CNN Classification of Finger Movements using Spectrum Analysis of sEMG Signals

Raya Sakashita Worasawate
Kasetsart University Laboratory School
Center for Educational Research and Development, Bangkok,
Thailand
rworasaw@gmail.com

Pined Laohapiengsak
School of Information Science and Technology, Vidyasirimedhi Institute of Science and Technology, Rayong,
Thailand
pined.laohapiengsak@gmail.com

Muthita Wangkid
Kasetsart University Laboratory School
Center for Educational Research and Development, Bangkok,
Thailand
muthita.w@gmail.com

***Abstract*—Electromyography (EMG) is a technique for measuring muscle responses or electrical activities in response to stimulation of nerves. EMG recorded by using electrodes attached to skin or surface electrodes is called surface electromyogram (sEMG) signals. sEMG signals contain information related to movements of muscles such as finger movements, arm movements, etc. Utilizing sEMG for a user to control a prosthesis is a non-invasive way. In this work, sEMG data is used to classify finger movements. sEMG signal is divided into small segments of either 200ms or 500ms. Each segment is converted to a two-dimensional array similar to an image using short-time Fourier transform (STFT) with different types of taper windows. STFT using Tukey window gives better accuracy than Hann window. The combination of Hann and Tukey windows gives similar results to Tukey window alone. Ten-fold cross-validation are performed on the proposed method. Accuracies of 85.29 (±0.53)% and 97.78 (±0.38)% are obtained for the window lengths of 200ms and 500ms, respectively.**

Keywords—sEMG, Finger movements, short-time Fourier transform, Hann window, Tukey window

I. Introduction

In recent years, there have been many researches on applications of bioelectrical signals to help people with disabilities improve their quality of life. The bioelectrical signals widely used in research are electroencephalography (EEG) and electromyography (EMG). These bioelectrical signals can be used as input to control a prosthesis. EEG is a technique for measuring electrical activities in the brain. EEG is difficult to record and has low resolutions. EMG is a technique for measuring muscle responses or electrical activities in response to stimulation of nerves. EMG is the most convenient procedure. EMG obtained by surface electrodes or sEMG is preferred because it is a non-invasive procedure in contrast to painful needle electrodes which are used in an invasive EMG method. Using sEMG signals to control a prosthesis is not new. Previous works using sEMG to identify the movement of fingers are discussed here.

Khushaba et al. [1] proposed a majority vote (MV) technique which requires the knowledge of previous decisions. The disjoint windowing scheme with an analysis window length of 100 ms with 9 voting decisions was used to obtain the reported accuracy of 90%. This is equivalent to the overlapping windowing scheme with an analysis window length of 900 ms with the overlap time of 800ms. Thi et al. [2] used 56 features extracted from EMG signals. Data from each subject were used to train the classifiers separately. The overlapping windowing scheme with an analysis window length of 240 ms or more with 0 - 50% overlap time was used to obtain the reported accuracy of 96.08% (± 0.9) which is an average of the model for each subject. In [1] and [2], EMG signals were filtered between 20 and 450 Hz with a notch filter implemented to remove the 50 Hz line interference. Recently, deep convolutional neural networks (CNN) for object recognition have become an active topic. The research by Liu et al. [3] proposed a method to transform an EEG signal into a 2-dimensional representation in time and frequency using the short-time Fourier transform (STFT). The CNN was then used to model the consciousness level.

This work proposed a method to transform a window of sEMG signal as a one-dimensional time series into a 2-dimensional representation of the data in time and frequency similar to the work in [3]. The effect of taper windows on accuracy is studied. Hann window and Tukey window are used in STFT. These window types are chosen because Hann and Tukey are the default window types in scipy.signal.stft() [4] and scipy.signal.spectrogram() [5], respectively. Hann gives a better noise floor but worse frequency resolution than Tukey. The combination of both windows is studied, as well. This proposed method is demonstrated by classifying ten different finger movements from EMG signals. In this work, a publicly available EMG dataset of finger movements provided by [1] was used. In this work, the overlapping windowing scheme with analysis window lengths, t_win, of 200ms and 500ms are used. The time increment, t_inc, is 100ms. In this work, EMG signals are filtered by bandpass filter between 5 and 650Hz [6] with a notch filter implemented to remove the power line interference at 50Hz as demonstrated in [7]. The model trained by 500ms segments gives better results. Tukey window gives better accuracy than Hann window. The combination of both window types gives similar results on accuracy compared to Tukey window alone.

