Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 184 (2021) 99–106

Procedia Computer Science

www.elsevier.com/locate/procedia

The 12th International Conference on Ambient Systems, Networks and Technologies (ANT)
March 23 - 26, 2021, Warsaw, Poland

pyEDA: An Open-Source Python Toolkit for Pre-processing and Feature Extraction of Electrodermal Activity

Seyed Amir Hossein Aqajari[a,*], Emad Kasaeyan Naeini[b], Milad Asgari Mehrabadi[a], Sina Labbaf[b], Nikil Dutt[a,b,d], Amir M. Rahmani[a,b,c]

[a] *University of California, Irvine, Department of Electrical Engineering and Computer Science, Irvine, California, USA*
[b] *University of California, Irvine, Department of Computer Science, Irvine, California, USA*
[c] *University of California, Irvine, School of Nursing, Irvine, California, USA*
[d] *University of California, Irvine, Department of Cognitive Sciences, Irvine, California, USA*

Abstract

Physiological response is an automatic reaction that triggers a physical response to a stimulus such as stress, emotion, pain, etc. Examples include changes in heart rate, respiration, perspiration, and eye pupil dilation. Electrodermal Activity (EDA), also known as Galvanic Skin Response (GSR), measures changes in perspiration by detecting the changes in electrical conductivity of skin. Previous studies have already shown that EDA is one of the leading indicators for a stimulus. However, the EDA signal itself is not trivial to analyze. To detect different stimuli in human subjects, variety of features are extracted from EDA signals such as the number of peaks, max peak amplitude, to name a few, showing the prevalence of this signal in bio-medical as well as ubiquitous and wearable computing research. In this paper, we present an open-source Python toolkit for EDA signal preprocessing and statistical and automatic feature extraction. To the best of our knowledge, this is the first effort for developing a versatile and generic tool to extract any number of automatic features from EDA signals. We evaluate our toolkit using different machine learning algorithms applied to the Wearable Stress and Affect Detection (WESAD) dataset. Our results show higher validation accuracy for a stress detection task using the the features automatically extracted by pyEDA.

Keywords: Health Monitoring; Internet of Things; Electrodermal Activity; Galvanic Skin Response; Physiological signals; Wearable Electronics; Machine Learning; Autoencoders; Convolutional Neural Networks; Open-Source; Toolkit

* Corresponding author. Tel.: +1-949-537-9373
E-mail address: saqajari@uci.edu

10.1016/j.procs.2021.03.021

1. Introduction

The human body generates a number of physiological signals that can be used to extract valuable information which measures the functional state of various physiological systems [1]. Many studies have validated the effect of different stimuli on the functioning of the physiological systems in the human body, such as emotion, stress, and sleep [2–6].

Table 1. Comparing the existing toolkits compared with pyEDA for preprocessing and feature extraction of EDA signals.

Related Works	Statistical Features	Automatically Extracted Features	Generic (application-agnostic)	Python
pyEDA	✓	✓	✓	✓
TEAP [7]	✓	×	×	×
PhysioLab [8]	✓	×	×	×
ANSLAB [9]	✓	×	×	×
NeuroKit [10]	✓	×	×	✓
Pysiology [11]	✓	×	×	✓

The Electrodermal Activity (EDA), also known as galvanic skin response (GSR), is one of these physiological signals widely used in biomedical and digital health research to detect certain stimuli in human subjects. EDA refers to changes in sweat gland activity, which reflects the intensity of the individual's emotional state – or emotional arousal [12].

In prior studies, numerous prediction models are constructed to predict a stimulus based on raw EDA signals [13, 14]. The EDA signals collected from commercially available devices are usually raw and have motion artifacts that are common in natural, uncontrolled settings that involve body gestures and movements. Therefore, the raw EDA itself cannot be used in these prediction models directly. First, several signal processing steps are needed to remove noise and extract the clean signal. Next, a variety of features are extracted from the clean EDA signal. These features can then be fed to machine learning algorithms to build prediction models detecting different types of stimuli.

