroupNorm)):
            nn.init.constant_(m.weight, 1)
            nn.init.constant_(m.bias, 0)
    return model
```

7.1.3 Training

```
def train_loop(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    losses, residuals, SAGRs = 0, 0, 0

    for batch, (X, y) in enumerate(tqdm(dataloader, position=1)):
        # Compute prediction and loss
        X, y = X.cuda(), y.cuda()
        pred = model(X)
        loss = loss_fn(pred, y)

        # Backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        losses += loss.item() * len(X)
        residuals += torch.sum(torch.square(y - pred), 0)
        SAGRs += torch.sum(torch.sign(y - 0.5) == torch.sign(pred - 0.5), 0)
        if batch % 2 == 0:
            loss, current = loss.item(), batch * len(X)
            print(f"\t[{current:>4d}/{size:>4d}]  loss: {loss:>7f}")

    train_loss = losses / size
    rmse_v = np.sqrt(residuals[0].item() / size)
    rmse_a = np.sqrt(residuals[1].item() / size)
    sagr_v = SAGRs[0].item() / size
    sagr_a = SAGRs[1].item() / size

    print(f'Training\t|\tLoss: {train_loss:>7f}  RMSE: ({rmse_v:>7f}, {rmse_a:>7f})
    ↪  SAGR: ({sagr_v:>7f}, {sagr_a:>7f})')

    return train_loss, rmse_v, rmse_a, sagr_v, sagr_a

def test_loop(dataloader, model, loss_fn):
    size = len(dataloader.dataset)
    num_batches = len(dataloader)
    losses, residuals, SAGRs = 0, 0, 0

    with torch.no_grad():
        for X, y in tqdm(dataloader, position=2):
            X, y = X.cuda(), y.cuda()
            pred = model(X)

            losses += loss_fn(pred, y).item() * len(X)
            residuals += torch.sum(torch.square(y - pred), 0)
            SAGRs += torch.sum(torch.sign(y - 0.5) == torch.sign(pred - 0.5), 0)

    test_loss = losses / size
    rmse_v = np.sqrt(residuals[0].item() / size)
    rmse_a = np.sqrt(residuals[1].item() / size)
    sagr_v = SAGRs[0].item() / size
    sagr_a = SAGRs[1].item() / size

    print(f"Test\t\t|\tLoss: {test_loss:>7f}  RMSE: ({rmse_v:>7f}, {rmse_a:>7f})  SAGR:
    ↪  ({sagr_v:>7f}, {sagr_a:>7f})\n")

    return test_loss, rmse_v, rmse_a, sagr_v, sagr_a

def save_checkpoint(state, model_dir, model_name, filename="checkpoint",
↪   lastname=".pth"):
    epoch_num = state["epoch"]

    if not os.path.exists(model_dir):
        os.makedirs(model_dir)

    if not os.path.exists(os.path.join(model_dir, model_name)):
        os.makedirs(os.path.join(model_dir, model_name))

    cp_path = os.path.join(model_dir, model_name, filename + lastname)

    if epoch_num % 1 == 0 and epoch_num >= 0:
        torch.save(state, cp_path.replace("checkpoint", "checkpoint_" + str(epoch_num)))

def train(model, model_name, training_data, val_data, epochs, bs, lr, wd):
    train_dataloader = DataLoader(training_data, batch_size=bs, shuffle=True,
    ↪  num_workers=2, pin_memory=True)
    val_dataloader = DataLoader(val_data, batch_size=bs, shuffle=True, num_workers=2,
    ↪  pin_memory=True)

    loss_fn = nn.MSELoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=lr)

    for epoch in tqdm(range(epochs), position=0):
        print(f"Epoch {epoch+1}\n-------------------------------")
        train_loss, train_rmse_v, train_rmse_a, train_sagr_v, train_sagr_a = \
            train_loop(train_dataloader, model, loss_fn, optimizer)
        val_loss, val_rmse_v, val_rmse_a, val_sagr_v, val_sagr_a = \
            test_loop(val_dataloader, model, loss_fn)