II. Materials and Methods

A. sEMG dataset

In this work, publicly available sEMG finger movements dataset [1] is used to demonstrate the proposed method. In this dataset, ten finger patterns were collected from eight subjects, six males and two females, aged between 20 and 35 years. The data collection process is described in [1]. The EMG data was recorded using two EMG channels (Delsys DE 2.x series EMG sensors). Fig. 1 shows the positions of these electrodes. The EMG signals were sampled at 4000 Hz.

Ten finger patterns shown in Fig. 2 include individual and combined finger movements. The finger arrangements are the following, i.e. the individual fingers as Thumb (T-T), Index (I-I), Middle (M-M), Ring (R-R), Little (L-L) and the combined fingers as Thumb-Index (T-I), Thumb-Middle

(T-M), Thumb-Ring (T-R), Thumb-Little (T-L), Hand-Close (H-C). Each movement was repeated six times with a contraction period of 5 seconds.

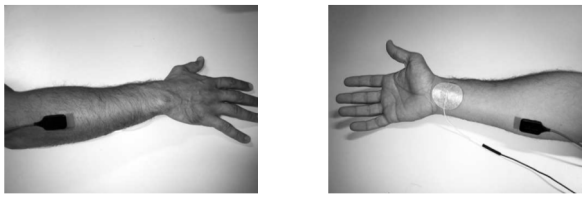


Fig. 1. Electrodes placement on the right forearm [1].

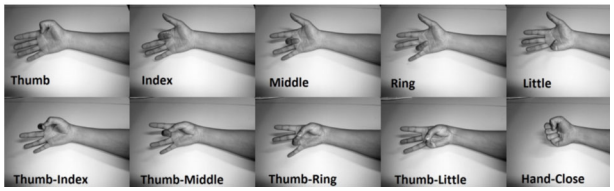


Fig. 2. Different finger movements were collected in dataset [1].

B. *Preprocessing*

The recorded sEMG signals are split into segments with a length of t_win where t_win is either 200ms or 500ms. The t_inc is 100ms. Fig. 3 shows the overlapping window scheme for t_win = 200ms. For t_win = 200ms, we obtain 49 segments for each 5-second recorded file. For the total of 480 recorded files, we obtained 23,500 segments which are 2,350 segments for each class. For t_win = 500ms, we obtain 45 segments for each 5-second recorded file. For the total of 480 recorded files, we obtained 21,600 segments which are 2,160 segments for each class. Each segment is labelled with its class.

There is a signal processing step before converting each segment into a 2-dimensional representation using STFT. It is to filter the signals between 5 and 650Hz with a notch filter at 50Hz according to [6] and [7]. The function scipy.signal.iirfilter() [8] was used for both bandpass and notch filters. The parameter 'btype' was set as 'bandpass' for bandpass filter and as 'bandstop' for notch filter with 5Hz bandwidth. Parameters 'ftype' and 'N' are set as 'butter' and 3, respectively.

Each filtered segment is then passed to scipy.signal.stft() [5] with boundary = None, nperseg = t_win-50ms, noverlap = nperseg-5ms, and nfft = 2000 with sampling frequency is 4000Hz. In the Python code, the units of nperseg and noverlap must be converted into samples by multiplying the time values with the sampling frequency. With the sampling rate of 4000Hz, nfft = 2000 gives a frequency resolution of 2Hz. Only frequency bins from 3 to 303 or frequencies between 6 to 606 Hz are saved for further evaluation. This converts each segment into a 300 by 11 array for each EMG channel.