Traditional methods extract statistical features such as the number of peaks, max peak amplitude, average, standard deviation, etc., from the EDA signal for prediction models. However, with rapid development of AI and machine learning algorithms, various automatic methods are implemented to extract automatic features from the signal using neural networks [15, 16]. In many situations, these methods outperform the traditional methods' performance in prediction accuracy. Needless to say, an open-source tool providing automatic and statistical set of features extracted from EDA signal can significantly facilitate the research studies in EDA signal processing. Its worth mentioning that these neural network techniques are not limited only to automatic feature extraction from the signals. Zargari et al. used combination of convolutional neural network and recurrent neural network to accurately track in-mouth nutrient sensors position [17]. Mehrabadi et al. used convolutional neural network to detect COVID-19 in patients with ARDS [18]. Ashrafiamiri et al. used deep neural networks to secure autonomous driving [19].

The existing open-source tools for EDA signal processing only focus on the statistical features and do not take into account the embedding features extracted using automatic methods (Table 1). Soleymani *et al.* [7] develop a toolbox to extract features from different signals including EDA. The author build classifiers solely using statistical features, including: amplitude and number of peaks, mean, and standard deviation of the signal. Furthermore, there are toolboxes that provide integrated software. PhysioLab [8] and ANSLAB [9] are open-source tools for EDA analysis, which are implemented in Matlab. These tools aggregate the information extracted from different signals including EDA. However, the feature extraction module is limited to non-automated statistical features. Besides, researchers have implemented toolboxes for EDA analysis in Python [10, 11]. These toolboxes also have the same limitation since they also only consider statistical features including: number of peaks, amplitude, rise time and decay time. It should be noted that all of these existing toolboxes tailor the feature extraction phase to their target application (e.g., emotion recognition, pain assessment, etc.) making them application-dependent to a certain degree.

In this paper, we present pyEDA [20], an open-source tool in Python with the ability to process EDA signals and extract statistical and automatic features from them. To the best of our knowledge, this is the first work presenting a user-friendly open-source Python tool which can be used to extract any number of automatic features of EDA signals without the need to have a background in artificial neural networks and auto-encoders. Depending on the

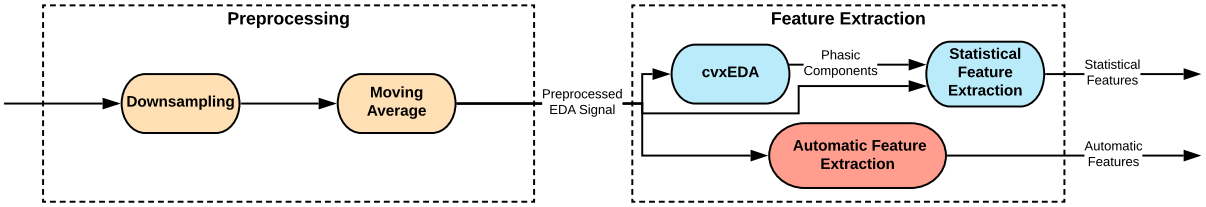

Fig. 1. Proposed Processing Pipeline Architecture for pyEDA.

application, different statistical features might be conceded for acceptable prediction accuracy to a stimulus [21]. In other words, certain features to a stimulus may not be captured using statistical methods. For these reasons, we present a tool to automatically extract any number of automatic features highly correlated to any type of stimuli. Providing such a versatile and generic tool to extract automatic features of the EDA signal is valuable for the health science and technology community. We also demonstrate the efficacy of our tool by using the Wearable Stress and Affect Detection (WESAD) dataset [22] and a set of machine learning algorithms to evaluate both statistical and automatic features extracted using pyEDA. Specifically, this work makes the following key contributions:

- Develop an open-source toolkit to extract any number of features of EDA signals.
- Build a scalable architecture to work with EDA signals with any arbitrary length.
- Enable the toolkit to perform generic automatic feature extraction.
- Evaluate the extracted features using a publicly available dataset on stress assessment.

The rest of this paper is organized as follows. Section 2 briefly outlines the EDA background. Our proposed processing pipeline architecture is presented in Section 3. In Section 4 we evaluate our result using the WESAD data set. Finally, Section 5 concludes the paper.

2. EDA Background

To better understand how EDA is captured, it is helpful to study the physiological characteristics of the skin described in [12]. Sweat glands are small tubular structures of the skin producing sweat. Our body has approximately three million sweat glands having different density across the body. Sweat glands can be found in large numbers on the soles of the feet, the palms and fingers, and on the forehead and cheeks. They produce moisture through pores towards the surface of the skin, whenever they are triggered. When the balance of positive and negative ions in this secreted fluid changes, the electrical current flows more readily. This results in decreased skin resistance, or in other words, increased skin conductance. Electrodermal Activity (EDA) is a term used for this change. EDA is also known as Galvanic Skin Response (GSR), Skin Conductance (SC), Electrodermal Response (EDR), and Psychogalvanic Reflex (PGR).

