# Development of a Convolutional Neural Network for Generalized Periodic Discharge Classification in Cardiac Arrest Patients: Pilot Study

By

Syphong Ha

B.S., California State University Long Beach, 2019

Thesis

Submitted in partial fulfillment of the requirements for the
Degree of Master of Science in Biomedical Engineering at Brown University

PROVIDENCE, RHODE ISLAND

MAY 2022

**AUTHORIZATION TO LEND AND REPRODUCE THE THESIS**

**As the sole author of this thesis, I authorize Brown University to lend it to other institutions for the purpose of scholarly research.**

Date:_____ _____

Syphong Ha, Author

**I further authorize Brown University to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.**

Date:_____ _____

Syphong Ha, Author

This thesis by Syphong Ha is accepted in its present form by the
Center of Biomedical Engineering as satisfying the
thesis requirement for the degree of Master of Science.

Date_____ _____

Dr. Jason Richards, MD, Advisor

Recommended to the Graduate Council

Date_____ _____

Dr Thomas Serre, Reader

Date_____ _____

Dr. John Simeral, Reader

Approved by the Graduate Council

Date_____ _____

Dr. Marissa Gray, Biomedical Engineering Master's
Program Director

**Acknowledgements**

I would like to thank the staff, fellows, residents, doctors, and EEG technicians at Rhode Island Hospital for all the help on the project. I would also like to thank Dr. Gray for helping me navigate my master's program and helping me every step of the way. To my friends back home, thank you for the support.

To my committee Dr. Serre and Dr. Simeral, thank you for your time and knowledge.

To my family especially my parents, brothers, cousins, and grandma, thank you for the endless love, support, and prayers.

I would like to give a big thanks to Alekh for helping with the machine learning portion of the project, particularly for helping me come up with 1D CNN architecture, and for meeting with me regularly to keep me on track.

I would like to especially thank my friend Shreyas Raman for helping me with the machine learning process and for being there for me when I was alone debugging and running code.

To my friend Benjamin thanks for being there for me when times were hard.

To Dr. Richards, thank you for everything, for teaching me about EEGs, for meeting with me to discuss the project, for talking to me about life, and simply, for letting me work on this project. I appreciate everything you have done for me.

Finally, I would like to dedicate this paper and work to my late grandpa, the person whose words not only taught me about hard work and perseverance but also his actions. Thank you for the unconditional love, support, and wisdom without them I would not be the person I am today. To end, I would like to say a few simple words.

I did it ông.

## Table of Contents

## Table of Figures

## Table of Tables

***Development of a Convolutional Neural Network for Generalized Periodic Discharge Classification in Cardiac Arrest Patients,***

*by Syphong Ha, ScM, Brown University, May, 2022*

**Objective:** To develop a neural network for the binary classification of cardiac arrest patient electroencephalograms (EEGs) between two categories: general periodic discharges (GPDs) or non-GPDs. In addition, two different EEG montages, the longitudinal bipolar and the average reference, were used to train and test the model to see how montages affected classification.

**Methods:** The dataset consisted of 14 neurologist-annotated EEGs from nine different cardiac arrest patients hospitalized at Rhode Island Hospital. The dataset contained 9 EEGs with GPDs and 5 EEGs with a diffusely slow background without GPDs. The data was filtered with a 60 Hz notch filter and a 70 Hz low pass filter before creating the montages and dividing the EEGs into 10 second snippets with a 50% overlap between each snippet. A 1-dimension convolutional neural network was designed and used for the task. One EEG with GPDs and one EEG without GPDs (non-GPDs) were selected to form the test set with the remaining 12 EEGs forming the training set; this process was repeated 45 times until all possible combinations of GPD and non-GPD pairs formed the test set. With 45 different train-test splits, 45 different models were created. Each model's performance was evaluated using the following metrics: accuracy, sensitivity, specificity, precision, F1 score, false negative rate, and false positive rate. The final performance of the model was reported as the median of each respective metric from the models. This process was done for both montages.

**Results:** Using the metrics described in the *Methods* section, the median accuracy, sensitivity, specificity, precision, and F1 score for the longitudinal bipolar montage were 97.46%, 100%, 98.13%, 97.39%, and 97.50% respectively. In the same order as the previous montage's metrics

were listed, the average reference montage's median scores were 97.55%, 100%, 99.44%, 99.43%, and 97.62%.

**Conclusions:** The comparable final metric scores of the two montages show the model's potential at performing the task. However, much more EEG samples are needed for more generalization and to remove the imbalance between GPDs and non-GPDs. Future work includes exploring the frequency domain and testing other machine learning models (RNNs, 2D CNNs, etc.).

## Chapter 1: Introduction

The electroencephalogram (EEG) is a measure of electrical activity in a person's brain and a vital diagnostic tool for neurologists, particularly in unconscious patients. By connecting electrodes to the scalp, the electrical impulses of the brain are captured and recorded. EEGs display these impulses as a graph of the change in voltage over time. A channel is the graph of the potential difference between two electrodes. Chains are a group of channels and arranging these chains in a specific way forms a montage. When reading an EEG, the neurologist utilizes montages to visualize the EEG data and form a clinical impression. (Acharya J. and Acharya V., 2019). EEG recordings can vary in length depending on how long a patient is connected to the EEG electrodes. Routine EEGs are approximately between 20 to 40 minutes in length. Some patients undergo long-term monitoring (LTM) on EEG. These studies are a few hours to several days long (Tatum, 2001). Due to the length of the recordings, LTM EEG analysis and annotation can be a labor-intensive task for neurologists. As a result, research is currently being done on automating analysis through machine learning techniques to assist neurologists in reading and detecting abnormal EEGs.

Because of the dynamic nature of the brain, EEGs can be considered non-stationary time series making them a candidate for machine learning methods (Klonowski, 2009). Delsy et al. (2021) showed recurrent neural networks (RNNs) could be used in binary classification of two non-stationary signals: the sawtooth and burst pattern. Using spectral domain features, they achieved 100% sensitivity, specificity, and accuracy for burst signals and 100% accuracy for sawtooth signals (Delsy et al., 2021). Talathi (2017) utilized a gated recurrent unit RNN trained and tested on the publicly available dataset from Bonn University for seizure and early seizure detection. After training, the model achieved a 99.6% accuracy on the validation set for seizure

detection and approximately 98% accuracy for early seizure detection within 5 seconds of the onset of the seizure (Talathi, 2017). In a similar study, Vidyaratne et al., (2016) used a deep RNN with a bidirectional RNN as its basis. This study used the CHB-MIT EEG database for training and testing. Longitudinal EEG montages were formed and mapped in a 2-dimensional grid-like fashion to process the EEGs for training and testing. After training, the model achieved a 100% sensitivity on seizure detection with an average delay of approximately 7 seconds (Vidyaratne et al., 2016). Gao et al., (2020) proposed using power spectral energy density diagrams, deep convolutional neural networks (CNNs), and transfer learning to classify epileptic EEGs into four classes: interictal, preictal duration to 30 minutes, preictal duration to 10 minutes, and seizures; the network achieved a 92.6% sensitivity for seizure classification and a 97.1% specificity for non-seizure classification (Gao et al., 2020). These studies indicated the predictive potential neural networks had in learning the primary contributing features to detect abnormal EEGs.

During a cardiac arrest, the amount of blood and oxygen flowing to the brain is reduced causing approximately 80% of cardiac arrest patients to become comatose despite being resuscitated (Marion, 2009). As a result, EEG analysis is imperative for these patients. Because of the severity of the condition, cardiac arrest patients' EEGs are monitored long term with LTMs to look for any abnormalities. Abnormal patterns such as isoelectric, low voltage, and burst suppressed EEGs are often associated with poor clinical outcome (Dougherty, 2017). In contrast EEGs displaying reactivity, normal physiological rhythm, or a lack of malignant patterns often indicate a favorable prognosis for patients (Dougherty, 2017*)*. Patterns such as generalized periodic discharges (GPDs), lateralized periodic discharges (LPDs), generalized rhythmic delta activity (GRDA), and lateralized rhythmic delta activity (LRDA) are said to be indeterminate

indicators of prognosis (Dougherty, 2017; Foreman et al., 2012); however, there is debate in the literature with evidence strongly suggesting GPDs are indicators of poor prognosis (Ebersole et al, 2014; Ribeiro et al., 2015). GPDs (Figure 1) are defined by the American Clinical Neurophysiology Society in two parts. The word *generalized* (G) can be defined as occurring in both hemispheres of the brain (Hirsch et al., 2021). The next part *periodic discharge* (PD) is defined as the recurrence of a relatively uniform waveform at regular intervals; the discharge is defined as a waveform lasting less than 0.5 seconds, or, in simple terms a spike (Hirsch et al., 2021). GPDs are described by their prevalence, duration, frequency, sharpness, and other factors (Sully and Husain, 2018). While there may be debate on the prognostic implications of GPDs, research has shown a correlation between GPDs, nonconvulsive seizures (NCSz), and nonconvulsive status epilepticus (NCSE) with the last of the three being strongly associated with mortality; the evolution of GPDs into seizures occurs when the frequency and duration exceed 2.5 Hz and 10 seconds respectively (Foreman et al. 2012; Hirsch et al., 2021).

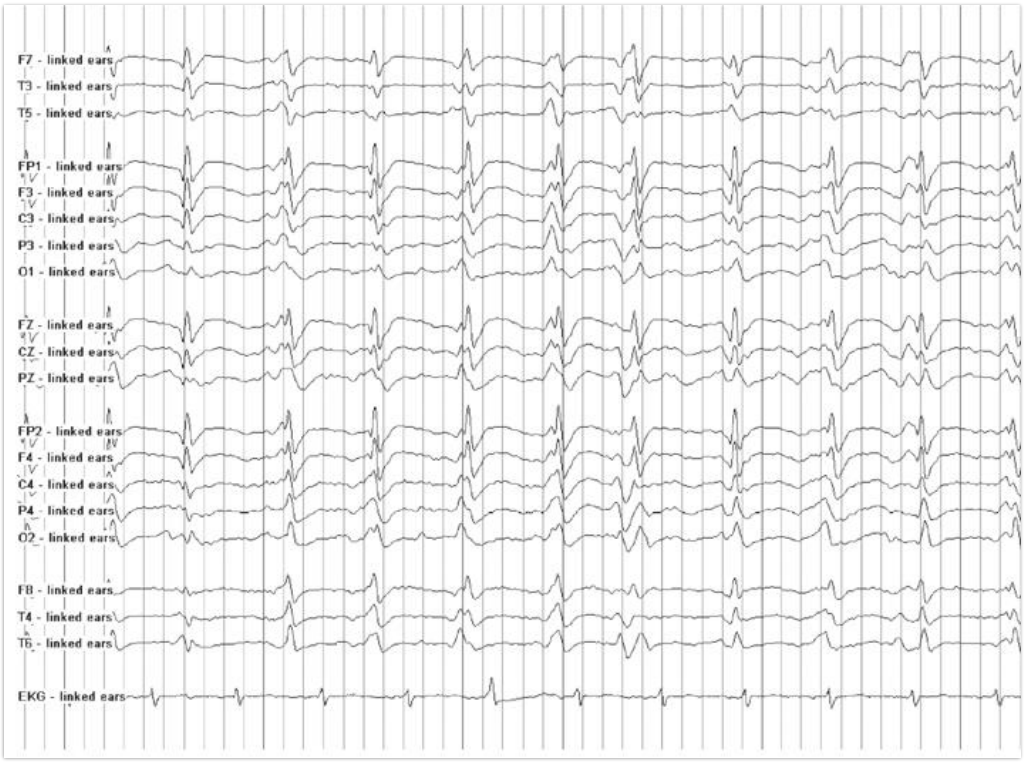


Figure 1: Example of 1 Hz GPDs, Patient Post-Cardiac Arrest (Ebersole et al., 2014, Ch 7)

Thus, this paper's goal is to develop a CNN for binary classification of cardiac arrest patient EEGs into GPD or non GPD before NCSz or NCSE occurs. In addition, the CNN will be trained and tested using two different montages to see if there is any effect on classification.