Two types of window functions i.e. Hann and Tukey windows are used. Hann window gives a better noise floor allowing detection of small frequency components. Tukey window gives a better frequency resolution allowing separation of closed frequency components. Figures 3 and 4 show the output arrays of STFT for t_win = 200ms and 500ms, respectively. Three experiments were performed for each t_win value i.e. 1) only Hann window is used, 2) only Tukey window is used, and 3) the combination of both window is used. For only Hann window, the arrays in (a) and (c) were concatenated and used as training data with its shape of 300 by 22. For only Tukey window, the arrays in (b) and (d) were concatenated and used as training data with its shape of 300 by 22. For the combination of both windows, the arrays in (a), (c), (b), and (d) were concatenated in this order with its shape of 300 by 44 and shown in (e). The labels (a), (b), (c), (d) and (e) are referred to the labels in either Fig. 3 or Fig. 4.

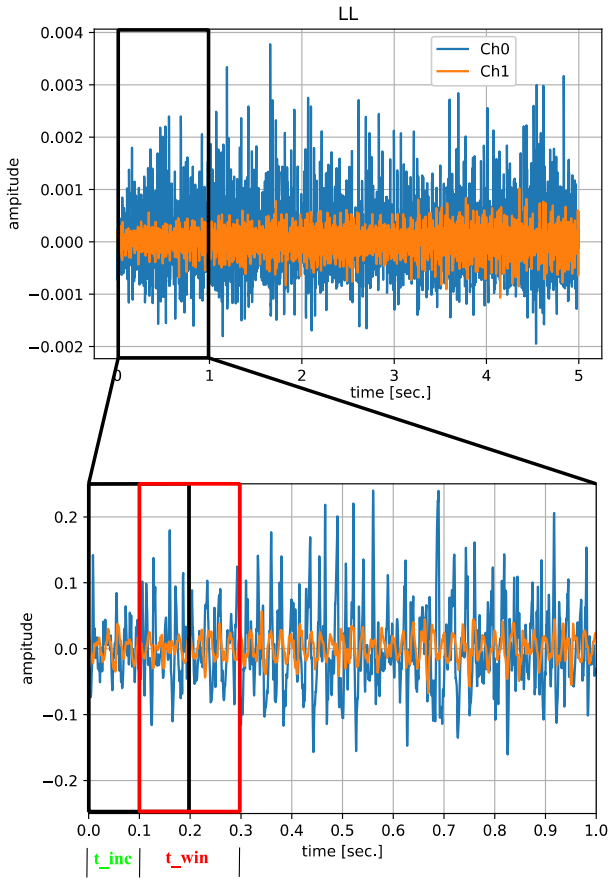


Fig. 3. Overlapping widow scheme used in this research with analysis window lengths, t_win, of 200ms.

C. *CNN model*

ResNet [9] proposed by He et al. was the selected CNN architecture for the experiments because it is a popular CNN architecture for an image classification in recent years. Python module of fastai was used to train the model. There are many variants of ResNet. The available variants in fastai are ResNet18, ResNet34, ResNet50, ResNet101, and ResNet152. ResNet18 and ResNet34 consist of two-layer blocks while ResNet50, ResNet101, and ResNet152 consist of three-layer blocks. ResNet34 was selected because it is the largest model with two-layer blocks. The function cnn_learner() [10] with arch = resnet34 was used to construct CNN model. The function fit_one_cycle() [11] with n_epoch = 20 was used to train the model.