Although one of the main purposes of sweating is thermoregulation, sweating is also triggered whenever a person is exposed to a stimulus such as emotionally loaded images. This type of sweating is called emotional sweating. Sweat secretion, which reflects the changes in arousal, is driven unconsciously by the automatic nervous system (ANS) in order to meet behavioral requests. A number of commercially available devices (e.g., RespiBAN professional, Empatica E4, Fitbit Sense, and Shimmer3 GSR+) can be used to collect EDA signals.

According to [12], EDA signals consist of two main components: Skin Conductance Level (SCL) and Skin Conductance Response (SCR). The SCL changes slightly on a time scale of tens of seconds to minutes. Depending on hydration, skin dryness or automatic regulation of an individual respondent, the rising and declining SCL is continuously changing. SCL, which is also called the tonic level of EDA signal, can differ significantly across different individuals. Due to this, the actual tonic level on its own is not completely informative. SCR, which is also known as the phasic component of EDA, rides on top of the tonic changes and shows much faster alterations. Variations in the phasic component of a EDA signal are visible as EDA bursts or EDA peaks. The phasic component is sensitive

to specific emotionally arousing stimulus events (event-related SCRs, ER-SCRs). These bursts can occur between 1-5 seconds after the onset of emotional stimuli. Quite the opposite, non-specific skin conductance responses (NS-SCRs) are not a consequence of any eliciting stimulus. These responses happen at a rate of 1-3 per minute spontaneously. There is a need for a flexible and usable processing toolchain that can be used to efficiently analyze these EDA signals. In the following, we describe our pyEDA processing pipeline architecture that is designed to meet these needs.

3. Proposed Processing Pipeline Architecture

Figure 1 shows the proposed processing pipeline architecture for pyEDA to analyze the EDA data. There are two different stages in this pipeline: The pre-processing stage and the feature extraction stage. As shown in this figure, the pre-processing stage consists of two different modules. In the pre-processing stage, the signals are cleaned and prepared for the feature extraction stage. Then, the feature extraction stage uses two different procedures to extract the features from the pre-processed data. We use traditional manual statistical feature extraction as well as automatic feature extraction methods in our proposed architecture. In the following, we explain each stage of the pipeline in detail.

3.1. Pre-processing

In this stage, we use down-sampling and moving averaging to pre-process the data. At the end of this stage, a pre-processed EDA signal is ready and accessible for further analysis and feature extraction.

The EDA data is usually sampled at much higher frequency than needed. Therefore, down-sampling is done to reduce memory footprint and processing time of the data with a negligible risk of losing important information in the signal. In the pre-processing stage, the raw EDA data is down-sampled to the lower sampling rate. Based on the studies conducted in [12], the EDA data can safely be down-sampled to 20 Hz or even less if the data originally was collected at 128 Hz.

A raw EDA signal varies before or after a peak. This is due to individual differences in the tonic component of EDA or due to the noise caused by movements or respiration artifacts [12]. After down-sampling the data, a moving average across a 1-second window is first used to smooth the data and reduce artifacts such as body gestures and movements, which are common in everyday settings.

3.2. Statistical Feature Extraction

The number of peaks, the mean of EDA, and the max peak amplitude are three statistical features extracted in our pipeline. Calculating the mean of EDA is straightforward. To calculate the other two features, we need to extract the EDA peaks that are induced by eliciting stimulus. A number of signal processing steps are required to derive EDA peaks that are a consequence of eliciting stimulus [12]:

1. The phasic component is extracted from the pre-processed EDA signal.
2. A low-pass Butterworth filter is applied on the phasic data to remove line noise. A cutoff frequency of 5 Hz divided by sampling rate is typically used.
3. Onsets and offsets are identified from the phasic data.
4. The maximum amplitude value within each onset-offset is considered as a peak if and only if its difference with the amplitude value at onset is higher than the threshold, which typically is 0.005 μS.

