Designing an Authentication System for Augmented Reality Devices

Submitted to:
Professor Krishna Kumar VENKATASUBRAMANIAN

Submitted by:
Rushdi ABUALHAIJA
Meghana BHATIA
Nikolaos KALAMPALIKIS
Alejandro SOLER GAYOSO

MQP Proposal, 2018-2019

Contents

List of Figures

List of Tables

Abstract

The following report details the potential of using BVP and BCG waveforms as a biometrics means for user authentication using a head-mounted device. As part of the research, we trained a convolutional neural network to retrieve an optimal model that will allow for correct user authentication. The genetic algorithm was utilized in order to find the optimal parameters to be placed into the neural network. We analyzed the results using ROC curves to determine the best and worst performing models from the neural network. Through the ROC curve analysis we determined that there were errors with our BVP results and therefore we moved on with only the BCG waveform. This research aimed to find alternative methods of authentication for users while considering the security of their passwords. The waveforms help us achieve this goal of authentication as users provide their subtle head movements which are much harder for adversaries to mimic than a typed password.

1 Introduction

The secure implementation of biometric authentication methods is becoming a more prevalent challenge in technology today. New ways of authenticating users are developing and are being made more available to the public. This development can be seen with the advent of authentication systems such as fingerprinting, retinal and face id scans. Primarily, the new technology is being used to authenticate users on smartphones, however there are broader applications for this type of technology. In this paper, we propose the idea of utilizing subtle head movements as a new kind of biometric to authenticate a user through the use of machine learning.

Our main goal for the project was to focus on developing a new authentication method using a head-mounted device. To achieve this, we derived two different cardiac signals from a users' involuntary head movements. The three parts to our work was data collection, data preprocessing, and finally authentication as can be seen in **Figure 2** in the Approach section.The signals we derived were Blood Volume Pulse **(BVP)** and Ballistocardiogram **(BCG)**. Both signals were measured with two main sensors: an accelerometer, and a gyroscope. The accelerometer measures acceleration and the gyroscope is used to measure angular velocity and orientation. This initial step was part of the data collection phase, and then we had raw accelerometer and gyroscope signals to work with. As part of data preprocessing, the raw signals were then filtered using bandpass, rolling average and normalization filters. Filtering the raw signals is vital in order to reduce the noise to derive meaningful waveforms. Next, we overlapped the data in order to create more data points that we could use to train a convolutional neural network **(CNN)** in conjunction with the genetic algorithm to retrieve an optimal model to help authenticate users with the greatest accuracy. We trained the CNN, as part of our authentication phase, to perform binary classification to represent a yes or no success of user authentication. As we were training and testing the CNN, we analyzed the performance of users using Receiver Operating Characteristic **(ROC)** curves to determine the best and worst performers in each of the models. This helped us distinguish the outliers and the success rate with which our system was performing authentication. These results also helped us access the success rate for each different user in the different sessions conducted. ROC curves help test the validity of the models, and can be found in the *Results* section.

The following sections describe the whole process our team took in order to accomplish this project. We first give a description of concepts that are vital to the project in the *Background* section, and in the next section, *Problem Statement*, we explain the goals and motivations driving this project. Following that, we lay out our *Approach* section. First, we provide details on how we obtained the data in 4.1, and then the preprocessing steps in order to obtain the different waveforms in 4.2. Then, section 4.3 explains the CNN and model training steps. Section 4.4 details the steps for authentication. Then we describe all the details of the *Parameter Selection* process. First, 5.1 explains the different attributes of the genetic algorithm, and the neural network. Section 5.2 explains the different hyperparameters that were selected from the genetic algorithm. Then section 5.3 explains the details of model training and the process of evaluating the results. Finally, in section 6 we present all our results, in 6.1 we first describe the metrics that were used for evaluating the results. Then sections 6.2 and 6.3 present the results of BVP and BCG respectively. After the last section, we present our discussion, related work, conclusion and future work sections.

2 Background

BVP is the measurement of "dynamic changes in blood volume underneath the surface" of the tissue [**?**]. Gathering BVP is most commonly used in monitoring heart rate through intravenous perfusion throughout the area where the sensor is placed. Sensors for measuring heart rate can be placed anywhere on the body, although they are commonly clipped on the tip of the index finger to measure BVP. Since the characteristics of a subject's perfusion is unique, the BVP signal measurement will provide different results per user. This important component allows for BVP to be a unique identifier that can be utilized to authenticate a user, making it nearly impossible for an adversary to replicate the same cardiac movements [**?**]. The biometric authentication technique greatly benefits the handicapped users as they simply provide their cardiac movements to authenticate themselves instead of openly telling people their password.