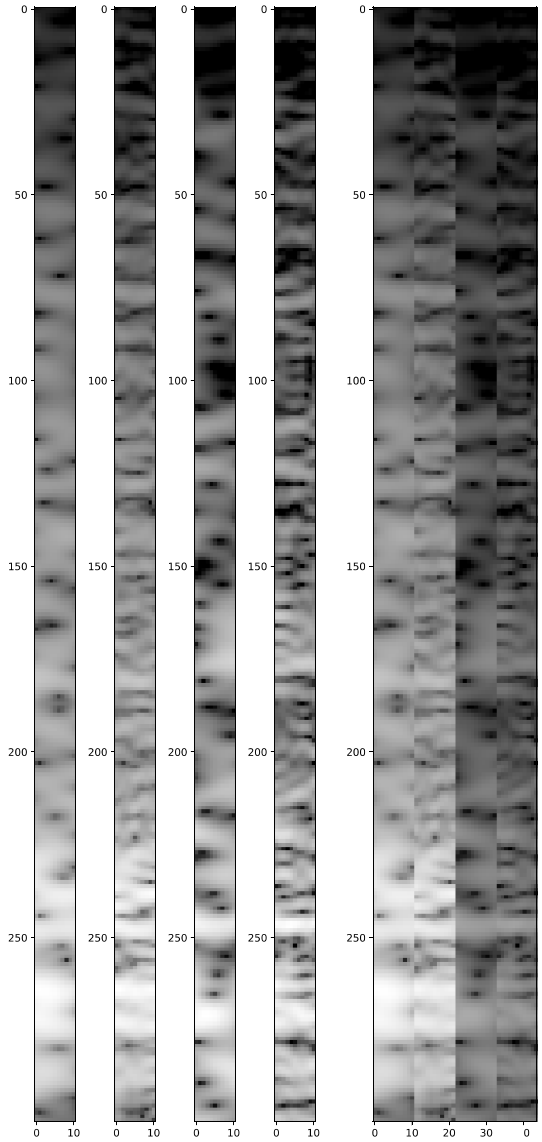


Fig. 4. Two-dimensional representation for t_win = 200ms, (a) Channel 0 with Hann windowing, (b) Channel 0 with Tukey windowing, (c) Channel 1 with Hann windowing, (d) Channel 1 with Tukey windowing, (e) concatenated array.

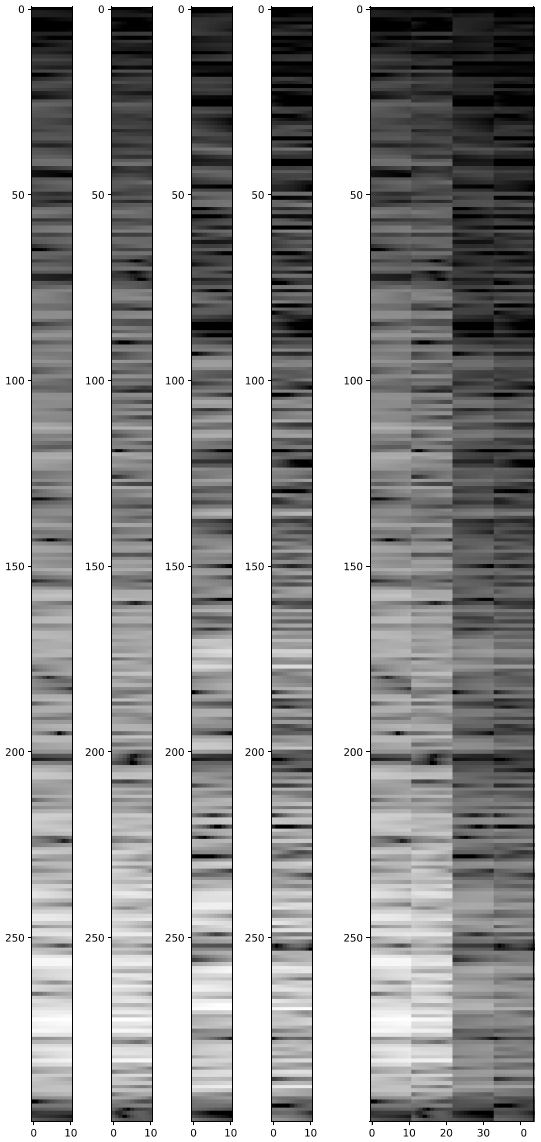


Fig. 5. Two-dimensional representation for t_win = 500ms, (a) Channel 0 with Hann windowing, (b) Channel 0 with Tukey windowing, (c) Channel 1 with Hann windowing, (d) Channel 1 with Tukey windowing, (e) concatenated array.