One of the major tasks in analyzing the EDA signal is to correctly extract the phasic component of the signal from the original signal. In this paper, we use the cvxEDA algorithm to decompose the original signal into a sparse phasic component and a smooth tonic component. The cvxEDA algorithm is a novel algorithm presented in [23] for the EDA analysis based on maximum a posteriori probability, convex optimization, and sparsity. This algorithm has a desirable capability of properly describing the activity of the autonomic nervous system in response to affective stimulation. This model describes the recorded EDA as the sum of three terms: the phasic component, the tonic component, and an additive white Gaussian noise term incorporating model prediction errors, measurement errors, and artifacts. We use this algorithm to extract the phasic component of the pre-processed EDA signal for further analysis.

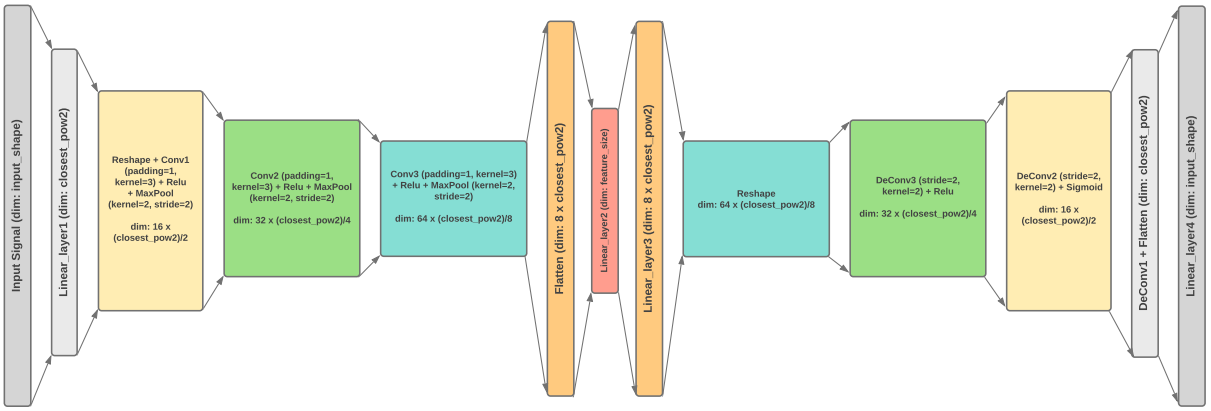

Fig. 2. The architecture of the proposed autoencoder.

To calculate the number of peaks corresponding to events related to SCRs, we need to find the maximum amplitude value within each onset-offset window in the original EDA signal [12]. Onsets are all the points in which the phasic component of the signal crosses above the onset threshold, which is typically 0.01 μS. To find the corresponding offset of the computed onset, we check for the point in which the phasic component of the EDA signal crosses below the offset threshold, which is typically 0 μS. For each window, the time difference between its onset and offset needs to be above the duration threshold of 1s. Any peaks before this duration threshold are considered as nonspecific skin conductance responses and does not need to be extracted.

The clean EDA data, the peaks, and the tonic and phasic components of signal can be used to easily extract extra statistical features if needed.

3.3. Automatic Feature Extraction

Feature extraction becomes increasingly important when the data is high dimensional. There have been several studies which attempt to create classification models based on statistical features extracted from physiological signals such as EDA. However, there is no study suggesting which specific set of statistical features is informative for any type of EDA analytics regardless of the final application (e.g., assessing stress, emotion, pain, risk of seizure, etc.). Therefore, using automatic feature extraction and selection to find the most important set of features for any type of application can be significantly important.

An autoencoder is a type of unsupervised artificial neural network which is used to learn efficient data coding [24, 25]. The aim of an autoencoder is to efficiently learn how to compress and encode the data to reduce the dimensional representation. This lower dimensional representation can be regarded as an abstract set of features of the original high dimensional data. An autoencoder consists of two different parts: the encoder and the decoder. These two parts can be defined as two different functions as follows:

$$\phi : \mathcal{X} \rightarrow \mathcal{Y} \tag{1}$$

$$\psi : \mathcal{Y} \rightarrow \mathcal{X}' \tag{2}$$

Given one hidden layer for encoder and decoder parts in the simplest case: Function (1) can be defined as $\boldsymbol{y} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$. σ is an element-wised activation function such as sigmoid. $\boldsymbol{W}$ is a weight matrix and $\boldsymbol{b}$ is a bias vector of the encoder part. $\boldsymbol{x}$ is $\in \mathcal{X}$ and $\boldsymbol{y}$ is $\in \mathcal{Y}$. Function (2) can be defined as $\boldsymbol{x'} = \sigma(\boldsymbol{W'}\boldsymbol{y} + \boldsymbol{b'})$. σ is an element-wised activation function such as sigmoid. $\boldsymbol{W'}$ is a weight matrix and $\boldsymbol{b'}$ is a bias vector of the decoder part. $\boldsymbol{x'}$ is $\in \mathcal{X}'$ and $\boldsymbol{y}$ is $\in \mathcal{Y}$.

Autoencoders are trained to minimize the following reconstruction loss:

$$loss(\boldsymbol{x}, \boldsymbol{x'}) = ||\boldsymbol{x} - \boldsymbol{x'}||^2 \tag{3}$$

Weights and biases are initialized randomly, and then iteratively updated based on the reconstruction loss computed in (3) during training through Backpropagation. The image $\boldsymbol{y}$ represents a latent code or latent representation of the input $\boldsymbol{x}$. It can be directly used as the set of features of input $\boldsymbol{x}$ for classification.

Figure 2 shows the architecture of the autoencoder implemented in our pipeline for automatic feature extraction. First, a linear layer is used to downsample the input EDA signal to the closest power of two length. This is done to make our tool scalable; thus, the input EDA data with any arbitrary length can be entered to our proposed model. Based on this figure, the encoder consists of three 1d convolutional layers. The output of the encoder is flattened and then downsampled to the latent code size. The output of this linear layer is the lower representation of the input signal which is extracted as the set of automatic features. In the decoder part there are three 1d deconvolutional layers to reconstruct the input signal from the latent code. At the end of this part, the data is flattened and a linear layer is used for reconstruction. The Rectified Linear Units (ReLU) are used here as activation functions in the network. It is worth mentioning that the number of extracted features are fed to our tool as an input parameter before training the model (in this paper the number of features is 64). We use PyTorch library to implement our autoencoder architecture. PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing [26].

4. Results

4.1. Machine Learning based Classification

To demonstrate the efficacy of the pyEDA toolkit, we use it to build machine learning based stress and affect models using the WESAD dataset and evaluate the performance of our extracted features. We use four different machine learning algorithms: (1) K-nearest-neighbor (kNN) with k between 1 to 10, (2) Naïve Bayes Gaussian classifier, (3) Random Forest with depth between 1 to 10, and (4) support vector machine (SVM). The kNN method uses k number of nearest data-points and predicts the result based on a majority vote [27]. The Naïve Bayes Gaussian classifier predicts the result based on the probabilities of each feature's Gaussian distribution [28]. The SVM tries to find the best hyper-plane to divide the data points into different classes [29]. The Random Forest classifier fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting [30]. We use the scikit-learn software for classification and prediction. The scikit-learn is an open-source machine learning library for the Python programming language [31].

4.2. WESAD Dataset

Wearable Stress and Affect Detection (WESAD) is a publicly available multimodal dataset for stress and affect detection [22]. In this data set, physiological and motion data are recorded from Empatica E4 and RaspbiAN professional devices from 15 subjects during a lab study.

The goal of this dataset is to elicit three affective states (neutral, stress, amusement) in the participants. There are two different versions of the study protocol in this dataset. These protocols consist of six different tasks labeled as Baseline, Amuesment, Medi I, Stress, Rest, and Medi II. They distinguish two different classification tasks based on these protocols. First, they define a three-class problem: baseline vs. stress vs. amusement. Second, they define a binary classification: baseline vs. stress. In this paper, we focus on creating a binary classification to detect stress. We consider EDA data in the Baseline section labeled as "not-stressed" (0), and EDA data in the Stress section labeled as "stressed" (1) to create our model. Our model is created based on the EDA data collected from Empatica E4 wristband [32].

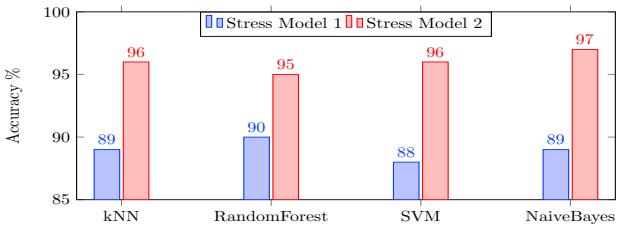

Fig. 3. Validation accuracy on two different stress models using different machine learning algorithms.