**References**

Acharya, J. N., & Acharya, V. J. (2019). Overview of EEG montages and principles of localization. *Journal of Clinical Neurophysiology*, *36*(5), 325–329. https://doi.org/10.1097/wnp.0000000000000538

Delsy, T. T., Nandhitha, N. M., & Rani, B. S. (2021). Feasibility of recurrent neural network for the binary classification of non stationary signals. *Microprocessors and Microsystems, 82*, 103955. https://doi.org/10.1016/j.micpro.2021.103955

Dougherty, M. (2017, April). *The utility of EEG in prognosis post-cardiac arrest*. Practical Neurology. Retrieved March 20, 2022, from https://practicalneurology.com/articles/2017-apr/the-utility-of-eeg-in-prognosis-post-cardiac-arrest

Ebersole, J. S., Nordli, D. R., & Husain, A. M. (2014). *Current practice of clinical electroencephalography*. Lippincott Williams & Wilkins.

Foreman, B., Claassen, J., Abou Khaled, K., Jirsch, J., Alschuler, D. M., Wittman, J., Emerson, R. G., & Hirsch, L. J. (2012). Generalized periodic discharges in the critically ill: A case-control study of 200 patients. *Neurology*, *79*(19), 1951–1960. https://doi.org/10.1212/wnl.0b013e3182735cd7

Gao, Y., Gao, B., Chen, Q., Liu, J., & Zhang, Y. (2020). Deep convolutional neural network-based epileptic electroencephalogram (EEG) Signal Classification. *Frontiers in Neurology*, *11*. https://doi.org/10.3389/fneur.2020.00375

Hirsch, L. J., Fong, M. W. K., Leitinger, M., LaRoche, S. M., Beniczky, S., Abend, N. S., Lee, J. W., Wusthoff, C. J., Hahn, C. D., Westover, M. B., Gerard, E. E., Herman, S. T., Haider, H. A., Osman, G., Rodriguez-Ruiz, A., Maciel, C. B., Gilmore, E. J., Fernandez, A., Rosenthal, E. S., … Gaspard, N. (2021). American Clinical Neurophysiology Society's standardized critical care EEG terminology: 2021 version. *Journal of Clinical Neurophysiology*, *38*(1), 1–29. https://doi.org/10.1097/wnp.0000000000000806

Klonowski, W. (2009). Everything you wanted to ask about EEG but were afraid to get the right answer. *Nonlinear Biomedical Physics*, *3*(1). https://doi.org/10.1186/1753-4631-3-2

Marion, D. W. (2009). Coma due to cardiac arrest: Prognosis and contemporary treatment. *F1000 Medicine Reports*, *1*. https://doi.org/10.3410/m1-89

Ribeiro, A., Singh, R., & Brunnhuber, F. (2015). Clinical outcome of generalized periodic epileptiform discharges on first EEG in patients with hypoxic encephalopathy postcardiac arrest. *Epilepsy & Behavior*, *49*, 268–272. https://doi.org/10.1016/j.yebeh.2015.06.010

Sully, K. E., & Husain, A. M. (2018). Generalized periodic discharges. *Journal of Clinical Neurophysiology*, *35*(3), 199–207. https://doi.org/10.1097/wnp.0000000000000460

Talathi, S. (2017). Deep recurrent neural networks for seizure detection and early seizure detection systems. https://doi.org/10.2172/1366924

Tatum, W. O. (2001). Long-term EEG monitoring. *Journal of Clinical Neurophysiology*, *18*(5), 442–455. https://doi.org/10.1097/00004691-200109000-00009

Vidyaratne, L., Glandon, A., Alam, M., & Iftekharuddin, K. M. (2016). Deep recurrent neural network for seizure detection. *2016 International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/ijcnn.2016.7727334

## Chapter 2: Methods

### Data Acquisition

This is a retrospective analysis of EEG data recorded from patients at Rhode Island Hospital between the dates January 1, 2019, and June 30, 2021. The EEGs were recorded by trained EEG technicians at a sampling rate of 256 or 512 Hz using the 10-20 system (Figure 2).

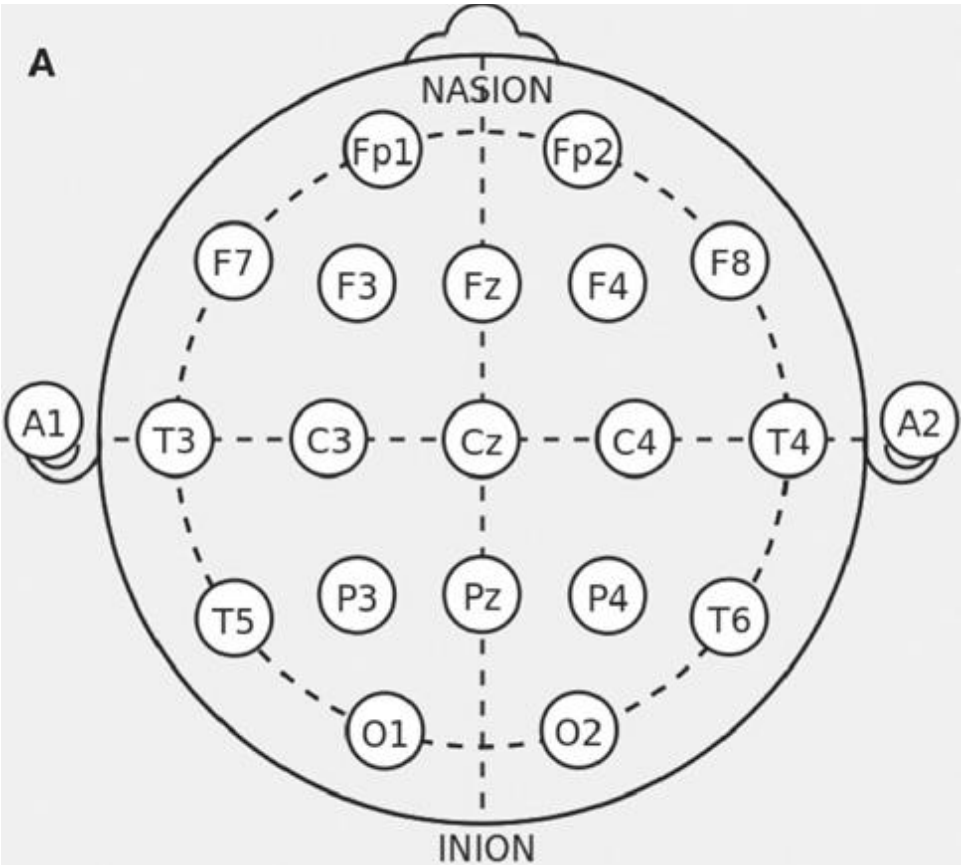


Figure 2: 10-20 System Electrode Placement Reference (Ebersole et al., 2014, Ch 4)

The data were reviewed to find a representative selection of EEG data sets with GPDs and a selection of non-GPD EEGs. GPD EEGs were chosen if the frequency was between 0 to 2.5 Hz, according to neurologist reports, and had a distinguishable peak and amplitude from the background activity. Non-GPD EEGs were chosen to be EEGs with a diffusely slow background (Figure 3). All data collected were approximately one hour long, reviewed, and annotated by a

neurologist beforehand with as few artifacts as possible. In total there were 9 different cardiac arrest patients with a total of 14 EEGs: 9 GPD and 5 non-GPD EEGs. It should be noted leads T3, T4, T5, and T6 were renamed T7, T8, P7, and P8, respectively, in 2006; this new nomenclature was adopted for the current study ("Guideline 5: Guidelines," 2006).


Figure 3: Example of a non-GPD, Diffusely Slow EEG (Ebersole et al., 2014, Ch 13)

**Preprocessing:**

Using the open source Python MNE library for neurophysiological data analysis, the data was read into Python for preprocessing. Electrodes Fp1, F7, T7, P7, O1, Fp2, F8, T8, P8, O2, F3, C3, P3, F4, C4, P4, Fz, Cz, and Pz were selected from the list of available channels interchanging T7, T8, P7, and P8 with T3, T4, T5, and T6 respectively when necessary. The sampling rate was extracted, and any EEGs with a sampling rate of 512 Hz were downsampled to 256 Hz for consistency. A 60 Hz notch filter was used to filter out the power line noise. In

clinical settings it is standard to use a 1 Hz high pass filter (HPF) and a 70 Hz low pass filter (LPF) to create a 1-70 Hz bandpass filter. However, because GPDs have a frequency less than 2.5 Hz, only the 70 Hz LPF was used to minimize the effects of filtering on the GPDs (Libenson, 2010). As such, the EEGs were filtered with a 70 Hz Blackman finite impulse response LPF (Diana et al., 2016).

After filtering, the EEG montages were created from the channels. The bipolar and referential montage types were used in this study. Bipolar montage channels are formed by connecting adjacent electrodes to each other. The longitudinal bipolar montage was used in this study, formed by vertically connecting leads from the anterior region to the posterior region of the scalp: Fp1-F7, F7-T7, T7-P7, P7-O1, Fp2-F8, F8-T8, T8-P8, P8-O2, Fp1-F3, F3-C3, C3-P3, P3-O1, Fp2-F4, F4-C4, C4-P4, P4-O2, Fz-Cz, and Cz-Pz (Figure 4).