D. k-fold cross validation

k-fold cross validation is a procedure to evaluate the performance of the model by splitting the dataset into *k* folds. One of *k* folds is selected as a test set and the rest is a train set. The procedure is repeated until every fold is selected as a test set. The classification report is computed and recorded each time. The average and the standard variation of the classification matrix are computed as the representative of the performance of the model. In this work, $k = 10$ was used. Python module of scikit-learn was used to randomly split the whole dataset into 10 folds by applying the function sklearn.model_selection.StratifiedKFold() [12].

E. Simulation of a real-time environment

Twenty second sEMG signals were simulated by randomly selecting finger patterns from all available files of each subject. Each pattern is associated with a file. The start index of the data in the selected file is randomly chosen. The length of the signal is uniformly selected between 0.5 second and 1.5 second of which duration people change their finger patterns. Each block of the selected signals is concatenated to create a 20-second signal. Fig. 6(a) shows the simulated signal of one of the subjects.

III. Results and Discussion

The accuracy of each experiment is reported in Table I. The results show that the longer segment of 500ms gives a better accuracy as expected. Applying the conventional STFT method to 200ms segments using either Hann or Tukey window gives accuracy less than 90% because shorter time window has less information. STFT using Tukey window gives better accuracy than Hann windows. Combining the arrays obtained from both windows as a dataset gives similar results to the ones obtained by Tukey window alone. The combination of Hann and Tukey windows gives accuracy of 85.29 (±0.53)% compared to 85.21(±0.73)% obtained by Tukey alone. This shows that information in data from Hann window is already in data from Tukey window. Similar results can be seen with the data with 500ms segments. Combining the arrays obtained a similar accuracy of 97.68(±0.42)% compared to 97.79(±0.39)% for only Tukey window was used.

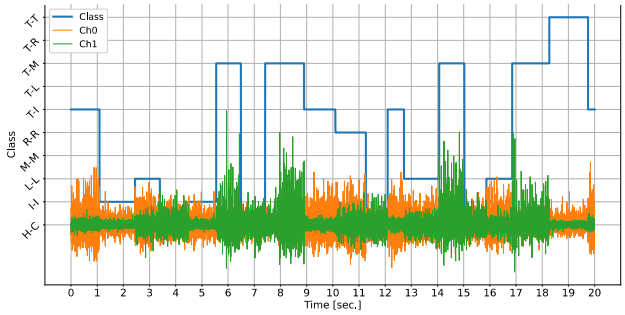
TABLE I. Accuray of Proposed Methods

t_win [ms]	Type of windowing	Model's accuracy [%]
200	Hann	81.46(±0.76)
200	Tukey	85.21(±0.73)
200	Hann+Tukey	85.29 (±0.53)
500	Hann	94.34(±0.44)
500	Tukey	97.79(±0.39)
500	Hann+Tukey	97.68(±0.42)

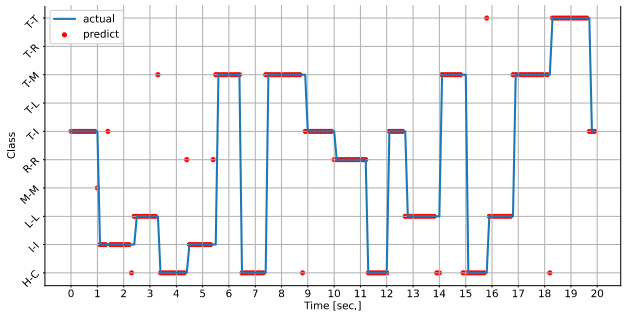
Table II shows the precision of each class of the models. It shows that the combination of arrays obtained from both windows gives similar results to Tukey window for all classes. Class H-C or Hand Close has the highest precision as expected because all fingers are active. Class L-L or Little has the lowest precision.