As mentioned earlier the BVP waveform varies by person as it is a unique identifier, however it is important to familiarize ourselves with a typical BVP waveform first. As can be seen in **Figure 1**, the BVP waveform consists of two peaks which define the pulses (P1 and P2) that is the measurement of relative blood flow. The intervals between the two peaks is the measurement of the heart rate in milliseconds, this is also called the pulse rate [**?**]. The V is the indicator of the blood volume during the specific time. This waveform is measured with time as its x-axis and the amplitude as its y-axis.

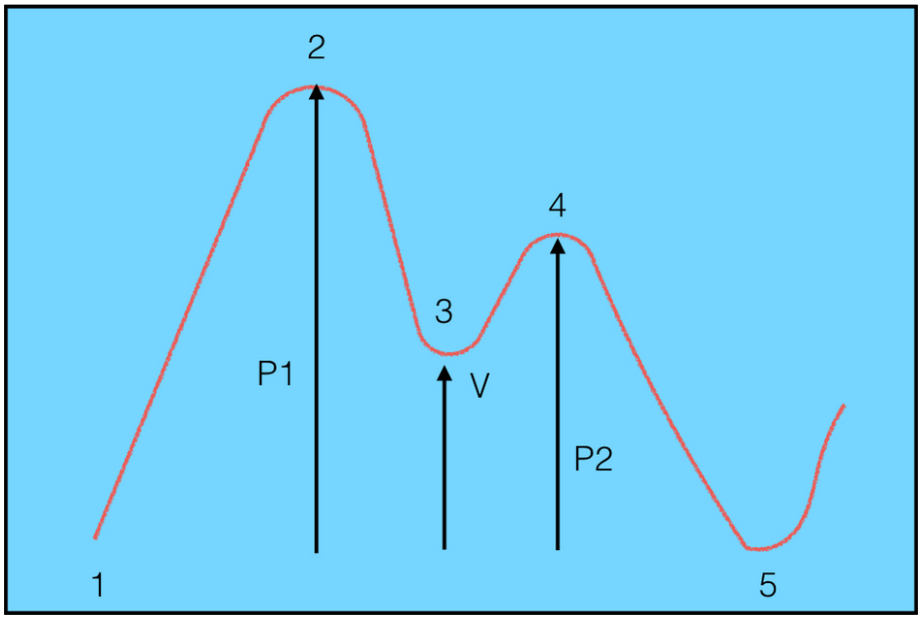

Figure 1: Typical BVP Waveform

To form unique BVP waveforms, we used data that was gathered in a previous project where the ballistocardiogram was derived [**?**]. The authors of the paper gathered data from subjects sitting still, while the sensors which were placed on a head-mounted device captured their subtle head movements. With this data, we derived the BVP waveform for each axis of the accelerometer and gyroscope for each fixed-length segment. Then, the BVP waveforms were used to train the convolutional neural network. We employed a CNN for training the model as this allows us to use raw data without extracting any of the features which then also helps prevent data loss [**?**]. A CNN also allows for a higher percentage of accuracy for our results analysis, and requires little pre-processing as the network learns the filters instead of hardcoding themselves manually.

Prior to entry into the CNN discussed above, we first had to determine a method of processing the raw accelerometer and gyroscope data we had available to us into the desired BVP signal. In the paper [**?**] the researchers from Massachusetts Institute of Technology enumerated a method for extracting heart rate accelerometer and gyroscope data. The first step to extract the desired

heart rate signal was to use an averaging filter to smooth the data. They determined empirically that the window size for the rolling average should be 1/7th of a second for their data set. In [**?**], the authors empirically determined that for the data collection method we used the window size should be 0.06 seconds. Following the averaging filter, [**?**] mentioned that a "band-pass Butterworth filter of order two, with high and low cut-off frequencies of 4 and 11 Hz respectively was applied to isolate the BCG changes". Finally they took the root sum square of the different sensor components (accelerometer and gyroscope X, Y, and Z) and applied one more filter of "order two with cut-off frequencies of 0.66 and 2.5 Hz (corresponding to 40 and 150 beats per minute)" [**?**]. Once these filtering processes are complete the remaining waveform should be that of a normal heart rate and ready for entry into the CNN.

Once the training component is completed, it is vital to choose an optimization technique which will allow the top performers of each generation to proceed to the next round. To implement this, we used the genetic algorithm, which is a search heuristic motivated by the theory on natural evolution in which the fittest individuals are selected as the best performers to proceed to produce offsprings in the next generations [**?**]. With many iterations, this method allows for us to find a generation with the fittest individuals. There are five general phases that are implemented in the genetic algorithm. The first one is the initial population, which is the original set of individuals where each of the individuals can be a solution to the problem that is being solved using the algorithm. The second phase is the fitness function in which a fitness score is given to determine how fit a given individual is, this helps in ranking the population. The third phase is selection where we select the fittest individuals from each of the generations to pass their genes onto the next one. The fourth phase is called crossover, where a crossover point is chosen for each of the parents that will pass their genes to the next generation. And, the final phase is mutation in which some of the genes of the offspring are flipped bit strings in order to maintain diversity in the population [**?**].