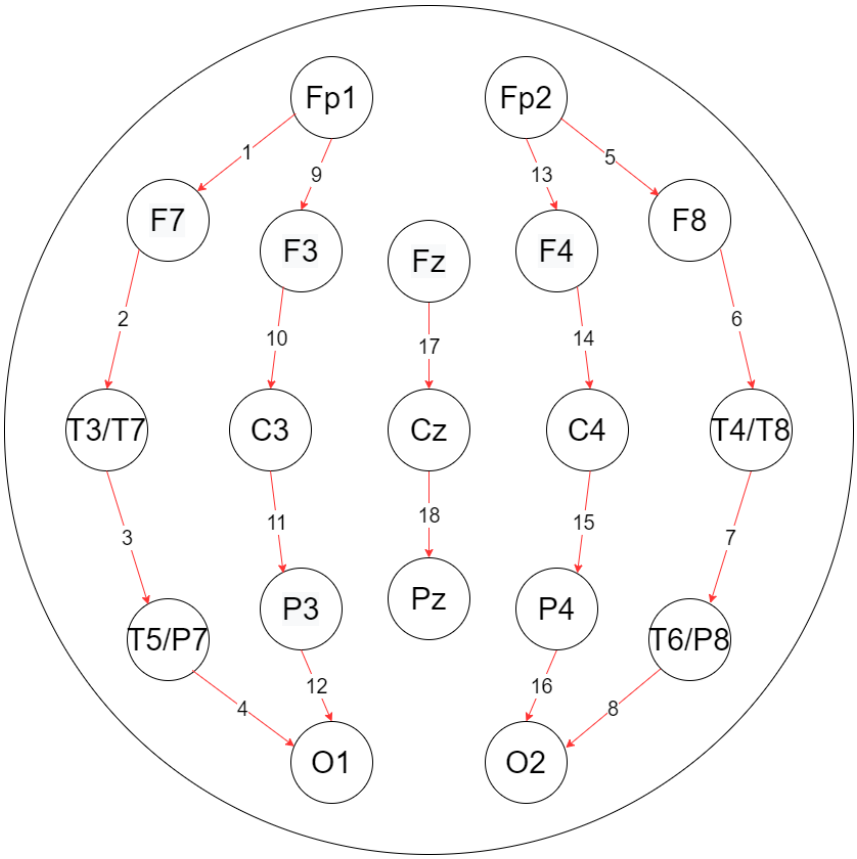


Figure 4: Connections of the 19 Electrodes on the Scalp in the 10-20 System to Form the Longitudinal Bipolar Reference Montage, Numbered in Order of Appearance

Reference or monopolar montages have their channels formed by referencing electrodes to a common point. In this study, the average reference montage was used which is formed by connecting the 19 electrodes to the average of the electrodes (Figure 5).

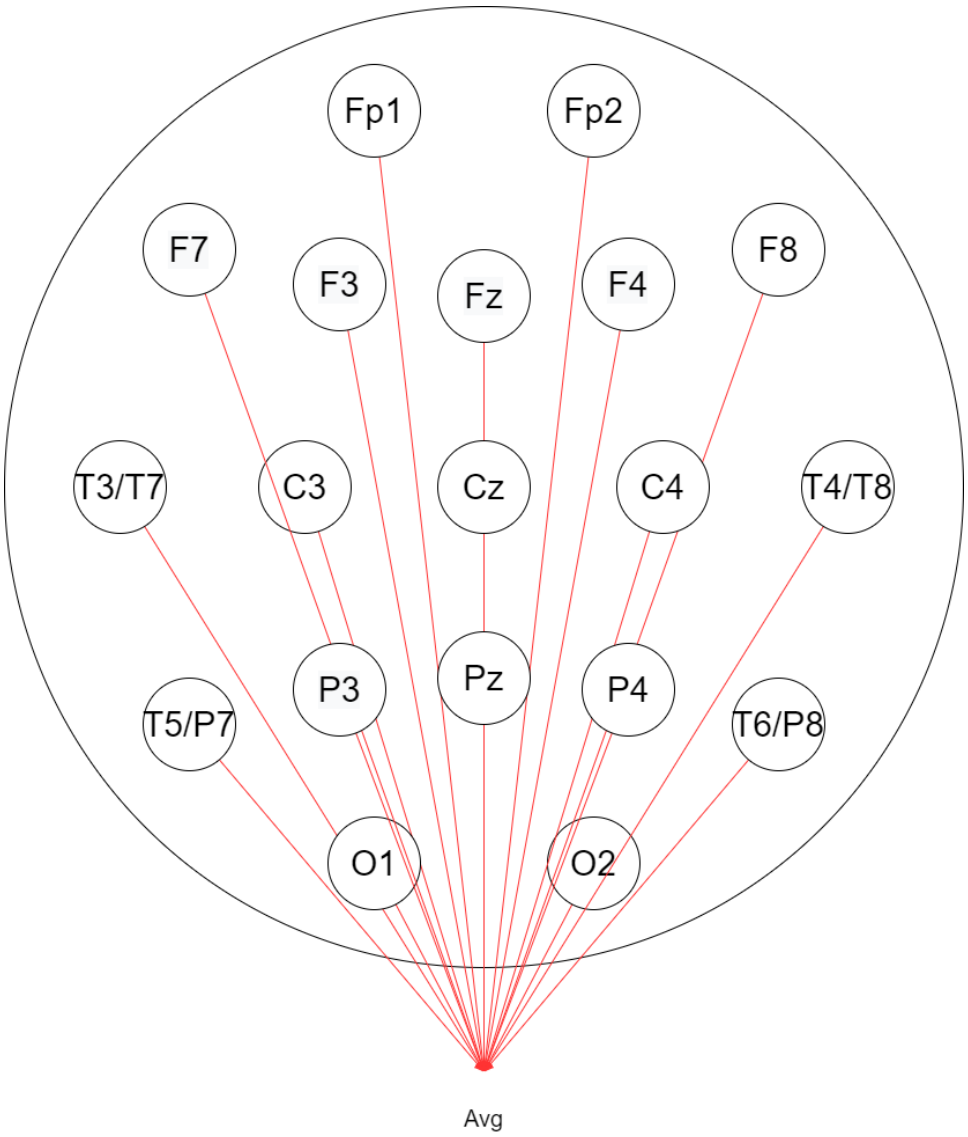


Figure 5: Connections of the 19 Electrodes on the Scalp in the 10-20 System to Form the Average Reference Montage

The arrays from the EEG montages were extracted and divided into 10 second snippets with a 50% (5 second) overlap between each snippet. As seen in the figures above, the average reference montage and the longitudinal bipolar montage have 19 and 18 rows for their matrices respectively, and with a sampling rate of 256 Hz, the final matrix shapes for these snippets were

(19, 2561) or (18, 2561). These snippets were saved in the numpy (.npy) file format to be loaded using the NumPy Python library to train and test the machine learning model.

**Machine Learning Method:**

The machine learning model used for training and testing was a 1D CNN. The 1D CNN was chosen because of the way data is processed—the data is convolved over the individual sequences. This may replicate how a neurologist reads an EEG by looking at all channels of the montage simultaneously but independently to form a diagnosis. Moreover, the 1D CNN has the benefits of being less computationally expensive than its 2D CNN counterpart for similar architectures, being compact with generally up to 2 convolutional layers, and being capable of learning biological signals such as the electrocardiogram (Kiranyaz et al., 2021). For these reasons, the 1D CNN was implemented to test its ability to learn this task.

The 1D CNN used cross entropy as the loss function and stochastic gradient descent (SGD) as the optimizer. The model consisted of two sequential convolutional blocks, a dropout layer, and a final dense layer for selection. Each convolutional block consisted of a 1D convolutional layer (Conv1D) activated by a leaky rectified linear unit (ReLU) function, followed by a batch normalization layer for stabilization, and a max pooling layer to scale down the number of model parameters and improve model generalization. The input shape to the first convolutional layer was 18 for the longitudinal bipolar montage and 19 for the referential montage. After the second convolutional block, the output was reshaped for use in the dense layer at the end. After reshaping, a dropout layer was used to adjust model complexity as needed and set to a rate of 30%. The last dense layer outputted to two nodes and was activated by the softmax function to generate a probability distribution over GPD non-GPD class predictions. The

predicted label was the argmax of the two nodes where node 0 and node 1 represented non-GPD and GPD respectively. The completed architecture for both montages can be seen in Figure 6.

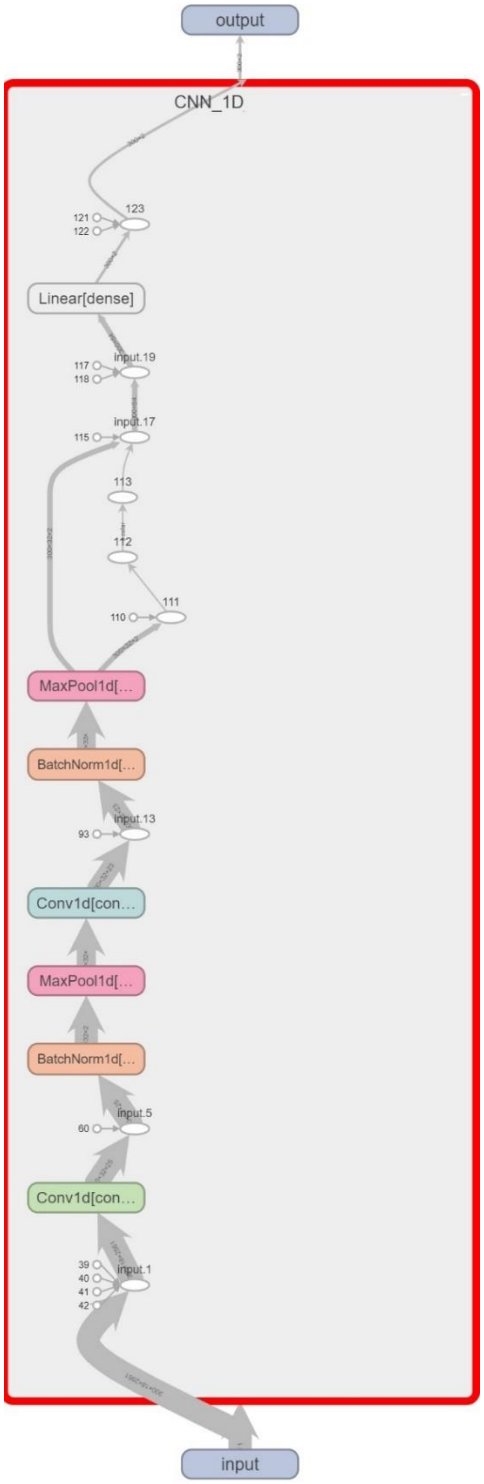

Figure 6: CNN Architecture. The Input Shape to the First 1D Convolutional Layer is Dependent on the Montage; Longitudinal Bipolar Montage: Input Shape of 18 and Average Reference Montage: Input Shape of 19

Due to the small and unbalanced dataset, a single train-test split may preserve bias in the test set and thus may not be indicative of how well the model performed at learning the features in the dataset—the model may have performed well for that specific train-test split but not have generalized well to other data. To achieve more robust estimates of model performance, 45 train-test splits were formed, each of which reserved a different combination of two EEGs (1 GPD and 1 non-GPD) to form the test set with the remaining 12 EEGs (8 GPDs and 4 non-GPDs) for training the model. For each split, a model was trained on the 12 EEGs in the training set and tested on the 2 unseen ("naïve") EEGs in the test set. Given 9 GPDs and 5 non-GPDs in the dataset, this resulted in 45 different models trained and tested on different splits of EEG data (Figure 7) but using the same hyperparameters (learning rate, number of training epochs, batch size, drop-out rate, etc.). Each model's performance was measured using the following metrics: loss, classification accuracy, sensitivity, precision, specificity, F1 score, false negative rate (FNR), and false positive rate (FPR). Because 45 different models were trained, 45 sets of metrics were created. This process was done for both the longitudinal bipolar and the average reference montages.

