Effects of EMD and Feature Extraction on EEG Analysis

Phuong Huynh, Gregory Warner, and Hong Lin
Department of Computer Science and Engineering Technology, University of Houston Downtown, USA
Email: {huynhp17, warnerg1}@gator.uhd.edu, linh@uhd.edu

***Abstracts*—Brain-computer interfaces have been investigated for more than 20 years and have great potential to develop applications for physicians to diagnose diseases or patients with severe neurologic disabilities to return to interact with society. To gain those purposes requires technics to analyze the EEG data as well as an algorithm to train the model for identifying the patterns or controlling the devices. TensorFlow is a machine learning developed by Google team for internal use and was released for public use in 2015. Since it can train and test on deep learning neural network, it can be used for EEG data. This project used TF-Keras and TensorFlow-DNN to train the models for classifying brain states using EEG data. Neurosky Mindwave Mobile headset and a new device developed from Micro:bit were the recorders for EEG signals in the project. Several technics such as min-max normalization, Ensemble Empirical Mode Decomposition (EEMD), extraction were applied to analyze the recorded EEG data. The results show that the accuracies of TensorFlow-Keras and TensorFlow - DNN models are 97% while the results from XGBoost is 98% when classifying the EEG data from Micro:bit device. The result confirms the ability of application of TensorFlow in identifying EEG data. The technics for processing data contributed to the above results are min-max normalization and data extraction. Moreover, we also verify that the low-frequency drifts in the recorded data is essential to identify the brain states using EEG data. The results also show the application of IMFs generated from EEMD technic as features to build the models for classifying brain states using the EEG data.**

***Index Terms*—TensorFlow, EEG, XGBoost, TensorFlow-Keras (TF-Keras), TensorFlow-DNN (TF-DNN), *Ensemble Empirical Mode Decomposition* (EEMD), Neurosky, Micro:bit, Brain-Computer Interface (BCI)**

I. Introduction

Electroencephalography (EEG) is a measurement of the human brain's potentials emitted by electrical activities from the brain. Galvani, the scientist and philosopher, is remembered as the first person who identified the electrical activity of a living organism in the 18th century [1], and the first EEG recording machine was introduced to the world by Hans Berger in 1929 [2]. EEG signals are recording using electrodes which are placed in different locations on the surface of the scalp. Its function is to detect tiny electrical changes that result from the activities of the brain cells. Each electrode connected to an amplifier and an EEG recording machine. Fig. 1 illustrates how electrodes placed on the scalp and the display on a recording machine.

Depending on the procedure required, several electrodes recorded in parallel can vary from 2 to 256 electrodes [2]. One pair of electrodes makes up a channel which produces a signal during an EEG recording. The amplitude of an EEG signal typically ranges from about 1 to 100 μV for an adult. Brain-Computer Interfaces (BCIs) arise enhancing the applications of EEG data. BCI is a computer-based system that records and analyzes brain signals and generate a control signal(s) to the device which performs the desired action(s). Fig. 2 illustrates the general architecture of a BCI system.

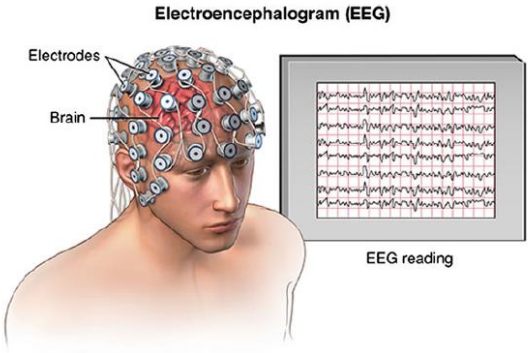

Figure 1. An illustration of EEG recording (Ref. EEG Saint Luke's Health System)

In the BCI architecture, pre-processing is to clean and denoise data to enhance the proper information in the EEG signal. Feature extraction is to transform an EEG signal into features by extracting the most important features from the signal and ensure the provision of sufficient information for classification. Classification assigns the label to each set of features of the EEG signal. The BCI system was used to diagnose a disease, for example, epilepsy or seizure. In a research, V. Srinivasan *et al*. used 128 channel amplifier system to record EEG signals [3]. Approximate entropy was used in extracting features, and two neural networks, Elman network and probabilistic neural network were trained to identify the patterns with the disease. The result was reported as high as 100%. Another application of BCI related to the cognitive load assessment from the combination of EEG data and EDA (electrodermal activity) resulted in high prediction rate in which Random Forest was employed

Manuscript received June 27, 2019; revised December 20, 2019.

doi: 10.12720/jait.11.1.26-34

[4]. The research aimed to develop a device to support visually impaired mobility aids.

Many algorithms for classification are currently utilized to identify EEG data. Among the algorithms selected to classify a specific EEG data in an experiment such as KNN, SVM, Random Forest, Bayes, and Boosting, Random Forest was reported to provide the highest accuracy while KNN and Boost made the second grade in identifying correctly the EEG signals [1].

TensorFlow is a machine learning library for research and production and surveying its performance in the classification of EEG signals is important for the extension of using EEG data in applications. TensorFlow owns the ability to build a deep neural network and therefore, it owns a potential to works well on large EEG data.