3 Problem Statement

Modern technology is making information accessible electronically, and accordingly ease of user authentication is becoming a more pertinent problem. Encumbering a user by forcing them to use a long written password every time they need to access their personal information makes it less convenient. Furthermore, for disabled users current authentication methods are prohibitive, for example it is difficult for a handicapped person to enter a password and impossible for other authentication methods such as fingerprint scanners. To address this problem our research explored whether an accelerometer and gyroscope could be used to derive a BVP signal from a user and use this user-unique signal to authenticate them. The successful implementation of this system would allow for seamless authentication for any user with a heartbeat. Additionally, this method prevents adversaries from gaining access to privileged accounts as mimicking cardiac rhythms of a subject is nearly impossible.We then built a prototype in order to demonstrate the authentication system which is described in the *Future Work* section below.

4 Approach

Our approach is broken up into four main stages. Data Collection describes the process through which we obtained our data along with its format. *Data Preprocessing* explains how raw data from different sensors was synchronize, as well as the different filtering techniques that were utilized to obtain BVP and BCG signals. *Model Training* goes over the training specification for our model and justifies the number of epochs that we train with. Finally, *Authentication* reveals the classification process and its evaluation.

The purpose of the project was to create an authentication system based on the user's head movements caused by cardiac rhythm. In such a system, the initial input is filtered into a more significant signal that can be used as the new input for a machine learning classifier that will distinguish between different users. Our system design starts by obtaining raw data of the user's head movements, utilizing a gyroscope and an accelerometer. These raw signals are then filtered using bandpass, rolling average and normalization filters which provide us with new and more meaningful waveforms: BVP and BCG. Once the filtered data has been generated, it is divided into overlapped groups or 'windows' of contiguous data points, a technique used to generate more data points, thus increasing the size of the dataset. At this point, we proceed to the training of the CNN classifier using the user's data as positive points, and other user's data as negative points. Once the CNN classifier is trained for the specific user, it can take new data and determine whether or not the signal belongs to the user. The overview and a visual representation of this process can be seen in Figure 2 below.

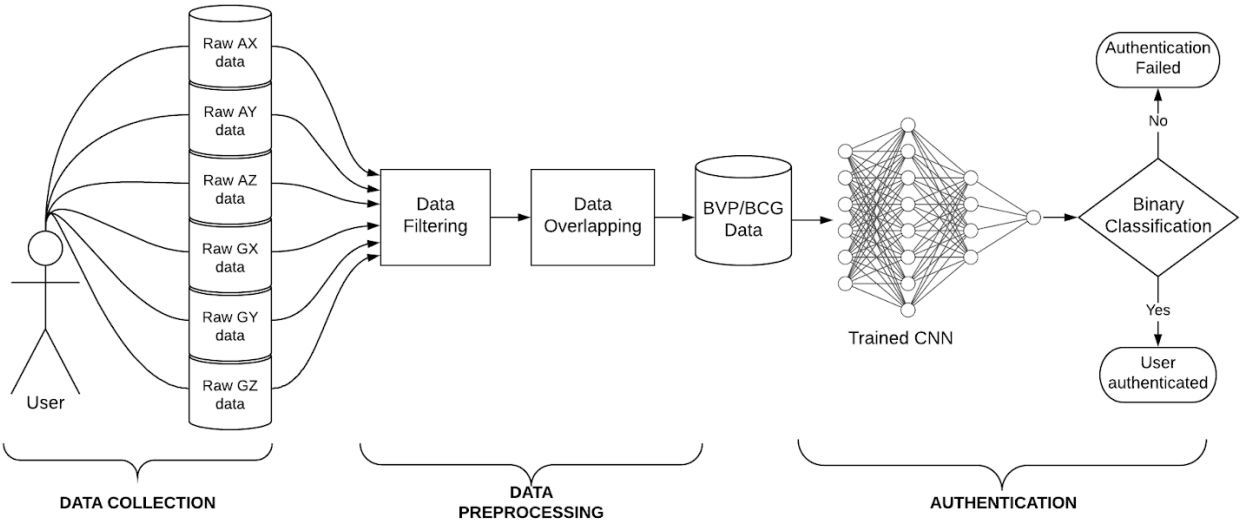

Figure 2: Authentication Model Overview

4.1 Data Collection

The data used for this experiment was collected in a previous study [**?**] in which the researchers recruited 12 able-bodied subjects and used a Smart Eyewear device (Google Glass) to collect accelerometer and gyroscope data. From each subject, three sessions of 10 minute data were obtained, which were broken down into two-minute segments of data to ensure the subject remains very still for the recording period. The three different sessions for each user where collected 15 days apart in order to later test the model's classification accuracy over time. The accelerometer and gyroscope sensors on the Google Glass had a sampling rate of 50 Hz. Through the data collection, the researchers obtained three accelerometer and three gyroscope raw sensor streams, each stream

corresponding to a different axis.