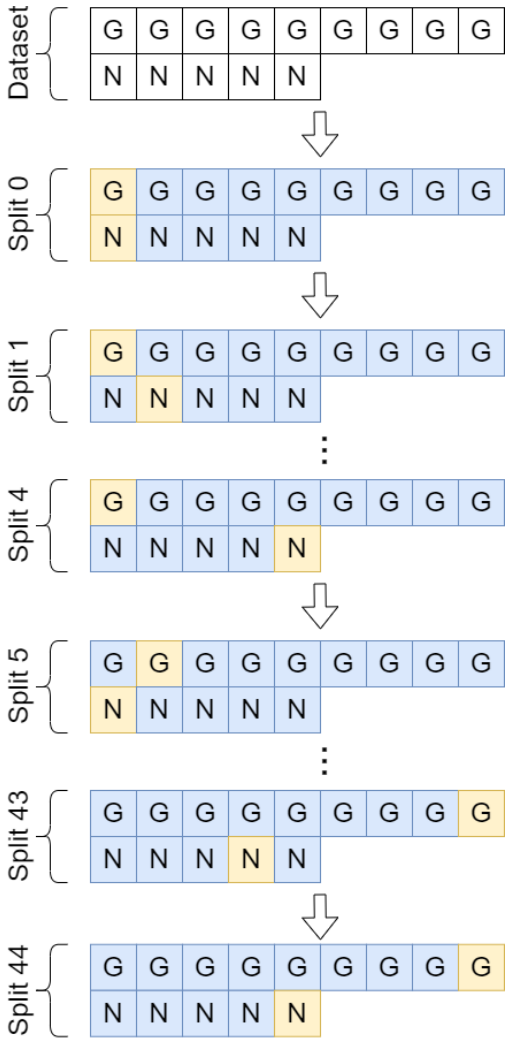


Figure 7: Train and Test Method, Boxes with "G" Represent GPD Cases, Boxes with "N" Represent Non-GPD Cases, Yellow Boxes Indicate Samples Reserved for Test Set of Split, Blue Boxes Indicate Samples Reserved for Training Set of Split

## References

Diana, N. E., Kalsum, U., Sabiq, A., & Jatmiko, W. (2016). Comparing Windowing Methods on Finite Impulse Response (FIR) Filter Algorithm in Electroencephalography (EEG) Data Processing. *Journal of Theoretical and Applied Information Technology*, *88*, 558–567.

Ebersole, J. S., Nordli, D. R., & Husain, A. M. (2014). *Current practice of clinical electroencephalography*. Lippincott Williams & Wilkins.

Guideline 5: Guidelines for standard electrode position nomenclature. (2006). *Journal of Clinical Neurophysiology*, *23*(2), 107–110. https://doi.org/10.1097/00004691-200604000-00006

Kiranyaz, S., Avci, O., Abdeljaber, O., Ince, T., Gabbouj, M., & Inman, D. J. (2021). 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, *151*, 107398. https://doi.org/10.1016/j.ymssp.2020.107398

Libenson, M. H. (2010). *Practical approach to electroencephalography*. Saunders.

## Chapter 3: Results

After creating the 45 models for each montage, the 45 set of metrics achieved were summarized using a box and whisker plot Figure 8. From the top left moving left to right and then downward, the order of the plots is as follows: loss, accuracy, sensitivity, specificity, precision, F1 score, FNR, and FPR. All metrics excluding loss have their y-axes as the respective metric as percent values. The median values of all the metrics for both montages are marked as red lines within the *boxes*. The bottom and top edges of the boxes displayed Q1, the value in which 25% of the metric are located, and Q3, the value in which 75% of the metric are located, respectively (Galarnyk, 2020). The points plotted beyond the *whiskers* on the boxplots represent the outliers from the dataset in their respective montage and metric. The median, Q1, and Q3 from the box and whisker plots for the longitudinal bipolar and average reference montages are recorded in Table 1 and Table 2 respectively. In addition to this information, the third column of Table 1 and Table 2 display the averages of the metrics across the 45 models.

The *Median* and *Mean* columns for the longitudinal bipolar montage in Table 1 show that the medians for accuracy, sensitivity, specificity, precision, and F1 score are greater than their respective mean values. On the other hand, the mean values for loss, FNR, and FPR are greater than their respective median values. The same relationships seen from the longitudinal bipolar montage between median and mean values is also observed in the average reference montage Table 2.

| Longitudinal Bipolar Montage | | | | |
|---|---|---|---|---|
| **Metric** | **Median** | **Mean** | **Q1** | **Q3** |
| **Loss** | 0.4162 | 0.4361 | 0.3775 | 0.4458 |
| **Accuracy (%)** | 97.4629 | 92.9489 | 92.5411 | 99.6224 |
| **Sensitivity (%)** | 100.0000 | 90.4592 | 99.4764 | 100.0000 |
| **Specificity (%)** | 98.1297 | 95.0241 | 92.6650 | 99.4429 |
| **Precision (%)** | 97.3926 | 94.6709 | 92.1610 | 99.4358 |
| **F1 Score (%)** | 97.4954 | 90.3593 | 93.0514 | 99.6535 |
| **FNR (%)** | 0.0000 | 9.5408 | 0.0000 | 0.5236 |
| **FPR (%)** | 1.8703 | 4.9759 | 0.5571 | 7.3350 |

Table 1: Longitudinal Bipolar Montage Median, Mean, Q1, and Q3 of Metrics Seen in Box Plot, Highlighted are the Greater Values Between Median And Mean

| Average Reference Montage | | | | |
|---|---|---|---|---|
| **Metric** | **Median** | **Mean** | **Q1** | **Q3** |
| **Loss** | 0.4035 | 0.4416 | 0.3719 | 0.5016 |
| **Accuracy (%)** | 97.5543 | 90.8032 | 84.5916 | 99.7609 |
| **Sensitivity (%)** | 100.0000 | 90.6294 | 99.4764 | 100.0000 |
| **Specificity (%)** | 99.4429 | 90.7198 | 89.1102 | 100.0000 |
| **Precision (%)** | 99.4294 | 92.2066 | 90.0966 | 100.0000 |
| **F1 Score (%)** | 97.6220 | 89.4826 | 86.5041 | 99.7709 |
| **FNR (%)** | 0.0000 | 9.3706 | 0.0000 | 0.5236 |
| **FPR (%)** | 0.5571 | 9.2802 | 0.0000 | 10.8898 |

Table 2: Average Reference Montage Median, Mean, Q1, and Q3 of Metrics Seen in Box Plot, Highlighted are the Greater Values Between Median And Mean

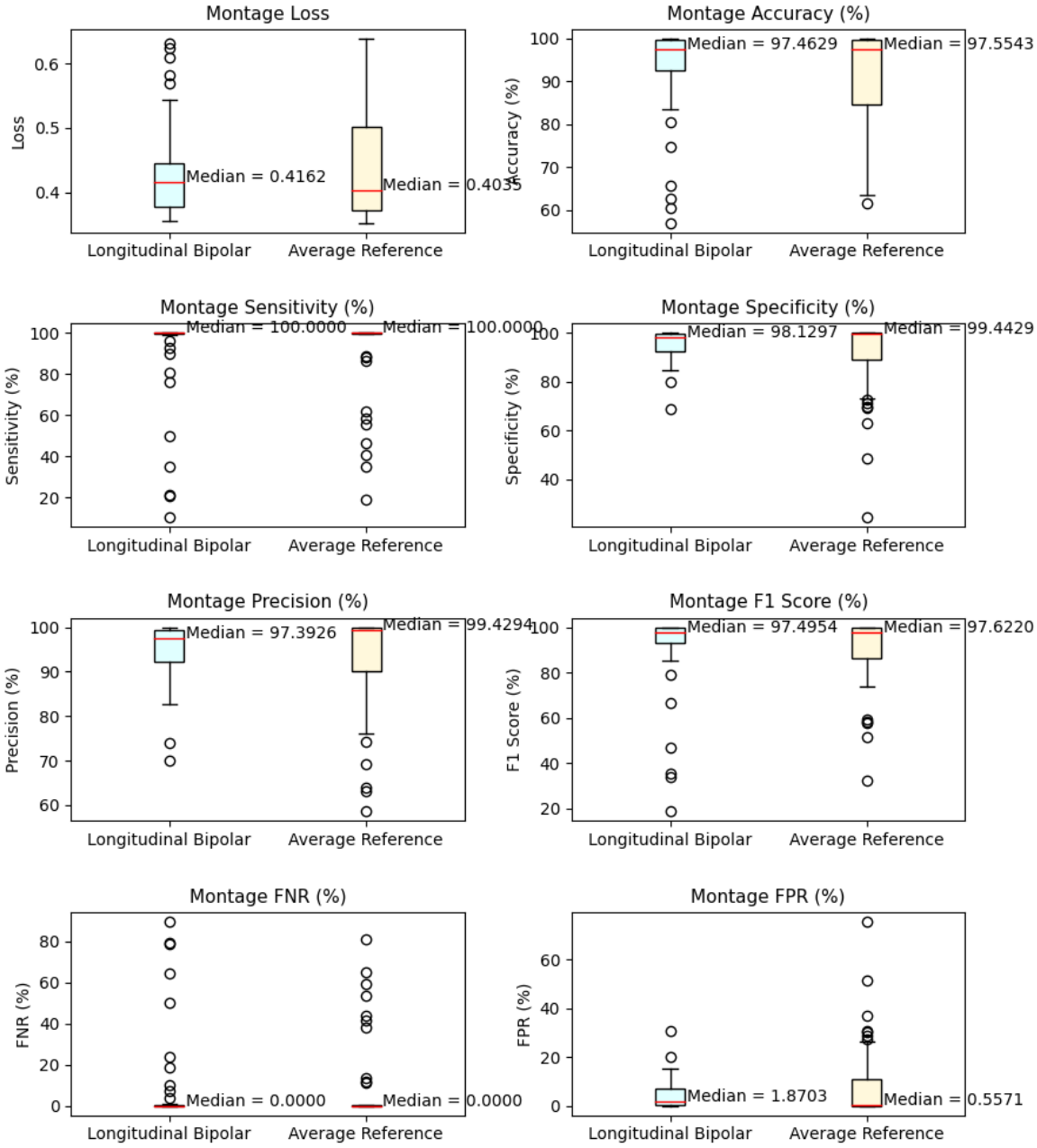


Figure 8: Box and Whisker Plots of Metrics. Longitudinal Bipolar Montage (Left, Blue) and Average Reference Montage (Right, Yellow). From Left to Right and Top Down: Loss, Accuracy, Sensitivity, Specificity, Precision, F1 Score, FNR, FPR

## References

Galarnyk, M. (2020, July 6). *Understanding Boxplots*. Medium. Retrieved April 24, 2022, from https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

## Chapter 4: Discussion and Conclusions

In this study, a 1D CNN architecture was developed and tested for the binary classification of cardiac arrest patient EEGs into two categories: GPDs and diffusely slow non-GPDs. In addition, the model was trained using different montages—the longitudinal bipolar and average reference montages—to assess the predictive power of the feature-space represented by the different montages.

The box plots (Figure 8) of the accuracy, specificity, precision, F1 score, and FPR indicated skews in these metrics for both montages. Particularly, the accuracy, specificity, precision, and F1 score were negatively skewed towards their third quartiles indicated by the median lines being closer to the Q3 edges of the boxes; the negative skewness of these metrics was also observed in Table 1 for the longitudinal bipolar montage and Table 2 for the average reference montage with the medians being greater than the means. On the other hand, the FPRs were positively skewed towards their first quartiles indicated by the median lines being closer to the Q1 edges of the boxes; this was further seen in Table 1 and Table 2 with the means being greater than the medians. In addition to these metrics, while not directly visible in the box plots, Table 1 and Table 2 showed the medians of the sensitivities were greater than the means, indicating a negative skew in this metric as well; on the other hand, the FNRs were seen to be positively skewed indicated by the means being greater than the medians. Since the distribution of the metrics were skewed, the final performance metrics for the montages were best reported as the medians of the metrics obtained from the 45 models, since the median is less affected by outliers and distorted distributions. Moreover, the metrics were skewed to indicate generally good model performance—high accuracy, high sensitivity, high specificity, high precision, high F1 score, low FNR, and low FPR.