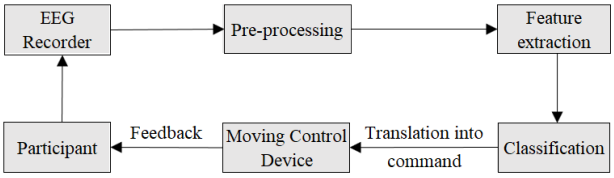

Figure 2. The general architecture of a BCI system

TensorFlow was created by Google Brain team and the version 1.0.0 was released in February 2017, for public use and development. As stated in their official website, "TensorFlow is an open source library for numerical computation using data flow graphs". In TensorFlow, there are three models which are data model, programming model and execution model. TensorFlow data model consists of tensors which are an n-dimensional collection of data. The programming model consists of data flow graphs or computation graphs, and the execution model is the implement of computation at nodes in a sequence, starting from the initial nodes that depend on the inputs. [5]. Fig. 3 shows a simple TensorFlow graph which represents the data flow of the computations. Dataflow graphs are the structures describing how data moves through a graph or a series of processing nodes and are created by users. Each node in the graph represents a mathematical operation, and each connection (also called edge) between nodes is a multidimensional data array (also called tensor).

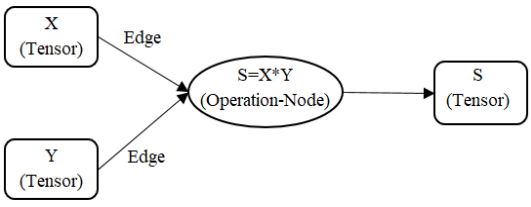

Figure 3. TensorFlow graph

In this project, we use TensorFlow to identify brain states using EEG data. For comparison, XGBoost, a scalable machine learning system for tree boosting, is also selected to train, test models and compare its performance with TensorFlow's. XGBoost was employed in 17/29 winning solutions published at Kaggle's blog in 2015 [6].

II. Methods of Data Processing Applied

A. Min-Max Normalization

Min-max normalization is used to rescale a feature so that all the values in the feature are in a range between 0 and 1 [7].

$$X_{new} = \frac{X - min(X)}{max(X) - min(X)} \quad (1)$$

where X is a feature to be normalized.

B. Power Spectral Intensity and Relative Intensity Radio

To a time series $[x_1, x_2, \dots x_N]$, denote its Fast Fourier Transformation (FFT) result as $[X_1\ X_2, \dots X_N]$. A continuous frequency band from f_{low} to f_{up} is sliced into K bins and forms a vector $band = [f_1, f_2, \dots, f_K]$ such that the lower and upper frequencies of the i^{th} bin are f_i f_{i+1}, respectively [8].

The Power Spectral Intensity (PSI) [9] of the k^{th} bin is evaluated as

$$PSI_k = \sum_{i=\lceil f_k/f_s \rceil}^{\lceil N(f_{i+1}/f_s) \rceil} |X_i|,\ k = 1,2, \dots, K-1 \quad (2)$$

where f_s is the sampling rate, and N is the series length.

Relative Intensity Radio (RIR) [9] is defined on top of PSI.

$$RIR = \frac{PSI_j}{\sum_{k=1}^{K-1} PSI_k},\ j = 1, 2, \dots, K-1 \quad (3)$$

PSI and RIR are both vector features.

C. Petrosian Fractal Dimension (PFD)

PFD [8], [10] is defined as

$$\text{PFD} = \frac{\log_{10} N}{log_{10} N + log_{10}\left(N/(N+0.4N_\delta)\right)} \quad (4)$$

where N is the number of samples in a segment, and N_δ is the number of sign changes in the signal derivative. PFD is a scalar feature.

D. Higuchi Fractal Dimension (HFD)

Higuchi's algorithm [8] constructs k new series from the original series by

$$x_m, x_{m+k}, x_{m+2k}, \dots, x_{m+\lfloor (N-m)/k \rfloor k} \quad (5)$$

where m = 1, 2, …, k

For each time series constructed above, the length $L(m,k)$ is computed by

$$L_{(m,k)} = \frac{\sum_{i=2}^{\lfloor (N-m)/k \rfloor} |x_{m+ik} - x_{m+(i-1)k}| (N-1)}{\lfloor (N-m)/k \rfloor k} \quad (6)$$

The average length is computed as

$$L(k) = \frac{\sum_{i=1}^{k} L(i,k)}{k}$$

This procedure repeats k_{max} times for each k from 1 to k_{max} and then uses as a least-square method to determine the slope of the line that best fit the curve of ln((k)) versus ln(1/k). The slope is the Higuchi Fractal Dimension. HFD is a scalar feature.

E. Hjorth Parameters

To a time series $[x_1, x_2, \ldots x_N]$, the Hjorth mobility [8] and complexity are, respectively, defined as

$$\sqrt{M2/TP} \tag{7}$$

and

$$\sqrt{(M4.TP)/(M2.M2)} \tag{8}$$

where $TP = \sum \frac{x_i}{N}$, $M2 = \frac{\sum d_i}{N}$, $M4 = \sqrt{(d_i - d_{i-1})^2/N}$, and $d_i = x_i - x_{i-1}$.

Hjorth mobility and complexity are both scalar features.

F. The Spectral Entropy

The spectral entropy [8] is defined as follows

$$H = -\frac{i}{log(K)} \sum_{i=1}^{K} RIR_i log RIR_i \tag{9}$$

where RIR_i and K are defined in (3). Spectral entropy is a scalar feature.

G. SVD Entropy

SVD entropy [8] measures using Singular Value Decomposition (SVD). Let the input signal be $[x_1, x_2, \ldots x_N]$. We construct the delay vectors as

$$y_i = \left[x_i, x_{i+\tau}, \ldots, x_{i+(dE-1)\tau}\right] \tag{10}$$

where τ is the delay and dE is the embedding dimension. The embedding space is then constructed by

$$Y = \left[y(1), y(2), \ldots, y\left(y_{i+(dE-1)\tau}\right)\right]^T \tag{11}$$

The SVD is then performed on matrix Y to produce M singular values, $\sigma_1, \sigma_2 \ldots, \sigma_M$ known as the singular spectrum.