4.2 Data Preprocessing

After having acquainted ourselves with the data, we explored methods to produce a final model to authenticate users based on their BVP and BCG signals. We first filtered the raw data collected from [**?**] and applied filtering methods from [**?**] in which the researchers use a similar signal processing to estimate heart rates from head motions in their subjects. The following steps were applied differently to obtain BVP and BCG, and can be seen in **Figure 3** below. (1) *Rolling Average Filter:* We first implemented a rolling average filter that subtracts a 35 sample rolling average for BCG, and 3 sample rolling average for BVP from each dimension of the vector, in order to remove sensor shifts and potential signal trends. (2) *Butterworth Band-pass Filter:* To isolate BVP changes, we used two butter bandpass filters. First, a 4th order filter with the high being 13 Hz and low being 10 Hz, followed by a 2nd order bandpass filter with the high and low being 2.5 Hz and 0.75 Hz respectively. Similarly, in order to isolate BCG changes, we added only one fourth order butter bandpass filter with a high of 11 Hz and a low of 4 Hz. (3) *Normalization:* Finally, each sensor stream was normalized to have unit variance within each window.

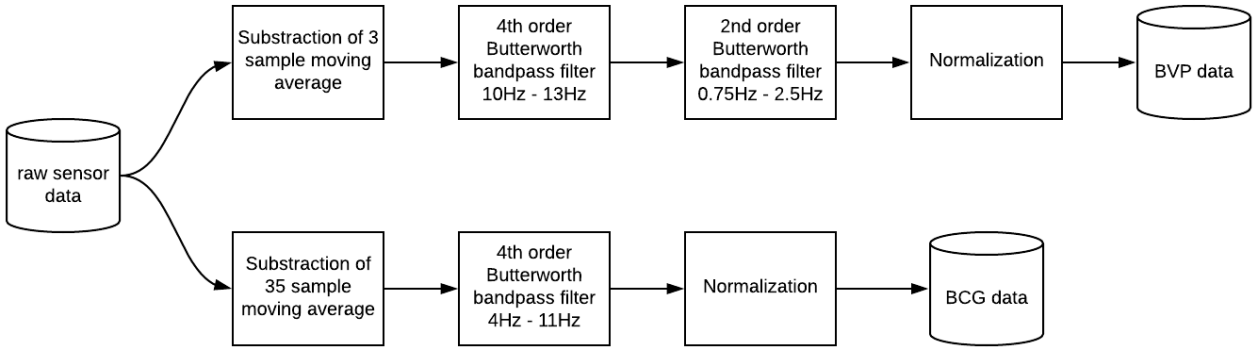

Figure 3: Deriving BCG and BVP from Raw Sensor Data

We then proceeded to prepare the processed data for training in the CNN. The first step in preparing the data was to group it into windows of w length in seconds, also referred to as window size. Two sequential segments have an overlap with each other of w - 1, as inspired by [**?**]. For instance, two consequent segments of $w = 5$ seconds would share a total of 4 seconds. Overlapping allowed us to create more data points for the CNN training, hence increasing the accuracy and making overfitting less likely as discussed in [**?**], a paper in which the researchers train a CNN for ballistocardiogram-based authentication. At this point, it was unclear which optimal segment length would produce the best results, so as described in the Parameter Selection section, we tested the performance of 2, 3, 4, and 5 second long segments respectively.

4.3 Model Training

At this point, we constructed a machine learning model that was capable of learning the unique characteristics of each user's BVP and BCG waveform, in order to be able to authenticate the user at a later time. We implemented a CNN classifier as our authentication model. As studied in [**?**], CNNs are precise models for effectively classifying time-series, such as the BVP and BCG waveforms. By training the model for each user, we end up with a series of customized subject-specific models that will differentiate the user they were trained for, from the rest. Hence, before being authenticated, each new user will have to provide head motion data to create a trained CNN that will then be utilized when such user attempts to authenticate. The model was trained for each user using BCG and BVP data separately, in order to see which waveforms yields better results.

For the training, we use the data obtained after the Data Processing stage, which is the original raw data from the accelerometer and gyroscope, that has been synchronized, filtered and divided into windows of w length, where w is the length of the window in seconds. Similar to [**?**], our CNN topology is designed taking into account the relationship between the three axis of accelerometer and the gyroscope derived waveform. For that reason, each window was rearranged as a **2 x 3 x (*w* x 50)** tensor that could be used an input to the CNN. This tensor is divided into three dimensions, each one with a particular purpose. The first dimension is based on the number of measurement sources, accelerometer and gyroscope are **two** different sources. The second represents the number of axis per source, so x, y and z are the **three** axis provided by each sensor. Finally, the third dimension is **(*w* x 50)** because it refers to the number of data points per segment, where w is the length of each window in seconds, and 50 (in Hz) is the sampling rate of the sensors.