From Table 1 and Table 2, the median metrics of the longitudinal bipolar and average reference montages were comparable with the largest difference being approximately 2% seen in the precision metrics. Furthermore, the median FNR scores for both montages were 0%; whereas the median FPR scores were slightly higher at 1.87% and 0.56% for the longitudinal bipolar and average reference montages, respectively. Similarly, the median sensitivities of both metrics were 100% and higher than the median specificities being approximately 99%. The value of these metrics indicated that the model's ability to correctly classify GPDs was slightly better than its ability to classify non-GPDs. These differences could be attributed to the imbalance in the dataset with there being more GPD EEGs than non-GPD EEGs which should have allowed the model to learn more features of GPDs and thus predict this class more often.

While the median values of the longitudinal bipolar and average reference montages were comparable, generally the average reference montage's box plots had larger boxes and larger whiskers than their longitudinal bipolar montage counterparts. This implied that while the final median performance metrics of both montages were approximately the same there was more variability in the performance of the 45 models in the average reference montage. This could be attributed to a possible bias in the dataset. There was no clear indication as to what montage the neurologist read and annotated the EEG in. However, the longitudinal bipolar montage is the more common of the two montages.

Moreover, while variability was more evident in the average reference montage than in the longitudinal bipolar montage, the variability observed in the box plots of both montages suggested the model was sensitive to the data being used for training. This variability was attributed to the small dataset and the training and testing method—excluding one of each EEG type for the test set. The largest variability for both montages was seen in the loss and accuracy

box plots while the smallest variability was seen in FNR and sensitivity. This implied these models' ability to correctly classify these GPD cases was at the cost of predicting more false positives in the non-GPD. This was supported by the precision and F1 median scores being less than 100%.

While the longitudinal bipolar and average reference montages used in this study performed similarly for this classification task, Trambaiolli et al. (2011) who evaluated the effects different montages had on Alzheimer's Disease diagnosis noted the performance of their machine learning models (support vector machine and logistic regression) varied depending on the montage used (Trambaiolli et al., 2011). Their results showed the counterpart bipolar montage performed the best. While it is possible that different montages may only affect Alzheimer's Disease diagnosis or support vector machines and logistic regression models, to confirm their results, more montages can be tested using the 1D CNN model in this study to confirm the findings of Trambaiolli et al.

In this study the 1D CNN was used for its compactness and ability to read bioelectrical signals such as the electrocardiogram; the results of this research showcased the ability of the 1D CNN to be used for this task in this patient group. Jonas et al. (2019) focused on a similar patient group—comatose patients post-cardiac arrest—and used a 1D CNN to predict prognosis using EEGs (Jonas et al., 2019). Their results support the ability of this type of neural network to read and extract relevant features of EEGs from this patient group.

While the 1D CNN was seen to be capable of learning how to read EEGs from this patient group, Zheng et al. (2022) were successful in using longitudinal bipolar montage EEGs and a bidirectional LSTM deep RNN to predict neurological outcomes of comatose post-cardiac

arrest patients (Zheng et al., 2022). Their success of using an RNN in this patient group prompt the need for an RNN architecture to be explored for this classification task.

Tjepkema-Cloostermans et al. (2018) tested different deep neural networks (1D CNNs, 2D CNNs, LSTMs, 1D CNNs with LSTMs and 2D CNNs with LSTMs) to detect epileptiform discharges using 50 focal epilepsy patients' EEGs and 50 normal EEGs (Tjepkema-Cloostermans et al., 2018). Their results suggested the 2D CNN architectures outperformed the 1D CNN architectures. This may suggest implementing a 2D CNN architecture similar to the current 1D CNN architecture may result in better performance. Thus, it seems that many different machine learning architectures can perform the task of reading EEGs; however, things such as computational cost and speed should be taken into consideration as well as performance.

To conclude, the 1D CNN architecture designed was successful in learning the features necessary for the binary classification of GPDs from non-GPDs from the dataset. This was seen in the comparable performances of the longitudinal bipolar and average reference montages in classifying the EEGs from the dataset. However, more data is needed to firstly correct the bias in the dataset towards the GPD class, allow for better model generalization, reduce variability seen in the metrics, and determine if the current model needs to be adjusted. In addition, data read in the average reference montage format should be added to correct this possible bias towards the longitudinal bipolar montage and further reduce the variability seen in the average reference montage's metrics. The results of the work done by other researchers such as Zheng et al. and Tjepkema-Cloostermans et al. prompt the need to explore other neural network architectures such as RNNs, 2D CNNs, and perhaps a combination of the two in the future. In addition to exploring different architectures, different features such as frequency domain features should be researched as well. This study is limited to a binary classification of a subset of patients

presenting with GPDs which are a distinct waveform. More complex and robust architectures are needed to generate models capable of classifying the diversity of EEG waveforms. The architecture and results of this paper should hopefully serve as a steppingstone for other researchers to expand upon, so one day machine learning models can more readily be deployed to assist neurologists at reading and analyzing EEGs.

## References


Jonas, S., Rossetti, A. O., Oddo, M., Jenni, S., Favaro, P., & Zubler, F. (2019). EEG-based outcome prediction after cardiac arrest with convolutional neural networks: Performance and visualization of discriminative features. *Human Brain Mapping*, *40*(16), 4606–4617. https://doi.org/10.1002/hbm.24724

Tjepkema-Cloostermans, M. C., de Carvalho, R. C. V., & van Putten, M. J. A. M. (2018). Deep learning for detection of focal epileptiform discharges from Scalp Eeg Recordings. *Clinical Neurophysiology*, *129*(10), 2191–2196. https://doi.org/10.1016/j.clinph.2018.06.024

Trambaiolli, L. R., Lorena, A. C., Fraga, F. J., Kanda, P. A., Nitrini, R., & Anghinah, R. (2011). Does EEG montage influence alzheimer's disease electroclinic diagnosis? *International Journal of Alzheimer's Disease*, *2011*, 1–6. https://doi.org/10.4061/2011/761891

Zheng, W.-L., Amorim, E., Jing, J., Wu, O., Ghassemi, M., Lee, J. W., Sivaraju, A., Pang, T., Herman, S. T., Gaspard, N., Ruijter, B. J., Tjepkema-Cloostermans, M. C., Hofmeijer, J., van Putten, M. J., & Westover, M. B. (2022). Predicting neurological outcome from electroencephalogram dynamics in comatose patients after cardiac arrest with deep learning. *IEEE Transactions on Biomedical Engineering*, *69*(5), 1813–1825. https://doi.org/10.1109/tbme.2021.3139007

## Appendix A

## Preprocessing

Eeg_montage_helpers.py

```
import os
import numpy as np
import scipy
import matplotlib.pyplot as plt
import mne

#montages
def long_montage(eeg):
   # anodes =   ['Fp1', 'F3', 'C3', 'P3', 'Fp2', 'F4', 'C4', 'P4', 'Fp1', 'F7',
'T3', 'T5', 'Fp2', 'F8', 'T4', 'T6', 'Fz', 'Cz']
   # cathodes = ['F3',  'C3', 'P3', 'O1', 'F4',  'C4', 'P4', 'O2', 'F7',  'T3',
'T5', 'O1', 'F8',  'T4', 'T6', 'O2', 'Cz', 'Pz']
    if 'T7' in eeg.ch_names:
        anodes =   ['Fp1', 'F7', 'T7', 'P7', 'Fp2', 'F8', 'T8', 'P8', 'Fp1',
'F3', 'C3', 'P3', 'Fp2', 'F4', 'C4', 'P4', 'Fz', 'Cz']
        cathodes = ['F7',  'T7', 'P7', 'O1', 'F8',  'T8', 'P8', 'O2',  'F3',
'C3', 'P3', 'O1', 'F4',  'C4', 'P4', 'O2', 'Cz', 'Pz']
    elif 'T3' in eeg.ch_names:
        anodes =   ['Fp1', 'F7', 'T3', 'T5', 'Fp2', 'F8', 'T4', 'T6', 'Fp1',
'F3', 'C3', 'P3', 'Fp2', 'F4', 'C4', 'P4', 'Fz', 'Cz']
        cathodes = ['F7',  'T3', 'T5', 'O1', 'F8',  'T4', 'T6', 'O2',  'F3',
'C3', 'P3', 'O1', 'F4',  'C4', 'P4', 'O2', 'Cz', 'Pz']
    eeg_long = mne.set_bipolar_reference(eeg, anode=anodes, cathode = cathodes)

    # Comment/uncomment for visualizaiton as needed
    ###############################################
    # eeg_long.plot()
    #eeg_bipolar.plot_psd()
    ###############################################

    return eeg_long

def average_montage(eeg):
    eeg_avg = eeg.copy().set_eeg_reference(ref_channels='average')

    # Comment/uncomment for visualizaiton as needed
    ###############################################
    #eeg_avg.plot()
    #eeg_avg.plot_psd()
```

```
    ################################################
    return eeg_avg

def transverse_montage(eeg):
    if 'T7' in eeg.ch_names:
        anodes =   ['F7',  'Fp1', 'Fp2', 'F7', 'F3', 'Fz', 'F4', 'T7', 'C3',
'Cz', 'C4', 'P7', 'P3', 'Pz', 'P4', 'P7', 'O1', 'O2']
        cathodes = ['Fp1', 'Fp2',  'F8', 'F3', 'Fz', 'F4', 'F8', 'C3', 'Cz',
'C4', 'T8', 'P3', 'Pz', 'P4', 'P8', 'O1', 'O2', 'P8']
    elif 'T3' in eeg.ch_names:
        anodes =   ['F7',  'Fp1', 'Fp2', 'F7', 'F3', 'Fz', 'F4', 'T3', 'C3',
'Cz', 'C4', 'T5', 'P3', 'Pz', 'P4', 'T5', 'O1', 'O2']
        cathodes = ['Fp1', 'Fp2',  'F8', 'F3', 'Fz', 'F4', 'F8', 'C3', 'Cz',
'C4', 'T4', 'P3', 'Pz', 'P4', 'T6', 'O1', 'O2', 'T6']
    eeg_transverse = mne.set_bipolar_reference(eeg, anode=anodes, cathode =
cathodes)

    # Comment/uncomment for visualizaiton as needed
    ################################################
    # eeg_transverse.plot()
    #eeg_transverse.plot_psd()
    ################################################
    return eeg_transverse

#####################################
# save montage
def slice_save_montage(montage, n_seconds, pct_overlap, case, save_path):
    s_freq = montage.info['sfreq']
    start = 0
    end = int(s_freq * n_seconds)
    # Should be noted that this shift only works for 50% overlap
    # Correct shift for other overlap should be shift = end - int(s_freq *
n_seconds * pct_overlap/100)
    shift = int(s_freq * n_seconds * pct_overlap/100)
    case_num = os.path.splitext(case)[0]
    while end <= montage._data.shape[1]:
        temp_montage = montage._data[:, start:end+1]

        montage_string = "".join([case_num, "_", str(int(start/s_freq)), 's-',
str(int(end/s_freq)), 's'])

        # add np.saves and move to different folder depending on train or test
        np.save(os.path.join(save_path, montage_string), temp_montage)
```

```
        start += shift
        end += shift

def eeg_preprocess(patient_data, case, n_seconds, pct_overlap, save_path,
montage):
    data_path = os.path.join(patient_data, case)
    eeg = mne.io.read_raw_edf(data_path)
    eeg.load_data()
    channels = eeg.ch_names

    # Depending on when EEG was taken need to include this as the naming
convention was changed
    if 'T7' in channels:
        eeg.pick(['Fp1', 'F7', 'T7', 'P7', 'O1', 'Fp2', 'F8', 'T8', 'P8', 'O2',
'F3', 'C3', 'P3', 'F4', 'C4', 'P4', 'Fz', 'Cz', 'Pz'])
    elif 'T3' in channels:
        eeg.pick(['Fp1', 'F7', 'T3', 'T5', 'O1', 'Fp2', 'F8', 'T4', 'T6', 'O2',
'F3', 'C3', 'P3', 'F4', 'C4', 'P4', 'Fz', 'Cz', 'Pz'])

    # Analyzed data beforehand and noted the most occuring sampling frequency was
256 will resample data otherwise
    if eeg.info['sfreq'] != 256.0:
        eeg.resample(256.0)

    # For unfiltered saving
    if montage == 'unfiltered' and save_path!= None:
        slice_save_montage(eeg, n_seconds, pct_overlap, case, save_path)
        return

    eeg.notch_filter(freqs = 60, filter_length = 'auto', phase = 'zero')
    eeg.filter(None, 70., fir_design='firwin', h_trans_bandwidth = 5, fir_window
= "blackman")
    print('EEGSfreq=',eeg.info['sfreq'])

    # Comment/uncomment for visualizaiton filtered time signal as needed
    ################################################
    # eeg.plot()
    ################################################

    # naming convention (eeg_montage)
    if montage == 'long':
        eeg_montage = long_montage(eeg)
    elif montage == 'avg':
        eeg_montage = average_montage(eeg)
```

```
    elif montage == 'grd':
        eeg_montage = eeg
    elif montage == 'trans':
        eeg_montage = transverse_montage(eeg)

    print('MontageSfreq=',eeg_montage.info['sfreq'])

    if save_path != None and montage != 'unfiltered':
        slice_save_montage(eeg_montage, n_seconds, pct_overlap, case, save_path)
```