        save_checkpoint(
            {"epoch": epoch + 1,
            "state_dict": model.state_dict(),
            "optimizer": optimizer.state_dict(),
            "train": {'loss': train_loss, 'rmse_v': train_rmse_v, 'rmse_a': train_rmse_a,
            ↪  'sagr_v': train_sagr_v, 'sagr_a': train_sagr_a},
            "val": {'loss': val_loss, 'rmse_v': val_rmse_v, 'rmse_a': val_rmse_a,
            ↪  'sagr_v': val_sagr_v, 'sagr_a': val_sagr_a}},
            model_dir='models',
            model_name=model_name
        )

model = custom_resnet(x=1)  # ResNet - 10 layers
model.cuda()
train(model, 'resnet10-bs64-lr-1e-4-l2-1e-4-', training_data, val_data, epochs=30, bs=64,
↪  lr=1e-4, wd=1e-4)
model = custom_resnet(x=2)  # ResNet - 18 layers
model.cuda()
train(model, 'resnet18-bs64-lr-1e-4-l2-1e-4-', training_data, val_data, epochs=30, bs=64,
↪  lr=1e-4, wd=1e-4)
model = custom_resnet(x=3)  # ResNet - 26 layers
model.cuda()
train(model, 'resnet26-bs64-lr-1e-4-l2-1e-4-', training_data, val_data, epochs=30, bs=64,
↪  lr=1e-4, wd=1e-4)
model = custom_resnet(x=4)  # ResNet - 34 layers
model.cuda()
train(model, 'resnet34-bs64-lr-1e-4-l2-1e-4-', training_data, val_data, epochs=30, bs=64,
↪  lr=1e-4, wd=1e-4)
model = custom_resnet(x=6)  # ResNet - 50 layers
model.cuda()
train(model, 'resnet50-bs64-lr-1e-4-l2-1e-4-', training_data, val_data, epochs=30, bs=64,
↪  lr=1e-4, wd=1e-4)
model = custom_resnet(x=8)  # ResNet - 66 layers
model.cuda()
train(model, 'resnet66-bs64-lr-1e-4-l2-1e-4-', training_data, val_data, epochs=30, bs=64,
↪  lr=1e-4, wd=1e-4)
```

7.1.4 Test

```
model = custom_resnet(x=2) # ResNet-18
best_val_loss = 100000
for t in range(30):
    current_val_loss =
    ↪  torch.load(f'models/resnet18-bs64-lr-1e-4-l2-1e-4/checkpoint_{t+1}.pth')['val']['loss']
    if current_val_loss < best_val_loss:
        best_val_loss = current_val_loss
        best_epoch = t + 1
```

```
state_dict =
↪  torch.load(f'models/resnet18-bs64-lr-1e-4-l2-1e-4/checkpoint_{best_epoch}.pth')['state_dict']
model.load_state_dict(state_dict)
model.eval()
model.cuda()
test_dataloader = DataLoader(test_data, batch_size=64, shuffle=False, num_workers=2,
↪  pin_memory=True)
loss_fn = nn.MSELoss()
test_loss, test_rmse_v, test_rmse_a, test_sagr_v, test_sagr_a =
↪  test_loop(test_dataloader, model, loss_fn)
```

7.2 MATLAB code

```
% load thepreprocessed data EEG in .mat format
raw_data =load('s01.mat')
data = raw_data.data % only take the value, not include the label
datap = permute(data,[2 3 1]) %arrange the data as number of channels x time series x
↪  number of video

% Pick only first trial ( 1 video) and 32 channels
data1 = datap(1:32,:,1)

% ADF Test of 32 channels
adf =[]
for i=1:32
    [h p]=adftest(data1(i,:))
    adf = [adf; [h p]]
end

% KPSS Test of 32 channel
kpss =[]
for i=1:32
    [h p]=kpsstest(data1(i,:))
    kpss = [kpss; [h p]]
end
```