Only the first four segments of the first session were used to train the model for each user. As described in [**?**], the data from the user's own segments was labeled as positive, where the data from all the other subjects was labeled as negative.

4.4 Authentication

Once the model was trained, we used it to perform the final authentication step. To authenticate a user we collected raw accelerometer and gyroscope data in x-second segments. Each segment represents one authentication attempt and we collected one or more of these. Once the accelerometer and gyroscope data was collected, we processed it in the respective signal that we were trying to authenticate for (BVP or BCG). Note that the authentication model must be trained for BVP or BCG in the training step, they are separate models and are singular in purpose once trained. Additionally each model was trained to authenticate a single user, so for example model A could be trained to authenticate User 1 using the BCG signal. The processed signal of choice is then the input into its corresponding model. The authentication model will take one of the segments and output a confidence value c where $0 \leq c \leq 1$ and c represents how confident the input signal belongs to the user it has been trained for. If $c = 1$ then we can assume that the authentication model is 100% sure that the signal belongs to the correct user. If $c = 0$, then that would indicate that the model is 100% sure that the signal does not belong to the correct user. Once this confidence value is calculated it is compared against a threshold value, T where $0 \leq T \leq 1$ and describes the minimum confidence value required to consider a user authenticated. High values of T will decrease the chances of false positives but also increase the chance of a false negative. In other words the higher the T value the more secure the system is but the more difficult it will be for the correct user to be authenticated. Conversely low values of T are less secure but more forgiving of a user whose data does not exactly match the model it has trained for. As mentioned above, each segment of data counts as an authentication attempt and if any segment authenticates, then the user will be authenticated. In order to test the validity of our models we used ROC curves, which measure the false acceptance rate vs. the true acceptance rate across multiple values of T. The details of this metric and the results that they produced are discussed later in section 6.2.

5 Parameter Selection

In order to determine the most appropriate window size, as well as the best CNN hyperparameters for training, we created numerous different models and trained them against all our user data, for different window sizes, using a genetic optimization method. Finally we chose the optimal number of epochs to train the final model. The parameter selection section is broken up into four main stages. *Genetic Algorithm* describes the process we took in generating the different models and then applying the genetic algorithm in order to find an optimal parameters. *Parameter validation* sections explains the different parameters that were set such as window size, and the neural network attributes. *Final Models*, which explains in detail the final CNN models, which are built using the the hyperparameters that were previously generated. the end of the section along with a detailed analysis. Finally, *Number of Epochs* presents the number of epochs that the final models will be trained for, as well as how we derived that number. Below in **Figure 4**, we can see an overview of the genetic algorithm for parameter selection, which will be described later in greater detail.

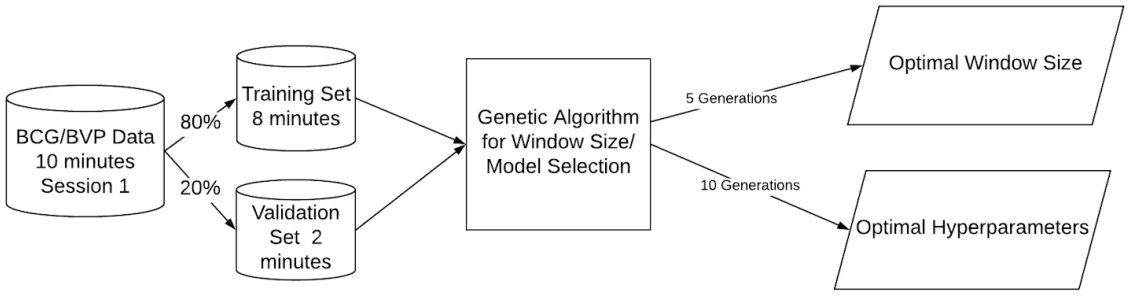

Figure 4: Overview of Genetic Algorithm Approach

5.1 Genetic Algorithm

As described in the *Background* section, the genetic algorithm is an optimization technique that consists of repeating a process over multiple generations or iterations, in which the top performers of a generation will pass on to the next generation along with slightly mutated versions of other models. We implemented this approach using the following procedure for parameter selection.