split.py

```
import random
import os
import numpy as np
import eeg_montage_helpers as emh
import argparse


def split(montage):
    n_seconds = 10
    pct_overlap = 50

    folder = os.getcwd()
    ca_folder = os.path.join(folder, "Cardiac Arrest - Copy")
    eegs = os.listdir(ca_folder)

    montage_folder = os.path.join(folder, montage)
    if os.path.exists(montage_folder) == False:
        os.mkdir(montage_folder)

    for eeg_type in eegs:
        patient_data = os.path.join(ca_folder, eeg_type)
        cases = os.listdir(patient_data)

        eeg_montage_folder = os.path.join(montage_folder, eeg_type)

        if os.path.exists(eeg_montage_folder) == False:
            os.mkdir(eeg_montage_folder)

        for case in cases:
            eeg_case_montage_folder = os.path.join(eeg_montage_folder,
os.path.splitext(case)[0])
```

```
            if os.path.exists(eeg_case_montage_folder) == False:
                os.mkdir(eeg_case_montage_folder)

            emh.eeg_preprocess(patient_data, case, n_seconds, pct_overlap,
eeg_case_montage_folder, montage)


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('-m', '--montage', default = None, help = 'Montage want
to create: long, avg (average), grd, or trans', type = str)
    args = parser.parse_args()
    split(args.montage)
```

**Machine Learning**

GpdDataset.py

```
import torch
from torch.utils.data import Dataset
import numpy as np
import os

class GpdDataset(Dataset):
    def __init__(self, root_dir, folders):
        self.root_dir = root_dir
        self.data = []
        eeg_type = os.path.split(root_dir)[1]
        for folder in folders:
            eeg_folder = os.path.join(root_dir, folder)
            for np_eeg in os.listdir(eeg_folder):
                np_eeg_in_type = os.path.join(eeg_folder, np_eeg)
                self.data.append([np_eeg_in_type, eeg_type])
        self.labels = {'GPDs' : 1, 'Non GPDs': 0}


    def __getitem__(self, idx):
        eeg = np.load(self.data[idx][0])
        id = self.labels[self.data[idx][1]]
        return eeg, id

    def __len__(self):
        return len(self.data)
```

CNN_1D.py

```
from torch import dropout, nn, seed
import torch.nn.functional as F

class CNN_1D(nn.Module):
    def __init__(self, num_rows):
        super(CNN_1D, self).__init__()
        self.kernel_size = 2

        self.conv1 = nn.Conv1d(num_rows, 32, kernel_size = self.kernel_size**5,
bias = False)
        self.bn1 = nn.BatchNorm1d(32)
        self.mp1 = nn.MaxPool1d(10)

        self.conv2 = nn.Conv1d(32, 32, kernel_size = self.kernel_size**4, bias =
False)
        self.bn2 = nn.BatchNorm1d(32)
        self.mp4 = nn.MaxPool1d(100)

        self.dense = nn.Linear(64, 2)

    def forward(self, x):

        x = self.conv1(x.float())
        x = F.leaky_relu(x)
        x = self.bn1(x)
        x = self.mp1(x)

        x = self.conv2(x)
        x = F.leaky_relu(x)
        x = self.bn2(x)
        x = self.mp4(x)

        x = x.reshape(x.size(0), -1)
        x = F.dropout(x, p = 0.3)

        x = self.dense(x)
        x = F.softmax(x, dim=1)

        return x
```

Run_train_test.py

```
from logging import root
from sklearn.model_selection import LeaveOneOut
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, f1_score
import torch
from torch import le, nn
from torch import optim
import torch.nn.functional as F
from CNN_1D import CNN_1D
from torch.utils.data import Dataset, DataLoader, random_split, ConcatDataset
from GpdDataset import GpdDataset
import os
import numpy as np
import random
import time
import matplotlib.pyplot as plt
import xlrd
import xlwt
from xlwt import Workbook

import argparse
from torch.utils.tensorboard import SummaryWriter


def train(root_dir, montage, device, save_folder, learning_rate, num_epochs, tb):
    start = time.time()

    montage_save_folder = os.path.join(save_folder,
time.strftime("%m.%d.%Y_%H%M%S", time.localtime()))
    os.mkdir(montage_save_folder)

    weights_folder = os.path.join(montage_save_folder, 'model_weights')
    os.mkdir(weights_folder)

    metrics_folder = os.path.join(montage_save_folder, 'model_metrics')
    os.mkdir(metrics_folder)

    # Excel sheet
    wb = Workbook()
    sheet1 = wb.add_sheet('Sheet 1')
    sheet1.write(0, 0, 'Test Set')
    sheet1.write(0, 1, 'Loss')
    sheet1.write(0, 2, 'Accuracy (%)')
    sheet1.write(0, 3, 'Sensitivity (%)')
```

```
    sheet1.write(0, 4, 'Specificity (%)')
    sheet1.write(0, 5, 'Precision (%)')
    sheet1.write(0, 6, 'F1 Score (%)')
    sheet1.write(0, 7, 'FNR (%)')
    sheet1.write(0, 8, 'FPR (%)')

    if tb:
        tb_folder = os.path.join(montage_save_folder, 'runs')
        os.mkdir(tb_folder)

    loss_fun = nn.CrossEntropyLoss()

    gpd_folder = os.path.join(root_dir, 'GPDs')
    gpds = os.listdir(gpd_folder)

    non_gpd_folder = os.path.join(root_dir, 'Non GPDs')
    non_gpds = os.listdir(non_gpd_folder)

    folds = len(gpds) * len(non_gpds) - 1
    for i, gpd in enumerate(gpds):
        test_gpd = GpdDataset(root_dir = gpd_folder, folders = [gpd])

        train_gpd_folders = list(set(gpds) - set([gpd]))
        train_gpd_folders.sort()
        train_gpd = GpdDataset(root_dir=gpd_folder, folders = train_gpd_folders)

        for j, non_gpd in enumerate(non_gpds):
            save_string = montage + '-test-set-' + gpd + '-' + non_gpd
            fold = len(non_gpds) * i + j

            test_non_gpd = GpdDataset(root_dir=non_gpd_folder, folders =
[non_gpd])
            train_non_gpd_folders = list(set(non_gpds) - set([non_gpd]))
            train_non_gpd_folders.sort()
            train_non_gpd = GpdDataset(root_dir=non_gpd_folder, folders =
train_non_gpd_folders)

            test_data = ConcatDataset([test_gpd, test_non_gpd])
            train_data = ConcatDataset([train_gpd, train_non_gpd])

            # For tensorboard
            if tb:
                writer = SummaryWriter(log_dir=os.path.join(tb_folder,
save_string))
```

```
            trainloader = DataLoader(
                train_data,
                batch_size = 300,
                shuffle = True,
            )

            testloader = DataLoader(
                test_data,
                batch_size = 300,
                shuffle = False,
            )

            model = None
            if montage == 'long' or montage == 'trans':
                model = CNN_1D(18).to(device=device)
            elif montage == 'avg' or montage == 'grd':
                model = CNN_1D(19).to(device=device)

            optimizer = optim.SGD(model.parameters(), lr = learning_rate,
momentum = 0.1, weight_decay=1e-5)

            print('Training set: GPDs = {}; Non GPDs =
{}'.format(train_gpd_folders, train_non_gpd_folders))
            print('Test set: GPDs = {}; Non GPDs = {}'.format(gpd, non_gpd))
            print('Training start: Fold: {}/{}'.format(fold, folds))
            print('----------------------------------------------------------------
--------------------------------------------------------------------------------
--------')
            for epoch in range(num_epochs):

                train_loss = 0.0
                train_accuracy = 0

                model.train()
                for (inputs, labels) in trainloader:
                    inputs = inputs.to(device)
                    labels = labels.to(device)

                    optimizer.zero_grad()

                    outputs = model(inputs)
                    loss = loss_fun(outputs, labels)
                    loss.backward()
```

```
                    optimizer.step()

                    train_loss += loss.item()
                    train_accuracy += sum(labels == outputs.argmax(1)).item()

                train_loss = train_loss/len(trainloader)
                train_accuracy = train_accuracy/len(train_data)*100

                # Tensorboard Logging
                if tb:
                    writer.add_scalar("Train Loss per Epoch", train_loss, epoch)
                    writer.add_scalar("Train accuracy per Epoch", train_accuracy,
epoch)

                print("Epoch: {}, Train Loss: {}, Train Accuracy:
{}".format(epoch, train_loss, train_accuracy))
            if tb:
                writer.flush
            print('Training complete: Fold: {}/{}'.format(fold, folds))

            test_metrics = test(model, device, test_data, testloader)
            torch.save(model.state_dict(), os.path.join(weights_folder,
(save_string + '.pt')))

            sheet1.write(fold+1, 0, (gpd + ', ' + non_gpd))
            for col, metric in enumerate(test_metrics):
                sheet1.write(fold+1, col+1, metric)
            wb.save(os.path.join(metrics_folder, 'metrics.xls'))

            print('------------------------------------------------------------
------------------------------------------------------------------------------
--------')
            print('')
    end = time.time()



def test(model, device, test_data, testloader):
    print('Testing:')
    print('-----------------------------------------------------')


    loss_fun = nn.CrossEntropyLoss()
```

```
    test_accuracy = 0.0
    test_loss = 0.0
    out_arr = np.empty(0)
    label_arr = np.empty(0)

    tp = 0
    tn = 0
    fp = 0
    fn = 0
    model.eval()
    with torch.no_grad():
        for inputs, labels in testloader:
            inputs = inputs.to(device)
            labels = labels.to(device)
            outputs = model(inputs)
            loss = loss_fun(outputs, labels)
            test_loss += loss.item()
            test_accuracy += sum(labels == outputs.argmax(1)).item()

            out_arr = np.append(out_arr,
outputs.argmax(1).cpu().detach().numpy())
            label_arr = np.append(label_arr, labels.cpu().detach().numpy())
    test_loss = test_loss/len(testloader)
    test_accuracy = test_accuracy/len(test_data) * 100
    print('Test Loss: {} Test Accuracy: {}'.format(test_loss, test_accuracy))
    cm = confusion_matrix(list(label_arr), list(out_arr))
    tn, fp, fn, tp = cm.ravel()

    sensitivity = tp/(tp + fn) * 100
    specificity = tn/(tn + fp) * 100
    precision = tp/(tp + fp) * 100
    f1 =  tp/(tp + 1/2 * (fp + fn)) * 100
    fnr = fn/(fn + tp) * 100
    fpr = fp/(fp + tn) * 100


    # Add something to create an excel file with columns: montage, accuracy,
loss, sens, spec, precision, and so forth
    print('Sensitivity: {}, Specificity: {}, Precision: {}, F1: {}, fnr: {}, fpr:
{}'.format(sensitivity, specificity, precision, f1, fnr, fpr))
    print('-------------------------------------------------------')
    print('Testing complete')
```

```
    return test_loss, test_accuracy, sensitivity, specificity, precision, f1,
fnr, fpr

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('-m', '--montage', default = None, help = 'Name of
montage want to create: long, avg, grd, or trans', type = str)
    parser.add_argument('-o', '--option', default = None, help = 'train or test',
type = str)
    parser.add_argument('-p', '--model_params', default = None, help = 'Name of
dated file for montage (ex. 04.25.2022_190157). Will run all 45 models')
    parser.add_argument('-e', '--epochs', default = 300, help = 'Number of epochs
for training', type = int)
    parser.add_argument('-lr', '--learning_rate', default = 0.0001, help =
'Learning rate used for training', type = float)
    parser.add_argument('-tb', '--tensorboard', default = 'False', choices =
('True', 'False'), help = 'Use tensorboard or not during the training process,
True or False as string', type = str)
    args = parser.parse_args()

    montage_folder = os.path.join(os.getcwd(), args.montage)
    model_param_folder = os.path.join(os.getcwd(), 'model_data')

    if os.path.isdir(model_param_folder) == False:
        os.makedirs(model_param_folder)

    save_folder = os.path.join(model_param_folder, args.montage)
    if os.path.isdir(save_folder) == False:
        os.makedirs(save_folder)

    # For reproducibility original seed was 7823490
    seed = 7823490
    torch.manual_seed(seed)
    random.seed(seed)

    device = ('cuda' if torch.cuda.is_available()
    else 'cpu')
    print(device)

    if args.tensorboard == 'True':
        tb = True
        print('Tensorboard tracking on')
    elif args.tensorboard == 'False':
        tb = False
```

```
        print('Tensorboard tracking off')

    print('Montage: {}'.format(args.montage))
    if args.option == 'train':
        train(montage_folder, args.montage, device, save_folder,
args.learning_rate, args.epochs, tb)
    elif args.option == 'test':
        if args.model_params != None:
            gpd_folder = os.path.join(montage_folder, 'GPDs')
            non_gpd_folder = os.path.join(montage_folder, 'Non GPDs')
            model_weight_folder = os.path.join(model_param_folder,
os.path.join(args.montage, os.path.join(args.model_params, 'model_weights')))
            for i, model_weight in enumerate(os.listdir(model_weight_folder)):
                temp = model_weight.split('-')
                gpd = temp[3]
                non_gpd = temp[4].split('.')[0]

                test_gpd = GpdDataset(root_dir=gpd_folder, folders = [gpd])
                test_non_gpd = GpdDataset(root_dir=non_gpd_folder, folders =
[non_gpd])
                test_data = ConcatDataset([test_gpd, test_non_gpd])

                testloader = DataLoader(
                    test_data,
                    batch_size = 300,
                    shuffle = False,
                )

                model = None
                if args.montage == 'long' or args.montage == 'trans':
                    model = CNN_1D(18).to(device=device)
                elif args.montage == 'avg' or args.montage == 'grd':
                    model = CNN_1D(19).to(device=device)

                model.load_state_dict(torch.load(os.path.join(model_weight_folder
, model_weight)))
                print('Test set {}: GPDs = {}; Non GPDs = {}; Weights:
{}'.format(i, gpd, non_gpd, model_weight))
                test(model, device, test_data, testloader)
                print('')
        else:
            print('Selected test but did not choose weights folder')
```