The SVD entropy is then defined as

$$H_{SVD} = -\sum_{i=1}^{M} \bar{\sigma}_i \, log_2 \bar{\sigma}_i \tag{12}$$

where M is the number of singular values and $\sigma_1, \sigma_2, \ldots, \sigma_M$ are normalized singular values such that $\bar{\sigma}_i = \sigma_i / \sum_{j=1}^{M} \sigma_j$. SVD entropy is a scalar feature.

H. The Fisher Information

The Fisher information [8] can be defined in normalized singular spectrum used in (10).

$$I = \sum_{i=1}^{M-1} \frac{(\bar{\sigma}_{i+1} - \bar{\sigma}_i)}{\bar{\sigma}_i} \tag{13}$$

Fisher information is a scalar feature.

I. Approximate Entropy

Approximate entropy (ApEn) [8] is a statistical parameter to quantify the regularity of a time series. ApEn is computed by the following steps.

1) Let the input signal be $[x_1, x_2, \ldots x_N]$

2) Build subsequence $x(i, m) = [x_i, x_{i+1}, \ldots, x_{i+m-1}]$ for $1 \le i \le N - m$, where m is the length of the subsequence. In [3], m = 1, 2, or 3.

3) Let r represent the noise filter level, defined as $r = k \times$ SD for k = 0, 0.1, 0.2, …, 0.9.

4) Build a set of subsequences $\{x(j, m)\} = \{x(j, m) \mid j \in [1..N - m]\}$, where $x(j, m)$ is defined in step 2.

5) For each $x(i, m) \in \{x(j, m)\}$, compute

$$C(i, m) = \frac{\sum_{j=1}^{N-m} k_j}{N-m} \tag{14}$$

where

$$k_j = \begin{cases} 1 & if\ |x(i,m) - x(j,m)| < r \\ 0 & otherwise \end{cases} \tag{15}$$

$$ApEn(m, r, N) = \frac{1}{N-M}\left[\sum_{i=1}^{N-m} ln \frac{C(i,m)}{C(i,m+1)}\right] \tag{16}$$

ApEn is a scalar feature.

J. Detrended Fluctuation Analysis (DFA)

The procedures to compute DFA of a time series $[x_1, x_2, \ldots x_N]$ are as follows [8].

1) First integrate x into a new series $y = [y_1, y_2, \ldots y_N]$, where $y(k) = \sum_{i=1}^{k}(x_i - \bar{x})$ and $\bar{x}$ is the average of $[x_1, x_2, \ldots x_N]$.

2) The integrated series is then sliced into boxes of equal length n. In each box of length n, a least-squares line is fit to the data, representing the *trend* in that box. The y coordinate on the straight-line segments is denoted by $y_n(k)$.

3) The root-mean-square fluctuation of the integrated series is calculated by

$$F(n) = \sqrt{(1/N)\sum_{k=1}^{N}[y(k) - y_n(k)]^2}$$

where the part $y(k) - y_n(k)$ is called detrending.

4) The fluctuation can be defined as the slope of the line relating log $F(n)$ to log n.

DFA is a scalar feature.

K. Hurst Exponent

Hurst exponent [8] is also called Rescaled Range statistics (R/S). To calculate the Hurst exponent for time series $X = [x_1, x_2, \ldots x_N]$, the first step is to calculate the accumulated deviation from the mean of time series within range T.

$$X(t, T) = \sum_{i=1}^{t}(x_i - \bar{x}), \ where\ \bar{x} = \frac{1}{T}x_i, t \in [1..N] \tag{17}$$

Then, R(T)/S(T) is calculated as

$$\frac{R(T)}{S(T)} = \frac{max(X(t,T)) - min(X(t,T))}{\sqrt{(1/T)\sum_{t=1}^{T}[x(t) - \bar{x}]^2}} \tag{18}$$

The Hurst Exponent is obtained by calculating the slope of the line produced by ln (R(n)/S(n)) versus ln (n) for $n \in [2, N]$. Hurst exponent is a scalar feature.

III. EEG Data Collection

The Neurosky Mindwave Mobile headset (Neurosky headset) and a simple device developed from Micro:bit (Micro:bit device) was used to record the EEG signals from participants.

The Neurosky headset, a single-sensor EEG device recorded the EEG signals from six persons. There was one person wearing the headset to record the EEG signal at a time. Each participant took three actions which are

meditation, reading and watching a video. Each action last 5 minutes (± 10s). The samples recorded from the Neurosky are called Neurosky data.

Micro:bit device recorded the EEG signals from one person who performs four actions which are controlling, doing nothing, playing piano and reading. Each action last 5 minutes (± 10s). The samples recorded from the Micro:bit device are called Micro:bit data.

Fig. 4 shows the NeuroSky headset and the EEG recording device made of Micro:bit.

IV. Methods to Build the TensorFlow Models Using Neurosky Data

A. Primary Data Processing

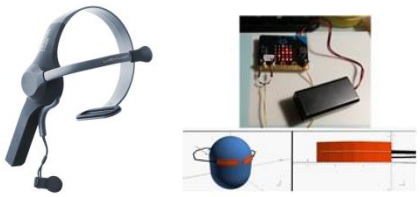

Figure 4. NeuroSky headset (Left) and Micro:bit EEG recording device (Right)

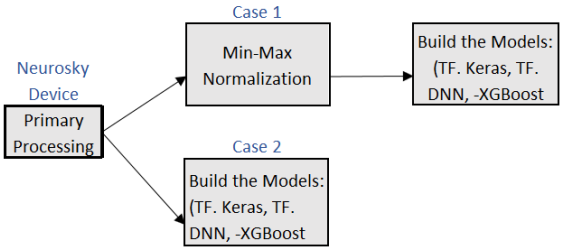

Figure 5. The processes to build models from Neurosky data

TABLE I. Neurosky Brainwave Frequency

Brainwave Type	Frequency Range (Hz)
Delta	0.5 - 2.75
Theta	3.5 - 6.75
Low Alpha	7.5 - 9.25
High Alpha	10 - 11.75
Low Beta	13 - 16.75
High Beta	18 - 29.75
Low Gamma	31 - 39.75
Mid-range Gamma	41 - 49.75

The process to build the models from Neurosky data are shown in Fig. 5. Neurosky data was preprocessed on Neurosky device using its proprietary analysis. Particularly, raw signals were first passed through a band-pass filter to remove frequencies <0.5Hz and >50Hz. They were then decomposed using Fast Fourier Transform to obtain components with frequencies of eight main brain wave bands as illustrated in Table I.