First the genetic algorithm generated 20 models, each with randomized parameters. We chose to use 20 models because the genetic algorithm is considerably time intensive and proportional to the number of models that are being used for training. Using 20 models is an artifact of our computational and time restraints, ideally we would train with more models than this if we had the resources to do so. The algorithm generates the parameters of the initial generation models randomly. Once we had the 20 models with randomly generated parameters, for each of our 12 users we trained each of the 20 models at 10 epochs per training using the first four segments of the first session. Once the models were trained we used the fifth segment from the first session as positive test points and the fifth segment of the first session from the other 11 users as negative points. Once all the training and testing was completed each model received an aggregate score that indicates how well it performed across all the users. From each test, we obtained the false acceptance and rejection rate, as well as the true acceptance and rejection rate. This score is calculated by adding the average false acceptance rate to the second power with the average false rejection rate to the second power. This entire process of training, testing, and scoring the 20 models is known as a generation. From the scored models the top five (Elitism) and the worst three models are passed on for reuse in the next generation. The parameters of the remaining 12 models are mutated randomly. After a finite number of iterations (when the algorithm terminates), the best model of the last generation is saved, since it is the best model so far explored by the algorithm.

5.2 Parameter Validation

A convolutional neural network is composed of several layers, each one consisting of a number of parameters that affect the way input data is processed as well as the performance of the resulting model. A relevant portion of the cross validation process was selecting the best parameters such that they were optimized for the correct classification. The length of the input data segments (window size), as well as their overlapping window size, the number of convolutional and dense layers, the activation function and number of neurons per layer, the dropout regularization values, as well as the keras optimizers and loss functions were all included in the parameter selection.

5.2.1 Window Size

The window size is length of the data segments that are fed into the CNN, as well as the size of the overlap that was performed in the preprocessing step. Indeed the size of the input data can affect the overall performance of the neural network. For that reason, before even choosing the attributes of the neural network itself, we focused on establishing which window size would yield the best results. To determine this we ran the genetic algorithm for five generations, for window sizes 2 through 5. We chose five generations because each generation at 10 epochs takes approximately 12 -15 hours with the computational resources we had available to us. Moreover, five generations were enough to get an early idea of which window size could potentially achieve the best scores. After running the genetic algorithm for window sizes of length 2, 3, 4, and 5 seconds, we determined the window size with the best performing model, for both signals.

It is shown in **Table 1** below that the highest accuracy among the best performing models for every window size is that of five seconds, for the BVP signal.

Window Size (seconds)	2	3	4	5
TAR	96.16	95.86	97.12	96.43
TRR	87.59	90.45	89.39	95.01
Accuracy (%)	91.875	93.155	93.255	**95.72**

Table 1: Top model accuracies by Window Size for BVP

It is shown in **Table 2** below that the highest accuracy among the best performing models for every window size is that of five seconds, for the BCG signal.

Window Size (seconds)	2	3	4	5
TAR	97.02	98.21	97.84	99.20
TRR	91.16	92.10	92.63	93.97
Accuracy (%)	94.09	95.15	95.23	96.58

Table 2: Top model accuracies by Window Size for BCG

5.2.2 Hyperparameters

Once the window size was selected, we repeated the process once more to determine the best CNN attributes. As show below in both **Figures 5 and 6**, 10 generations of the genetic algorithm were generated for window size 5, for each signal. Thus leading to the best topology parameters that would be used to authenticate each user.

More specifically on the BVP signal we can see below in **Figure 5**, how we reach a peak on the 9th generation. This way the best model is produced using the topology of the of the specific (out of 20) model which performed the best in the 9th generation.

Additionally in the BCG signal we can see below in **Figure 6**, how we reach a peak on the 7th generation. This way the best model is produced using the topology of the of the specific (out of 20) model which performed the best in the 7th generation.