4.3. Stress Models Accuracy

We build our stress model based on four different machine learning algorithms (kNN, Naive Bayes, Random Forest, and SVM). We train two different models based on the input features for each machine learning algorithm and report their accuracy: (1) The first model trains the data using only statistical features. (2) The second model trains the data using only automatic features. We consider 30 percent of the data (5 subjects) as test and the rest as training. Figure 3 shows the accuracy of the four different classifiers used based on two different stress models. The best accuracy for the first model belongs to the random forest with depth equals to 1, which is 90%. The best accuracy for the second model belongs to the NaiveBayes classifer which is equal to 97%. In [22], they achieve the accuracy of 79.71% on the binary classification for EDA signals collected from Empatica E4 wristband. The results show that our proposed pipeline outperforms their method in extracting related features for creating stress models (for both automatic and statistical features).

Our results show that, in all four machine learning algorithms, we achieve a higher accuracy using automatically extracted features compared with statistical features. To detect other types of stimulus than stress, one might need to add some extra statistical features to achieve an acceptable accuracy. However, automatic feature extraction module in our tool can still be used to extract the most important features regardless of the stimulus. This is the main advantage of our open-source tool in comparison with other tools for EDA feature extraction.

5. Conclusion

We presented pyEDA, a user-friendly open-source toolkit in Python, to extract statistical and automatic features from EDA data. To the best of our knowledge, this is the first work presenting an automatic feature extraction module in an open-source tool for EDA data. There is no previous study claiming what statistical features of EDA data are the best features to detect different types of stimuli. Our tool uses autoencoders to automatically extract any arbitrary number of features representing the lower dimensional representation of the input data. These features can directly be used in machine learning algorithms for classification and stimulus detection. Different types of stimuli can have different effects on EDA signal, therefore the type of statistical features to extract from the signal can be different in each scenario. As a result, presenting a toolkit which is able to extract both statistical and automatic features of EDA data can facilitate and accelerate research in EDA signal analysis.

Acknowledgements

This work was supported in part by the Academy of Finland through the SLIM Project under the grants 316810 and 316811, and in part by the US National Science Foundation through the UNITE Project under the grant SCC CNS-1831918.

References

[1] BIOMEDIKAL.IN, [Online]. http://biomedikal.in/2011/05/important-physiological-signals-in-the-body/. Accessed: Oct-2020.

[2] Lan Li and Ji-hua Chen. Emotion recognition using physiological signals. In *International Conference on Artificial Reality and Telexistence*, pages 437–446. Springer, 2006.

[3] Lin Shu, Jinyan Xie, Mingyue Yang, Ziyi Li, Zhenqi Li, Dan Liao, Xiangmin Xu, and Xinyi Yang. A review of emotion recognition using physiological signals. *Sensors*, 18(7):2074, 2018.

[4] Hee Jeong Han, Sina Labbaf, Jessica L Borelli, Nikil Dutt, and Amir M Rahmani. Objective stress monitoring based on wearable sensors in everyday settings. *Journal of Medical Engineering & Technology*, 44(4):177–189, 2020.

[5] Ali Rostami, Vaibhav Pandey, Nitish Nag, Vesper Wang, and Ramesh Jain. Personal food model. In *Proceedings of the 28th ACM International Conference on Multimedia*, MM '20, page 4416–4424, New York, NY, USA, 2020. Association for Computing Machinery.

[6] Milad Asgari Mehrabadi, Iman Azimi, Fatemeh Sarhaddi, Anna Axelin, Hannakaisa Niela-Vilén, Saana Myllyntausta, Sari Stenholm, Nikil Dutt, Pasi Liljeberg, and Amir M Rahmani. Sleep tracking of a commercially available smart ring and smartwatch against medical-grade actigraphy in everyday settings: Instrument validation study. *JMIR mHealth and uHealth*, 2020.

[7] Mohammad Soleymani, Frank Villaro-Dixon, Thierry Pun, and Guillaume Chanel. Toolbox for emotional feature extraction from physiological signals (teap). *Frontiers in ICT*, 4:1, 2017.