Table III shows one of the confusion matrix of the model using 200ms with dataset from both windows. It shows that 8 out of 9 classes could get wrong predictions as class L-L. This gives class L-L the lowest precision. Thumb and the index finger create the most confusion as highlighted in Table III. Class T-T is confused with class I-I, and vice versa. Class T-I is also confused with class I-I but only in one direction. Class I-I has lower confusion with class T-I. Class L-L also has high confusion with class T-L but lower confusion in the opposite direction.

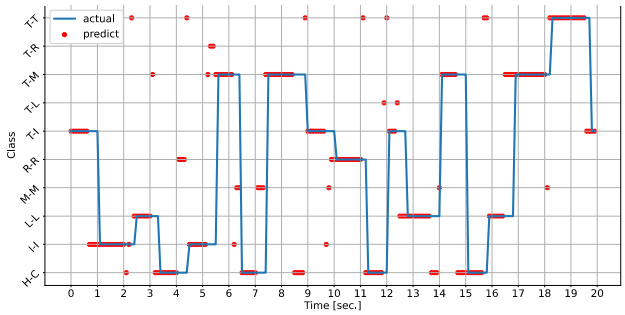
Simulation of a real-time environment was performed by the method described in Sec. II (E). The predicted classes compared to the actual ones obtained by 200ms and 500ms models are shown in Fig. 6(b) and (c), respectively. The error rate defined by 100 times the number of incorrect predictions divided by the number of predictions is shown in Table IV. The 200ms model has a lower error rate than the 500ms model. This shows that the model with lower accuracy could give better prediction because longer window length would give more prediction error where the transition occurred. Shorter window length gives better prediction for faster finger movement.



(a)



(b)



(c)

Fig. 6. (a) Simulated realtime sEMG signals, (b) predicted results from 200ms model, and (c) predicted resultes from 500ms model.

IV. Conclusions

In this paper we proposed CNN classification of finger movements using spectrum analysis of sEMG signals. Data of two EMG channels are converted to two dimensional array with two types of taper window i.e. Hann and Tukey. Analysis window lengths of 200ms and 500ms are used in the experiments. The results show that longer window length gives better accuracy. The spectrograms obtained from Tukey window gives better accuracy than Hann window. Using the spectrograms from both taper windows does not improve accuracy of the model compared to Tukey window alone. Finally, the experiment shows that the model with shorter window length gives better prediction in the real situation. The contribution of this work is to show the importance of taper windows. This work also shows that the model with better accuracy may not have better prediction results for the real-time signals. This proposed method could be applied to other types of hand gesture recognition using sEMG. The effects of the window length and size of CNN models on accuracy can be further studied.

TABLE II. PRECISION OF EACH CLASS OF PROPOSED METHODS

Class	200ms Hann [%]	200ms Tukey [%]	200ms Hann+ Tukey [%]	500ms Hann [%]	500ms Tukey [%]	500ms Hann+ Tukey [%]
H-C	88.43 (1.32)	92.17 (2.28)	91.41 (1.83)	97.78 (1.1)	99.21 (0.32)	99.22 (0.68)
I-I	77.86 (2.73)	82.55 (1.34)	81.53 (1.23)	92.61 (1.21)	96.46 (1.41)	96.68 (1.23)
L-L	72.16 (2.31)	76.85 (2.62)	77.22 (2.25)	90.28 (2.27)	95.88 (1.48)	94.93 (0.86)
M-M	84.89 (2.64)	88.54 (1.21)	88.71 (1.91)	96.32 (1.13)	98.75 (0.71)	98.94 (0.83)
R-R	86.96 (1.91)	88.86 (1.7)	88.67 (1.61)	96.75 (1.31)	98.99 (0.67)	99.08 (0.68)
T-I	78.98 (2.61)	83.35 (2.98)	82.98 (1.74)	92.73 (1.26)	96.74 (1.15)	97.59 (0.79)
T-L	77.37 (2.96)	81.18 (1.55)	83.13 (3.35)	91.65 (2.14)	96.94 (1.66)	96.29 (1.34)
T-M	79.92 (1.71)	83.67 (2.16)	84.91 (2.11)	93.96 (1.99)	98.26 (0.97)	97.53 (1.31)
T-R	87.18 (2.23)	90.19 (1.24)	90.96 (2.91)	97.3 (0.67)	99.03 (0.76)	99.03 (0.55)
T-T	80.82 (3)	84.74 (2.3)	84.09 (2.05)	94 (1.73)	97.66 (1.1)	97.48 (0.73)