The output from Neurosky for each person had 17 features and 04 features to eliminate are "timestamps", "poorSignal", "tagEvent" and "location". Therefore, Neurosky EEG samples finally had 13 features. Fig. 6 displays the waveforms of Neurosky EEG data of "meditation" state with different frequencies. EEG raw value and EEG raw value volts had very high frequencies comparing with other features. The features on the figure are EEG raw value, EEG raw value volts, attention, meditation, blink stretch, delta, theta, low alpha, high alpha, low beta, high beta, low gamma, and mid-range gamma.

B. Normalization and Model Building

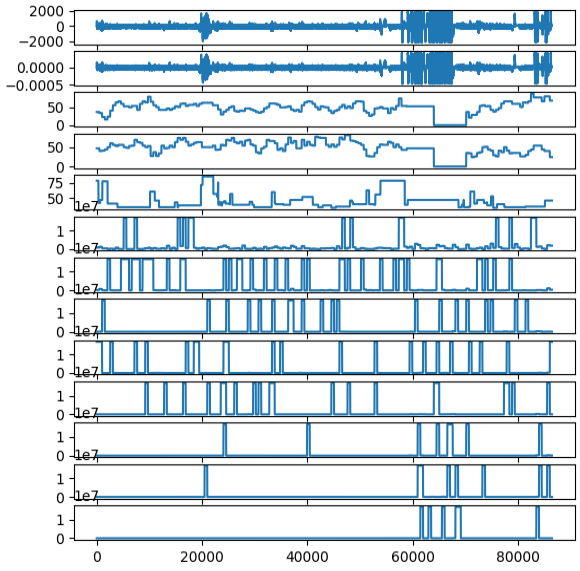

Figure 6. Raw EGG data for meditation

Neurosky data were fed into the model using two formats which were raw data and transformed data. In transforming EEG data, min-max normalization was used to rescale features.

We tested whether normalizing EEG data before feeding to the model can improve the model accuracy comparing with the raw data.

Case 1: From the Neurosky data of each person, 75% of samples were used to train three models and 25% to test the model accuracy in identifying new samples.

Case 2: The Neurosky data were transformed using min-max normalization, and then 75% of data were used to train the three models and 25% to test the model accuracy in identifying new samples.

We utilize from TensorFlow two types of models which are Keras model and deep learning neural network (DNN) model. XGBoost models are also built for the comparison of model accuracy in classification of EEG data between models. Additionally, the results from XGBoost model and DNN model are also combined for further assessment.

V. Methods to Build the Tensorflow Models Using Micro:bit Data

A. Primary Data Processing

Fig. 7 shows the waveform of EEG data from Micro:bit device. The signals for 04 actions look

distinguished. *Micro:bit device* did not have the function to pre-process the EEG data recorded. Therefore, we need to pre-process and extract the EEG data before feeding them into the models.

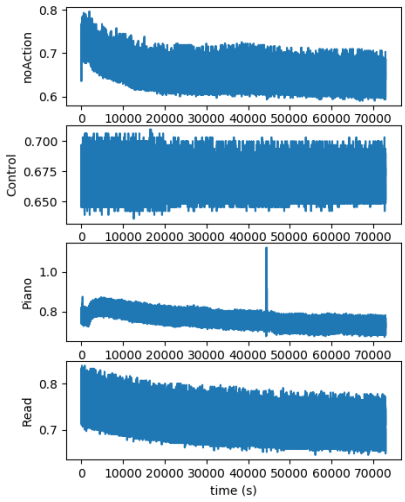

Figure 7. EEG data from Micro:bit device

The raw data from Micro:bit proceeded as indicated in Fig. 8. All the processes which were considered for this experiment were decomposition, filter, extraction, and normalization. Firstly, in data decomposition, *Ensemble Empirical Mode Decomposition* (EEMD) was used to determine Intrinsic Mode Functions (IMFs). EMD stands for Empirical Mode Decomposition and EEMD is a noise-assisted method to improve shifting and generate a better EEG data from Micro:bit device set of IMFs [10], [11]. Secondly, filtering process also used EEMD technic to decompose the data so that the first IMF and 10% of total IMFs with low frequency could be eliminated. We then compute the summary of remaining IMFs as the output of this process. Thirdly, data extraction was computed based on an open source python module for EEG feature extraction [8]. Particularly, Fast Fourier transform was used to extract the PSI (2), such as delta (0.5-4Hz), theta (4-7Hz), alpha (8-12Hz), beta (12-30Hz), and gamma (30-100Hz). RIR (3) computed from PSIs were deltaRIR, thetaRIR, alphaRIR, betaRIR, and gammaRIR. Though Neurosky data also used Fast Fourier transformation to transform its data, the range of frequency is just in the interval [0.5Hz, 50Hz]. Other features also extracted were PFD in (4), HFD, Hjorth Parameters in (7) & (8), Spectral Entropy in (9), SDV Entropy computed from (10), (11) & (12), Fisher Information in (13), Approximate Entropy in (16), Detrended Fluctuation Analysis, and Hurst Exponent computed from (17) & (18). There were 20 features extracted in total. One more datum transformation technic is min-max normalization which is described in (1). We processed EEG data through 8 cases. In *case 1*, raw data were filtered, extracted and normalized prior to being used to train the model; in *case 2*, raw data were filtered and extracted prior to being used to train the model; in *case 3*, raw data were extracted and normalized prior to being used to train the model; in *case 4*, raw data were extracted prior to being used to train the model; in *case 5*, raw data were decomposed and normalized prior to being used to train the model; in *case 6*, raw data were decomposed prior to being used to train the model; in *case 7*: raw data were used to train the model; finally, in *case 8*, raw data were filtered and normalized before used to train the model. The processed data was then divided into train data (75%) and test data (25%).