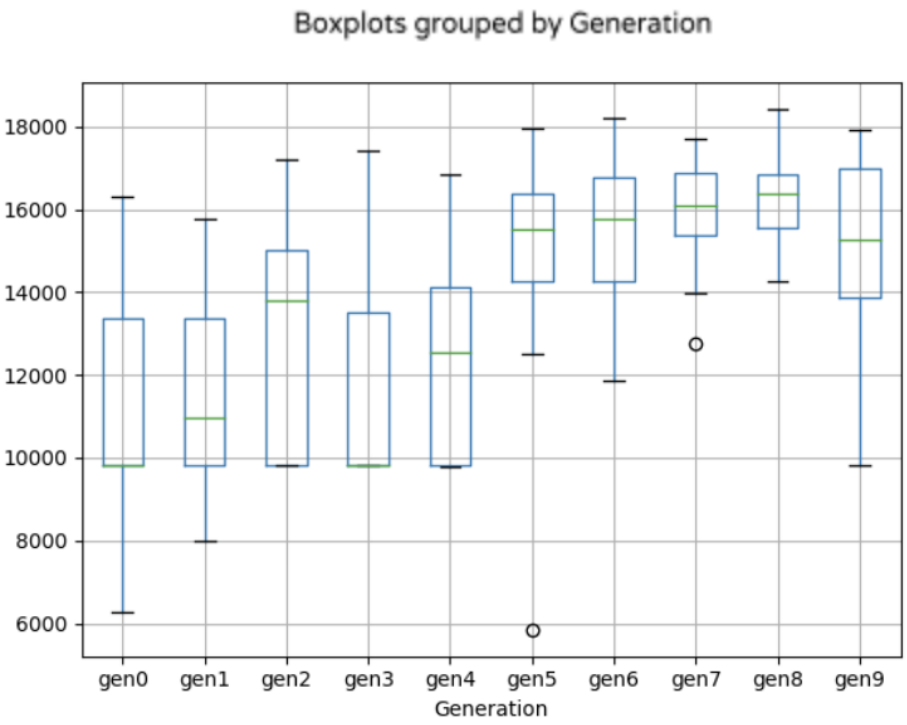

Figure 5: Boxplot for Window Size 5 BVP

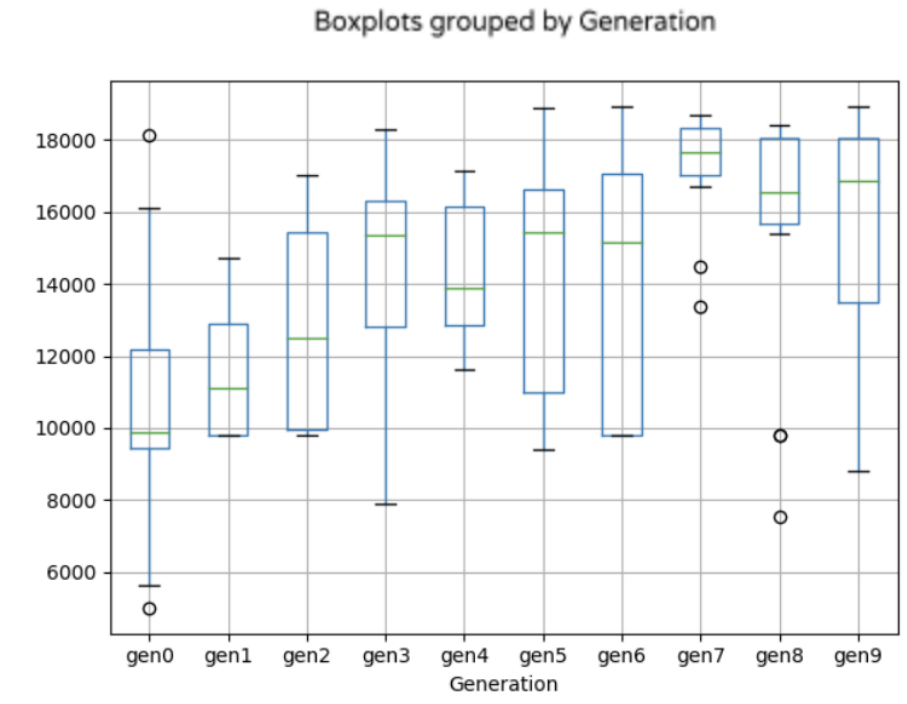

Figure 6: Boxplot for Window Size 5 BCG

5.3 Final Models

After the final attributes of the CNN were selected, the last step in order to train the model was to construct it with the previously selected parameters. Below in **Figure 7**, we can observe the how the Convolutional layers are deployed, this set-up is actually identical for both the BVP and BCG, showcasing that the classification methods of such signals are similar.

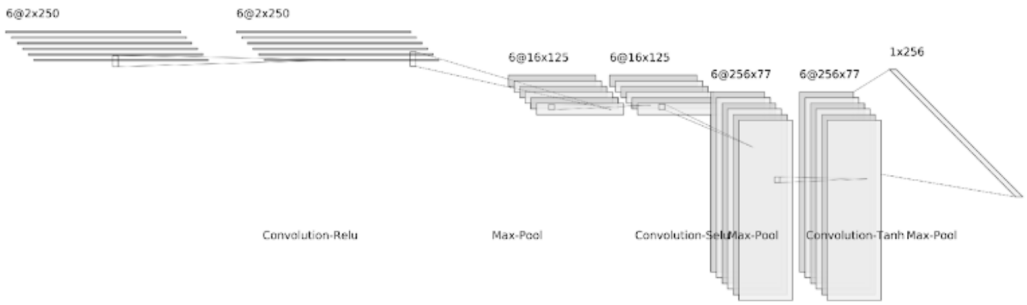

Figure 7: CNN Final Convolutional Layers set-up

We can also observe in **Table 3** the hyper parameters for the Dense Layers, which also happen to identical for the classification models of both signals. Finally it is worth mentioning that the classifiers of both signals have a 20% dropout between the dense and the rest of the layers, while the loss function is what differs between the two models. The BCG model uses the Mean Absolute Error function, and the BVP model uses the Mean square error function.