TABLE III. CONFUSION MATRIX OF THE MODEL USING 200MS WITH DATASET FROM BOTH WINDOWS.

		Predicted classes									
		HC	II	LL	MM	RR	TI	TL	TM	TR	TT
Actual classes	HC	205	0	0	0	5	0	0	1	5	0
	II	0	192	5	4	0	3	0	0	0	12
	LL	0	7	183	1	0	6	10	5	1	3
	MM	0	3	2	198	2	2	0	3	1	5
	RR	3	0	1	1	205	0	0	0	6	0
	TI	0	11	6	2	1	192	0	4	0	0
	TL	1	0	3	0	0	3	202	6	1	0
	TM	2	1	3	1	2	7	5	195	0	0
	TR	4	0	1	0	4	0	1	0	206	0
	TT	0	11	1	4	1	2	2	0	0	195

TABLE IV. ERROR RATE WITH DATASET FROM BOTH WINDOWS [%]

ms	S1	S2	S4	S5	S6	S7	S9	S10	avg
200	10.5	11.5	13.0	13.5	15.0	13.5	18.5	11.0	13.7
500	29.5	27.0	26.0	25.0	28.5	33.5	21.5	21.5	26.6

ACKNOWLEDGMENT

This work is supported by AI Builders project established by Vidyasirimedhi Institute of Science and Technology (VISTEC) and Central Tech. Thailand.

REFERENCES

[1] R. N. Khushaba, S. Kodagoda, M. Takruri, and G. Dissanayake, "Toward improved control of prosthetic fingers using surface electromyogram (EMG) signals," Expert Syst. Appl., vol. 39, no. 12, pp. 10731–10738, 2012.

[2] T. L.-. N. Thi, P. V. Ho, and T. Van Huynh, "A study of finger movement classification based on 2-sEMG channels," in 2020 7th NAFOSTED Conference on Information and Computer Science (NICS), 2020.

[3] Q. Liu et al., "Spectrum analysis of EEG signals using CNN to model patient's consciousness level based on anesthesiologists' experience," IEEE Access, vol. 7, pp. 53731–53742, 2019.

[4] "scipy.signal.stft — SciPy v1.7.1 Manual," Scipy.org. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.stft.html. [Accessed: 01-Sep-2021].

[5] "scipy.signal.spectrogram — SciPy v1.7.1 Manual," Scipy.org. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.spectrogram.html. [Accessed: 01-Sep-2021].

[6] "Standards for Reporting EMG Data," J. Electromyogr. Kinesiol., vol. 36, pp. I–II, 2017.

[7] K. Strzecha et al., "Processing of EMG signals with high impact of power line and cardiac interferences," Appl. Sci. (Basel), vol. 11, no. 10, p. 4625, 2021.

[8] "scipy.signal.iirfilter — SciPy v1.7.1 Manual," Scipy.org. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.iirfilter.html. [Accessed: 09-Sep-2021].

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[10] "Learner for the vision applications," Fast.ai. [Online]. Available: https://docs.fast.ai/vision.learner.html. [Accessed: 09-Sep-2021].

[11] "Hyperparam schedule," Fast.ai. [Online]. Available: https://docs.fast.ai/callback.schedule.html. [Accessed: 09-Sep-2021].

[12] "sklearn.model_selection.StratifiedKFold — scikit-learn 0.24.2 documentation," Scikit-learn.org. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html. [Accessed: 09-Sep-2021].