[8] John Edison Muñoz, Elvio Rubio Gouveia, Mónica S Cameirão, and Sergi Bermúdez i Badia. Physiolab-a multivariate physiological computing toolbox for ecg, emg and eda signals: a case of study of cardiorespiratory fitness assessment in the elderly population. *Multimedia Tools and Applications*, 77(9):11521–11546, 2018.

[9] Jens Blechert, Peter Peyk, Michael Liedlgruber, and Frank H Wilhelm. Anslab: Integrated multichannel peripheral biosignal processing in psychophysiological science. *Behavior Research Methods*, 48(4):1528–1545, 2016.

[10] Dominique Makowski, Tam Pham, Zen, Jan C. Brammer, Duy Le, Hung Pham (Pham Tien Hùng), François Lesspinasse, Chuan-Peng Hu, and Christopher Schölzel. neuropsychology/neurokit: 0.0.6, January 2020.

[11] Giulio Gabrieli, Atiqah Azhari, and Gianluca Esposito. Pysiology: A python package for physiological feature extraction. In *Neural Approaches to Dynamics of Signal Exchanges*, pages 395–402. Springer, 2020.

[12] Galvanic Skin Response. The complete pocket guide. *Imotions–Biometric Research, Simplified*, 2017.

[13] Pamela Zontone, Antonio Affanni, Riccardo Bernardini, Alessandro Piras, and Roberto Rinaldo. Stress detection through electrodermal activity (eda) and electrocardiogram (ecg) analysis in car drivers. In *2019 27th European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE, 2019.

[14] Busra T Susam, Murat Akcakaya, Hooman Nezamfar, Damaris Diaz, Xiaojing Xu, Virginia R de Sa, Kenneth D Craig, Jeannie S Huang, and Matthew S Goodwin. Automated pain assessment using electrodermal activity data and machine learning. In *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 372–375. IEEE, 2018.

[15] Dian Yu and Shouqian Sun. A systematic exploration of deep neural networks for eda-based emotion recognition. *Information*, 11(4):212, 2020.

[16] Beanbonyka Rim, Nak-Jun Sung, Sedong Min, and Min Hong. Deep learning in physiological signal data: A survey. *Sensors*, 20(4):969, 2020.

[17] Amir Hosein Afandizadeh Zargari, Manik Dautta, Marzieh Ashrafiamiri, Minjun Seo, Peter Tseng, and Fadi Kurdahi. Newertrack: Ml-based accurate tracking of in-mouth nutrient sensors position using spectrum-wide information. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):3833–3841, 2020.

[18] Milad Asgari Mehrabadi, Seyed Amir Hossein Aqajari, Iman Azimi, Charles A Downs, Nikil Dutt, and Amir M Rahmani. Detection of covid-19 using heart rate and blood pressure: Lessons learned from patients with ards. *arXiv preprint arXiv:2011.10470*, 2020.

[19] Marzieh Ashrafiamiri, Sai Manoj Pudukotai Dinakarrao, Amir Hosein Afandizadeh Zargari, Minjun Seo, Fadi Kurdahi, and Houman Homayoun. R2ad: Randomization and reconstructor-based adversarial defense on deep neural network. In *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, pages 21–26, 2020.

[20] pyEDA, [Online]. https://github.com/HealthSciTech/pyEDA. Accessed: Oct-2020.

[21] Heera Lee and Andrea Kleinsmith. Public speaking anxiety in a real classroom: Towards developing a reflection system. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–6, 2019.

[22] Philip Schmidt, Attila Reiss, Robert Duerichen, Claus Marberger, and Kristof Van Laerhoven. Introducing wesad, a multimodal dataset for wearable stress and affect detection. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, pages 400–408, 2018.

[23] Alberto Greco, Gaetano Valenza, Antonio Lanata, Enzo Pasquale Scilingo, and Luca Citi. cvxeda: A convex optimization approach to electrodermal activity processing. *IEEE Transactions on Biomedical Engineering*, 63(4):797–804, 2015.

[24] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[25] Quoc V Le et al. A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. *Google Brain*, pages 1–20, 2015.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[27] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[28] David J Hand and Keming Yu. Idiot's bayes—not so stupid after all? *International statistical review*, 69(3):385–398, 2001.

[29] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[30] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[32] empatica-e4, [Online]. https://www.empatica.com/research/e4/. Accessed: Oct-2020.