**Other**

Box_plot.py

```
import xlrd
import xlwt
from xlwt import Workbook
import matplotlib.pyplot as plt
import numpy as np
import argparse

def main(loc):
    wb_long = xlrd.open_workbook(loc[0])
    wb_avg = xlrd.open_workbook(loc[1])
    long = wb_long.sheet_by_index(0)
    avg = wb_avg.sheet_by_index(0)

    box_plot_values = Workbook()
    long_sheet = box_plot_values.add_sheet('Longitudinal')
    avg_sheet = box_plot_values.add_sheet('Average')
    values = ['Median', 'Mean', 'Q1', 'Q3']

    for sheet in [long_sheet, avg_sheet]:
        sheet.write(0, 0, 'Metric')
        sheet.write(1, 0, 'Loss')
        sheet.write(2, 0, 'Accuracy (%)')
        sheet.write(3, 0, 'Sensitivity (%)')
        sheet.write(4, 0, 'Specificity (%)')
        sheet.write(5, 0, 'Precision (%)')
        sheet.write(6, 0, 'F1 Score (%)')
        sheet.write(7, 0, 'FNR (%)')
        sheet.write(8, 0, 'FPR (%)')
        for i, value in enumerate(values):
            sheet.write(0, i+1, value)

    fig1, axs1 = plt.subplots(4, 2, figsize = (9, 10.5))
    fig1.tight_layout(pad = 3.5)
    row = 0

    for i in range(long.ncols):
        long_array = None
        avg_array = None
        colors = ['lightcyan', 'cornsilk']
        median_color = 'red'
```

```
        if i != 0:
            metric = long.cell_value(0, i)
            print('Metric: {}'.format(metric))
            long_metric = []
            avg_metric = []
            save_string = metric + '.png'
            for j in range(long.nrows):
                if j!= 0:
                    long_metric.append(long.cell_value(j, i))
                    avg_metric.append(avg.cell_value(j, i))
            long_array = np.array(long_metric)
            avg_array = np.array(avg_metric)
            data = [long_array, avg_array]

            if (i-1)%2 == 0 and i != 1:
                row += 1

            col = 0
            if i % 2 == 0:
                col = 1

            box = axs1[row, col].boxplot(data, patch_artist=True, labels =
['Longitudinal Bipolar', 'Average Reference'])
            for patch, color in zip(box['boxes'], colors):
                patch.set_facecolor(color)
            for median in box['medians']:
                median.set_color(median_color)
                x, y = median.get_xydata()[1]
                axs1[row, col].text(x+0.01, y, 'Median = %.4f' % y,
horizontalalignment='left')
                # save to median column
                if int(x) == 1:
                    long_sheet.write(i, 1, y)
                elif int(x) == 2:
                    avg_sheet.write(i, 1, y)
            axs1[row, col].set_title('Montage ' + metric, fontsize = 11)
            axs1[row, col].set_ylabel(metric, fontsize = 10)


            for count, whisker in enumerate(box['whiskers']):
                x, y = whisker.get_xydata()[0]
                # Order of appearance is [Q1_long, min_long], [Q3_long,
max_long], [Q1_avg, min_avg], [Q3_long, max_avg]
                if int(x) == 1:
```

```
                    if count % 2 == 0:
                        long_sheet.write(i, 3, y)
                    elif count % 2 == 1:
                        long_sheet.write(i, 4, y)
                elif int(x) == 2:
                    if count % 2 == 0:
                        avg_sheet.write(i, 3, y)
                    elif count % 2 == 1:
                        avg_sheet.write(i, 4, y)


            long_sheet.write(i, 2, np.average(long_array))
            avg_sheet.write(i, 2, np.average(avg_array))

            fig1.show()
    box_plot_values.save('box_values_112.xls')
    fig1.savefig('Box')

if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('-pl', '--path_long', default = None, help = 'Path to
long montage (metrics excel) to create histogram', type = str)
    parser.add_argument('-pa', '--path_avg', default = None, help = 'Path to avg
montage (metrics excel) to create histogram', type = str)
    args = parser.parse_args()
    loc = [args.path_long, args.path_avg]
    main(loc)
```

## Tables

Table of Metrics for 45 Splits Longitudinal Bipolar Montage:

| Split | Loss | Accuracy (%) | Sensitivity (%) | Specificity (%) | Precision (%) | F1 Score (%) | FNR (%) | FPR (%) |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.4098 | 99.9414 | 100.0000 | 99.8921 | 99.8720 | 99.9359 | 0.0000 | 0.1079 |
| 1 | 0.3905 | 99.5995 | 100.0000 | 99.1643 | 99.2366 | 99.6169 | 0.0000 | 0.8357 |
| 2 | 0.4099 | 96.5234 | 100.0000 | 93.1421 | 93.4132 | 96.5944 | 0.0000 | 6.8579 |
| 3 | 0.4445 | 98.6301 | 100.0000 | 97.2112 | 97.3783 | 98.6717 | 0.0000 | 2.7888 |
| 4 | 0.4347 | 96.2453 | 100.0000 | 92.6650 | 92.8571 | 96.2963 | 0.0000 | 7.3350 |
| 5 | 0.3644 | 99.9444 | 100.0000 | 99.8921 | 99.8853 | 99.9426 | 0.0000 | 0.1079 |
| 6 | 0.3561 | 99.6224 | 99.8852 | 99.3036 | 99.4286 | 99.6564 | 0.1148 | 0.6964 |
| 7 | 0.4106 | 95.5170 | 99.8852 | 90.7731 | 92.1610 | 95.8678 | 0.1148 | 9.2269 |
| 8 | 0.4665 | 90.7020 | 100.0000 | 79.9469 | 85.2250 | 92.0232 | 0.0000 | 20.0531 |
| 9 | 0.4290 | 95.5595 | 100.0000 | 90.8313 | 92.0719 | 95.8723 | 0.0000 | 9.1687 |
| 10 | 0.3594 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 0.0000 | 0.0000 |
| 11 | 0.3677 | 99.6521 | 100.0000 | 99.3036 | 99.3094 | 99.6535 | 0.0000 | 0.6964 |
| 12 | 0.4178 | 94.2801 | 100.0000 | 89.1521 | 89.2060 | 94.2951 | 0.0000 | 10.8479 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 13 | 0.3943 | 98.8451 | 100.0000 | 97.7424 | 97.6902 | 98.8316 | 0.0000 | 2.2576 |
| 14 | 0.5135 | 83.5394 | 100.0000 | 69.0709 | 73.9712 | 85.0384 | 0.0000 | 30.9291 |
| 15 | 0.3775 | 99.9402 | 99.8660 | 100.0000 | 100.0000 | 99.9329 | 0.1340 | 0.0000 |
| 16 | 0.3976 | 99.7268 | 100.0000 | 99.4429 | 99.4667 | 99.7326 | 0.0000 | 0.5571 |
| 17 | 0.3930 | 99.0310 | 100.0000 | 98.1297 | 98.0289 | 99.0046 | 0.0000 | 1.8703 |
| 18 | 0.4458 | 94.9300 | 100.0000 | 89.9070 | 90.7543 | 95.1531 | 0.0000 | 10.0930 |
| 19 | 0.4268 | 98.1458 | 100.0000 | 96.4548 | 96.2581 | 98.0934 | 0.0000 | 3.5452 |
| 20 | 0.4272 | 98.0713 | 95.9184 | 99.8921 | 99.8672 | 97.8530 | 4.0816 | 0.1079 |
| 21 | 0.4390 | 94.4740 | 89.9235 | 99.4429 | 99.4358 | 94.4407 | 10.0765 | 0.5571 |
| 22 | 0.5008 | 89.5334 | 80.9949 | 97.8803 | 97.3926 | 88.4401 | 19.0051 | 2.1197 |
| 23 | 0.5044 | 91.4118 | 92.4745 | 90.3054 | 90.8521 | 91.6561 | 7.5255 | 9.6946 |
| 24 | 0.5817 | 80.4619 | 76.0204 | 84.7188 | 82.6630 | 79.2027 | 23.9796 | 15.2812 |
| 25 | 0.3703 | 99.8841 | 99.8747 | 99.8921 | 99.8747 | 99.8747 | 0.1253 | 0.1079 |
| 26 | 0.3624 | 99.6702 | 100.0000 | 99.3036 | 99.3773 | 99.6877 | 0.0000 | 0.6964 |
| 27 | 0.3726 | 99.3125 | 100.0000 | 98.6284 | 98.6403 | 99.3155 | 0.0000 | 1.3716 |
| 28 | 0.4230 | 97.1631 | 100.0000 | 94.1567 | 94.7743 | 97.3171 | 0.0000 | 5.8433 |
| 29 | 0.4155 | 97.4629 | 100.0000 | 94.9878 | 95.1132 | 97.4954 | 0.0000 | 5.0122 |
| 30 | 0.3793 | 99.7043 | 99.4764 | 99.8921 | 99.8686 | 99.6721 | 0.5236 | 0.1079 |
| 31 | 0.3708 | 99.1903 | 99.0838 | 99.3036 | 99.3438 | 99.2136 | 0.9162 | 0.6964 |
| 32 | 0.3742 | 99.4891 | 99.6073 | 99.3766 | 99.3473 | 99.4771 | 0.3927 | 0.6234 |
| 33 | 0.4327 | 96.8359 | 99.8691 | 93.7583 | 94.1975 | 96.9504 | 0.1309 | 6.2417 |
| 34 | 0.4406 | 92.5411 | 99.8691 | 85.6968 | 86.7045 | 92.8224 | 0.1309 | 14.3032 |
| 35 | 0.3587 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 0.0000 | 0.0000 |
| 36 | 0.3655 | 99.7312 | 100.0000 | 99.4429 | 99.4832 | 99.7409 | 0.0000 | 0.5571 |
| 37 | 0.4162 | 96.5649 | 100.0000 | 93.2668 | 93.4466 | 96.6123 | 0.0000 | 6.7332 |
| 38 | 0.4925 | 92.4491 | 100.0000 | 84.7278 | 87.0056 | 93.0514 | 0.0000 | 15.2722 |
| 39 | 0.4081 | 97.8589 | 100.0000 | 95.8435 | 95.7711 | 97.8399 | 0.0000 | 4.1565 |
| 40 | 0.5684 | 65.6952 | 21.5278 | 100.0000 | 100.0000 | 35.4286 | 78.4722 | 0.0000 |
| 41 | 0.5436 | 74.6871 | 50.0000 | 99.4429 | 98.9011 | 66.4207 | 50.0000 | 0.5571 |
| 42 | 0.6100 | 56.9645 | 10.4167 | 98.7531 | 88.2353 | 18.6335 | 89.5833 | 1.2469 |
| 43 | 0.6316 | 60.5567 | 20.6944 | 98.6720 | 93.7107 | 33.9022 | 79.3056 | 1.3280 |
| 44 | 0.6243 | 62.6138 | 35.2778 | 86.6748 | 69.9725 | 46.9067 | 64.7222 | 13.3252 |

Table of Metrics for 45 Splits Average Reference Montage:

| Split | Loss | Accuracy (%) | Sensitivity (%) | Specificity (%) | Precision (%) | F1 Score (%) | FNR (%) | FPR (%) |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.3547 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 0.0000 | 0.0000 |
| 1 | 0.3530 | 99.7997 | 100.0000 | 99.5822 | 99.6169 | 99.8081 | 0.0000 | 0.4178 |
| 2 | 0.3881 | 99.8104 | 100.0000 | 99.6259 | 99.6169 | 99.8081 | 0.0000 | 0.3741 |
| 3 | 0.4319 | 97.5212 | 100.0000 | 94.9535 | 95.3545 | 97.6220 | 0.0000 | 5.0465 |
| 4 | 0.4566 | 90.1752 | 100.0000 | 80.8068 | 83.2444 | 90.8561 | 0.0000 | 19.1932 |
| 5 | 0.3596 | 99.8888 | 99.7704 | 100.0000 | 100.0000 | 99.8851 | 0.2296 | 0.0000 |
| 6 | 0.3602 | 99.7483 | 100.0000 | 99.4429 | 99.5429 | 99.7709 | 0.0000 | 0.5571 |
| 7 | 0.3719 | 99.7609 | 100.0000 | 99.5012 | 99.5429 | 99.7709 | 0.0000 | 0.4988 |
| 8 | 0.4539 | 91.3177 | 100.0000 | 81.2749 | 86.0672 | 92.5119 | 0.0000 | 18.7251 |
| 9 | 0.5775 | 63.4103 | 100.0000 | 24.4499 | 58.4956 | 73.8136 | 0.0000 | 75.5501 |
| 10 | 0.3727 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 0.0000 | 0.0000 |
| 11 | 0.3630 | 99.7216 | 100.0000 | 99.4429 | 99.4467 | 99.7226 | 0.0000 | 0.5571 |
| 12 | 0.3604 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 0.0000 | 0.0000 |
| 13 | 0.3994 | 97.5543 | 100.0000 | 95.2191 | 95.2318 | 97.5577 | 0.0000 | 4.7809 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 14 | 0.5161 | 83.6695 | 100.0000 | 69.3154 | 74.1237 | 85.1391 | 0.0000 | 30.6846 |
| 15 | 0.3881 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 0.0000 | 0.0000 |
| 16 | 0.3689 | 99.6585 | 99.8660 | 99.4429 | 99.4660 | 99.6656 | 0.1340 | 0.5571 |
| 17 | 0.3952 | 99.6770 | 99.8660 | 99.5012 | 99.4660 | 99.6656 | 0.1340 | 0.4988 |
| 18 | 0.4279 | 94.5297 | 100.0000 | 89.1102 | 90.0966 | 94.7903 | 0.0000 | 10.8898 |
| 19 | 0.5614 | 73.1458 | 100.0000 | 48.6553 | 63.9794 | 78.0335 | 0.0000 | 51.3447 |
| 20 | 0.5058 | 82.5833 | 61.9898 | 100.0000 | 100.0000 | 76.5354 | 38.0102 | 0.0000 |
| 21 | 0.4560 | 93.9414 | 88.9031 | 99.4429 | 99.4294 | 93.8721 | 11.0969 | 0.5571 |
| 22 | 0.5618 | 67.3392 | 34.8214 | 99.1272 | 97.5000 | 51.3158 | 65.1786 | 0.8728 |
| 23 | 0.5047 | 89.3949 | 88.1378 | 90.7039 | 90.8016 | 89.4498 | 11.8622 | 9.2961 |
| 24 | 0.6012 | 74.4070 | 86.3520 | 62.9584 | 69.0816 | 76.7574 | 13.6480 | 37.0416 |
| 25 | 0.3602 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 0.0000 | 0.0000 |
| 26 | 0.3788 | 99.7361 | 100.0000 | 99.4429 | 99.5012 | 99.7500 | 0.0000 | 0.5571 |
| 27 | 0.3623 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 0.0000 | 0.0000 |
| 28 | 0.4100 | 96.0026 | 100.0000 | 91.7663 | 92.7907 | 96.2606 | 0.0000 | 8.2337 |
| 29 | 0.5016 | 84.5916 | 100.0000 | 69.5599 | 76.2178 | 86.5041 | 0.0000 | 30.4401 |
| 30 | 0.3885 | 99.7635 | 99.4764 | 100.0000 | 100.0000 | 99.7375 | 0.5236 | 0.0000 |
| 31 | 0.3742 | 99.5277 | 99.6073 | 99.4429 | 99.4771 | 99.5422 | 0.3927 | 0.5571 |
| 32 | 0.3763 | 99.6807 | 99.3455 | 100.0000 | 100.0000 | 99.6717 | 0.6545 | 0.0000 |
| 33 | 0.4420 | 95.0560 | 99.7382 | 90.3054 | 91.2575 | 95.3096 | 0.2618 | 9.6946 |
| 34 | 0.4929 | 85.7775 | 99.7382 | 72.7384 | 77.3604 | 87.1355 | 0.2618 | 27.2616 |
| 35 | 0.3693 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 100.0000 | 0.0000 | 0.0000 |
| 36 | 0.3747 | 99.6640 | 99.8701 | 99.4429 | 99.4825 | 99.6760 | 0.1299 | 0.5571 |
| 37 | 0.3667 | 99.5547 | 100.0000 | 99.1272 | 99.0991 | 99.5475 | 0.0000 | 0.8728 |
| 38 | 0.4035 | 98.1615 | 100.0000 | 96.2815 | 96.4912 | 98.2143 | 0.0000 | 3.7185 |
| 39 | 0.4931 | 86.2091 | 99.8701 | 73.3496 | 77.9129 | 87.5356 | 0.1299 | 26.6504 |
| 40 | 0.4971 | 81.9065 | 58.6111 | 100.0000 | 100.0000 | 73.9054 | 41.3889 | 0.0000 |
| 41 | 0.5568 | 70.2364 | 40.9722 | 99.5822 | 98.9933 | 57.9568 | 59.0278 | 0.4178 |
| 42 | 0.5636 | 61.6951 | 19.1667 | 99.8753 | 99.2806 | 32.1304 | 80.8333 | 0.1247 |
| 43 | 0.6323 | 67.4813 | 46.3889 | 87.6494 | 78.2201 | 58.2389 | 53.6111 | 12.3506 |
| 44 | 0.6383 | 64.0442 | 55.8333 | 71.2714 | 63.1083 | 59.2483 | 44.1667 | 28.7286 |