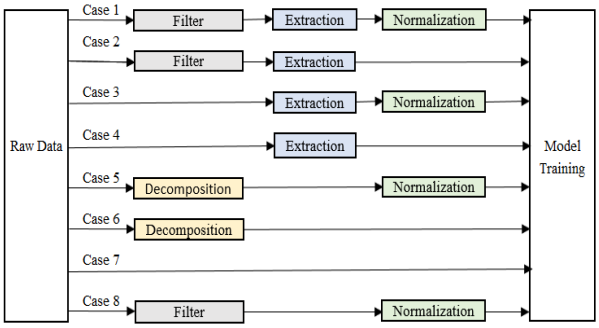

Figure 8. Micro:bit data processing

VI. The Results Recorded from the Models of Neurosky EEG Data

The TF-Keras model resulting in better accuracy in classifying the data had 03 dense layers, and the output shape for each dense layer was set at 50. The optimized TF-DNN model also had 03 layers and each layer had 50 nodes.

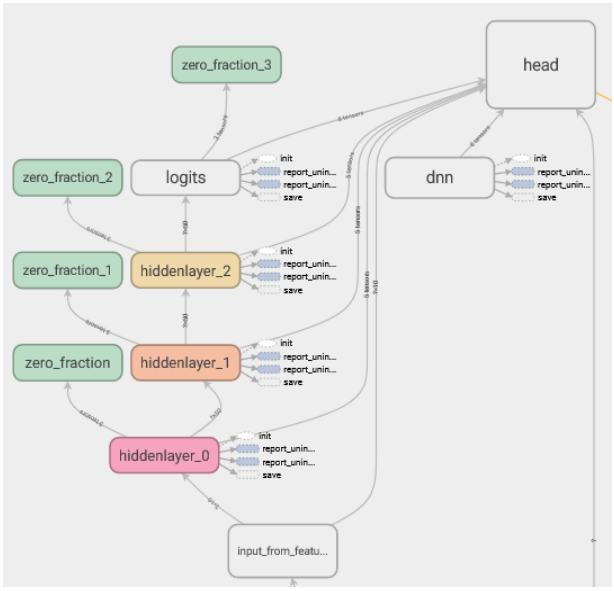

Figure 9. The structure of node DNN in TF-DNN model. (subject 1 and extracted-normalized EEG data)

TensorFlow provides tensorboard which is a tool for users to look inside the model. Fig. 9 displays the structure of this deep neural network of 03 hidden layers. This figure is a part in the general structure which is generated by tensorboard and displays all activities in training a deep learning model for subject 1. Similarly, the graph of TF-Keras model is also exported from tensorboard and is shown in Fig. 10. This model has 03 denses (layers) representing the fully connected nodes between layers. The accuracy of the model at each

iteration also recorded and shown as in Fig. 11. Though DNN and Keras model, both has the same number of layers, but their structures are different and so are their results. The third model to compare with two TensorFlow models is XGBoost and its tree is displayed in Fig. 12. XGBoost algorithm is quite different from neural network models since it is fitted based on the gradient of loss generated from the previous step.

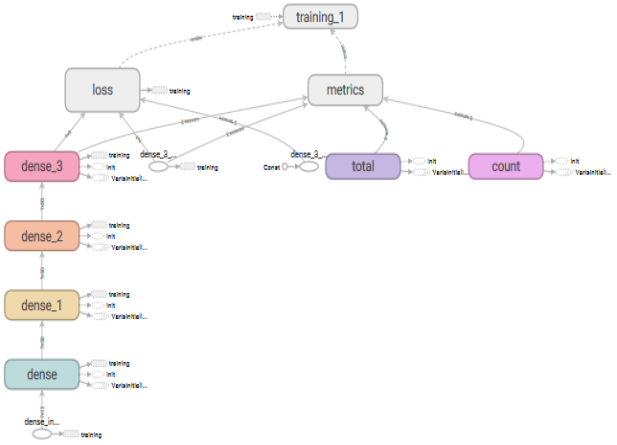

Figure 10. The graph of TF-Keras model. (subject 1 and extracted-normalized EEG data)

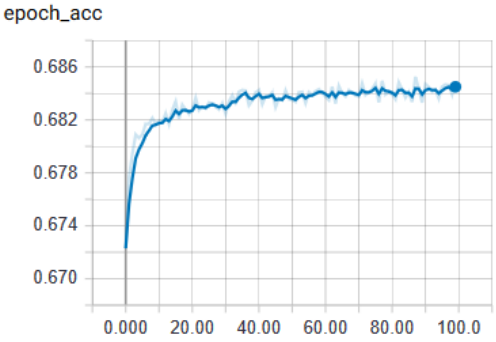

Figure 11. The accuracy of TF-Keras model in training. (The model of subject1)

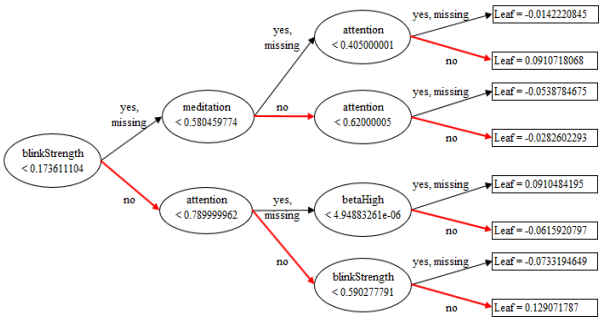

Figure 12. XGBoost tree of the model of subject 1

The results from 03 models are displayed in Fig. 13. Fig. 13(a) and 9(b) show that the models using normalized EEG data result in higher accuracy than the models using directly the raw EEG data. Moreover, the figures also shows that the increase in model accuracy between normalized and not normalized data is higher in TF-DNN models than in TF-Keras models. Thus, min-max normalization on EEG data before feeding to the models has improved the performance of Keras and DNN model, especially in DNN models.

We also see in Fig. 13(c) that the XGBoost models have high accuracy (≥0.95) whether the input data were normalized or not. This can be explained by the algorithm of XGBoost, a gradient boosted decision tree. While the neural networks such as Keras and DNN fit the models based on computing connection weights and activation functions, XGBoost is based on the best split for each step by computing a structure score. The smaller the score is, the better the structure is or in other words, the better model accuracy is.

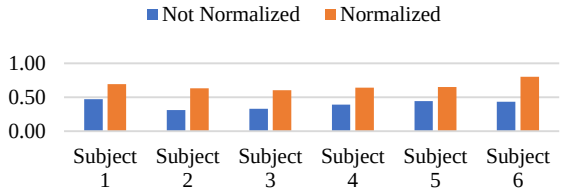

Figure 13(a). Comparison of TF-Keras models

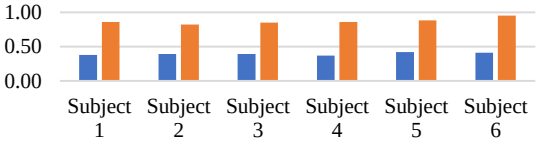

Figure 13(b). Comparison of TF-DNN models

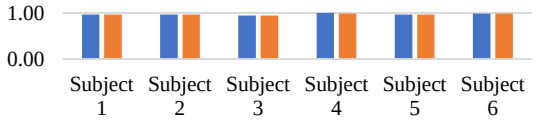

Figure 13(c). Comparison of XGBoost models

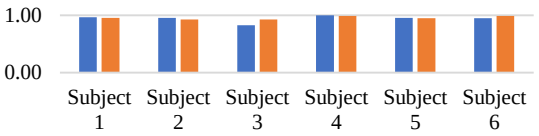

Figure 13(d). Comparison of resemble results from XGBoost and TF.DNN models

VII. The Results Recorded from the Models Using Micro:bit EEG Data

Similar with the model of Neurosky data, in the TF-Keras models for the Micro:bit data, the structure with 03 dense layers and the output shape for each dense layer with 50 nodes provides a better accuracy in classifying the EEG data. TF-DNN models also have 03 layers and each layer had 50 nodes. Tensorboard are used to generate the graph for the two models. TF-DNN model of case 3 has the same graph as subject 1 (Fig. 9). Though the structures of the models of subject 1 of Neurosky data and case 3 of Micro:bit are similar, the general activities in each model are different depending

on the character of the data input. Similarly, the graph of TF-Keras model for case 3 is also the same as subject 1's (Fig. 10). It also has 03 dense layers and each dense also has 50 nodes. The recording of the accuracy of the Keras model at each training step are shown that the models reach their highest accuracy rather soon since the accuracy curve is very steep and quickly approach a horizontal line (Fig. 14). The third model is XGBoost whose tree is displayed in Fig. 15. This tree is less complicated than the tree of subject 1 in Fig. 12.

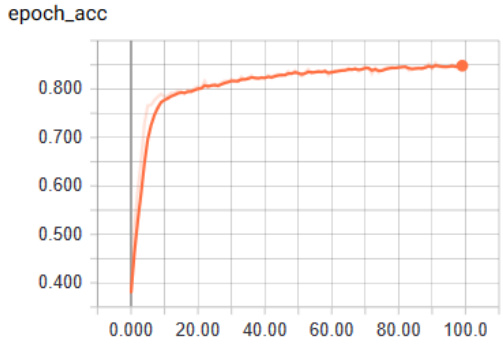

Figure 14. The accuracy and loss of TF-Keras model in training. (Extracted-normalized EEG data)

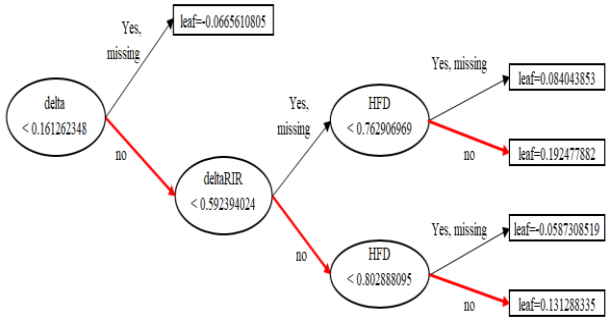

Figure 15. XGBoost tree of extracted-normalized-EEG-data model

These models were then used to make prediction. Fig. 16(a), (b), (c) shows the comparison of the performance of models using normalized data and not-normalized data. We can generally observe that when normalization is included in data processing before feeding to the model, it results in higher accuracy in classifying the test data. However, how much the influence of normalization on the model accuracy still depends on whether filtering or extracting or decomposing data was implemented before normalizing. Particularly, when extracting data were applied (case 3 & 4), the accuracies of all models are high and the differences of model accuracies when utilizing extracted data with (case 3) and without normalization (case 4) are small. Therefore, the influence of normalization on the model accuracies is small when EEG data were just extracted. Similarly, when filtering and extracting EEG data were implemented (case 1&2), the accuracies of all these models are also high, and the difference of the model accuracy between filtered-extracted data with (case 1) and without normalization (case 2) is also small. It provides the evidence that there is a small influence of normalization on the model accuracies when filtering and/or extracting data were applied. Conversely, when decomposing data was implemented (case 5 & 6), the differences of model accuracies between the models using decomposed data with (case 5) and without normalization (case 6) are high. This is the evidence that normalization has a large influence on the model performance when decomposed data are employed.

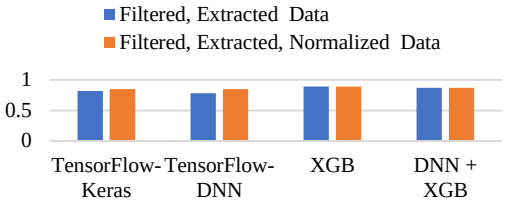

Figure 16(a). The comparison of models using filtered-extracted data with and without normalization

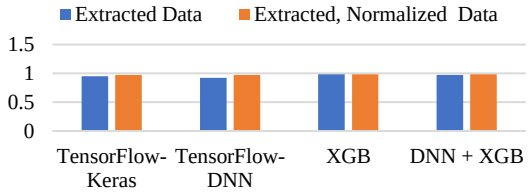

Figure 16(b). The comparison of models using extracted data with and without normalization

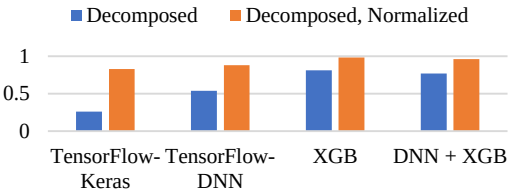

Figure 16(c). The comparison of models using decomposed data with and without normalization

Fig. 17 shows the accuracy of models recorded from the cases 1, 3, 5, 7 & 8. Normalizing are applied on case 1, 3, 5, 8 and case 7 is using raw data. At first, we can realize that all three models utilizing the raw data (case 7) result in the absolute accuracy (100%) and the ensemble result of DNN and XGBoost models is also absolute while the models using filtered-normalized data (case 8) obtains the lowest accuracy among models in Fig. 17. *Secondly*, the three models in case 3 (in which the data are extracted) also result in high accuracy (0.97) and the ensemble result of the two models is even higher (0.98). *Thirdly*, the accuracies of the three models in case 5 (in which data are decomposed) are much various. We can see that just XGBoost provides the highest accuracy (0.98) while two TensorFlow models have the lower accuracies (0.83 for Keras and 0.85 for DNN). The accuracy of the ensemble result remains high (0.96). *Finally*, the accuracy of the three models in case 1 (in which data are filtered and extracted) are rather lower than other cases. The lowest accuracy in case 1 is from

two TensorFlow models (0.85) and the highest is from XGBoost model (0.89). The ensemble accuracy is also low (0.89) comparing to the ones in the other cases (≥0.95). We also realize that Keras and DNN return the accuracies very close to each other when the same data input was fed to train the model, except for the decomposed data. Details of the model accuracy for normalized data are shown on Table II.

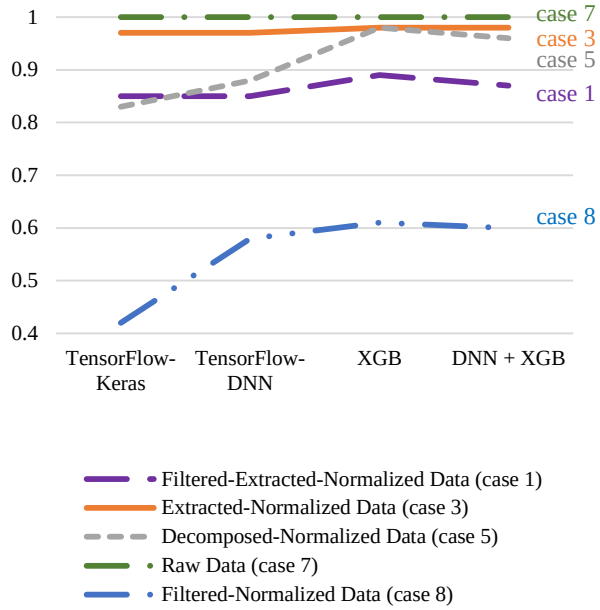

Figure 17. The comparison of models using the cases of normalized data and raw data

TABLE II. THE MODEL ACCURACY FOR NORMALIZED DATA

Data	TF-Keras	TF-DNN	XGB	DNN + XGB
Filtered-Extracted-Normalized Data (case 1)	0.85	0.85	0.89	0.87
Extracted-Normalized Data (case 3)	0.97	0.97	0.98	0.98
Decomposed-Normalized Data (case 5)	0.83	0.88	0.98	0.96
Raw Data (case 7)	1	1	1	1
Filtered-Normalized Data (case 8)	0.42	0.58	0.61	0.6

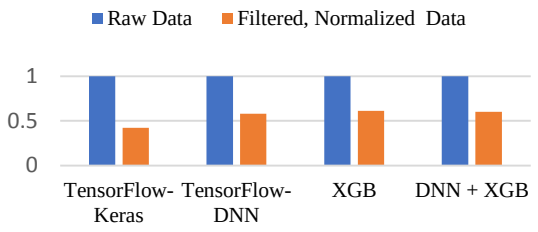

Figure 18. The comparison of models using filtered- normalized data (case 8) and raw data (case 7)

The absolute accuracies in prediction when using raw Micro:bit data to train the models is considerable. Fig. 18 shows the large different model accuracy between case 7 and 8. Observing closely the models in these two cases, the only difference in their data input is that case 7 used raw data while case 8 used filtered-normalized data. The filtering process in case 8 already removed 10% IMFs of lower frequencies while case 7 used directly the raw data. Therefore, the high accuracy obtaining from raw data is due to low frequency drifts. Additionally, another meaningful result is that if we do not combine IMFs but use them directly to fit the model, the model accuracy is also high as shown in case 5. This means even though we remove low frequency components, we can still achieve high accuracy by using IMFs as features. We also realize that XGBoost models have a good performance on all cases of transforming data, and the resemble results of DNN and XGBoost model are also very good. As a result, it generally enhances the reliability of XGBoost models in the classification of brain states using EEG data.

VIII. CONCLUSION

In this project, we have discovered that TensorFlow can be used to identify the brain states using EEG data. The project has surveyed two algorithms from TensorFlow. The difference in model accuracy between the results and from the XGBoost is small though the result from XGBoost is still higher. Additionally, data processing makes an important role in the Keras and DNN algorithm in TensorFlow.

In the usage of both, Neurosky and Micro:bit data in building models, the accuracy of models with data normalized using min-max normalization is always higher, especially in Neurosky data. In Micro:bit data, normalization has more influence when data were decomposed than when data were extracted. In the project, the high accuracy obtaining from the model with the extracted-normalized confirm the importance of Fourier Fast Transformation in classify EEG data. Some other features extracted such as PFD, HFD, Hjorth Parameters, Spectral Entropy, SDV Entropy, Fisher Information, Approximate Entropy, DFA, Hurst Exponent contribute to the very good performance of models as well.

The results from the project also verify that the low frequency drifts in the raw data make an important role to enhance the model accuracy in identifying the EEG data. When the low frequencies were removed from the input data, the model accuracy decreases though the min-max normalization already included to improve its performance (case 8). Moreover, in the usage of EEMD, the results in the project also highlight the application of IMFs as features to build the models for classifying the EEG data. We experienced this in case 5 of Micro:bit data.

Through the project, we can also experience how EEG recorder devices make big impact on the data collection when working on two datasets collected from two different devices. Understanding the pre-processing data inside the devices is very essential to navigate the data processing in a study, especially when working with the datasets from different recorder devices.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Phuong Huynh, Gregory Warner, and Hong Lin conducted the research; Gregory Warner designed the device and collected the data; Phuong Huynh analyzed the data; Phuong Huynh wrote the paper; all authors had approved the final version.

REFERENCES

[1] A. Chan, C. E. Early, S. Subedi, Y. Li, and H. Lin, "Systematic analysis of machine learning algorithms on EEG data for brain state intelligence," in *Proc. IEEE International Conference on Bioinformatics and Biomedicine*, Washington, DC, 2015, pp. 793-799.

[2] S. Siuly, Y. Li, and Y. Zhang, *EEG Signal Analysis and Classification*, Springer International Publishing, 2016, ch. 1, pp. 3-19.

[3] V. Srinivasan, C. Eswaran, and N. Sriraam, "Approximate entropy-based epileptic EEG detection using artificial neural networks," *IEEE Transactions on Information Technology in Biomedicine*, vol. 11, no. 3, pp. 288-295, May 2007.

[4] C. Saitis, *et al.*, "Cognitive load assessment from EEG and peripheral biosignals for the design of visually impaired mobility aids," *Wireless Communications & Mobile Computing*, pp. 1-9, Feb. 2018.

[5] A. Fandango, *Mastering TensorFlow 1.x*., Birmingham, UK: Packt Publishing, 2018.

[6] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[7] B. Lantz, *Machine Learning with R*, Lexington, KY, USA: Packt Publishing, 2018.

[8] F. S. Bao, *et al.*, "PyEEG: An open source Python module for EEG/MEG feature extraction," *Computational Intelligence & Neuroscience*, pp. 1-7, Jan. 2011.

[9] D. Laszuk. PyEMD documentation. [Online]. Available: https://buildmedia.readthedocs.org/media/pdf/pyemd/latest/pyemd.pdf

[10] A. Petrosian, "Kolmogorov complexity of finite sequences and recognition of different preictal EEG patterns," in *Proc. Eighth IEEE Symposium on Computer-Based Medical Systems*, Lubbock, TX, USA, 1995, pp. 212-217.

[11] A. Zeiler, *et al.*, "Empirical mode decomposition - An introduction," in *Proc. International Joint Conference on International Joint Conference on Neural Networks*, 2010, p. 1.

Phuong T. Huynh earned her Bachelor and Master's degree in electrical engineering at HoChiMinh City University of Technology. She has twenty years of experience in designing electrical systems in utility and oil and gas projects such as the Sonaref Refinery (Angola), the Jack and St. Malo Project (Gulf of Mexico), and the BP Toledo Refinery (Ohio). In 2019, Ms. Huynh received her Master of Science in Data Analytics at the University of Houston Downtown. Ms. Huynh's research currently focuses on machine learning and statistical applications.

Gregory N. Warner earned his bachelor's degree in computer science at the University of Houston – Downtown in 2018. He has a total of six years experience working in chemical research and development. This includes 2 years as a Lab Technician at Dow Chemical and 4 years as a Research Technician for the Merichem Company. Mr. Warner is currently focused on human-centered computing.

Hong Lin earned his PhD in Computer Science at the University of Science and Technology of China, and was a postdoctoral research associate at Purdue University, an assistant research officer at the National Research Council, Canada, and a software engineer at Nokia, Inc. Dr. Lin is currently a professor with the University of Houston – Downtown. His research interests include parallel/distributed computing, data analytics, and human-centered computing. He is a senior member of the ACM.