Layer	Neurons	Activation Function	Dropout
Dense 1	128	TANH	0.4
Dense 2	128	RELU	0
Dense 3	512	TANH	0.2
Dense 4	1	SIGMOID	0

Table 3: Hyperparameters for the dense Layers

5.4 Number of Epochs

After selecting the optimal hyperparameters for building the CNN, and the best window size, all that is remaining in terms of parameter selection is the number of epochs that the CNN will be trained for. In order to determine this parameter we decided to train for 100 epochs, a number that was within our computing capabilities, while keeping track of the training accuracy of the models, which we later plotted as shown in the **Figure 8** below.

As seen in **Figure 8** below, the training accuracy spikes up during the first 20 epochs, and then starts to consolidate until it stabilizes at 50 epochs. Hence, training for 100 epochs yields a very good accuracy.

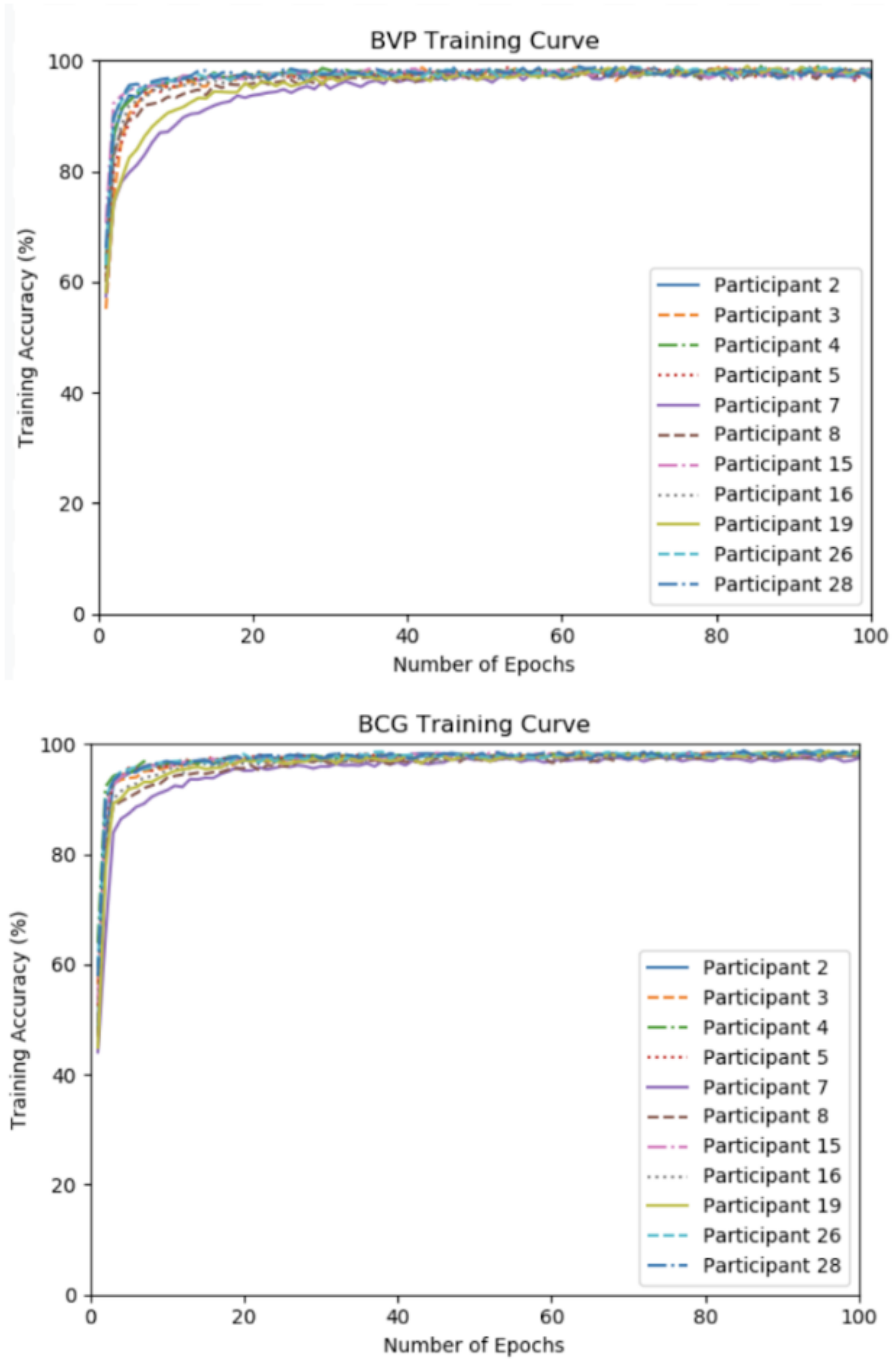

Figure 8: ROC Curves for Training of BVP (top) and BCG (bottom) Models

6 Results

Our *Results* sections is broken into three subsections. *Metrics* describes how ROC curves are used to evaluate models in order to determine a good one from a bad one. The Genetic Algorithm Results Section describes and evaluates the performance of the genetic algorithm for all the window sizes, as well as the whole performance of the full 10 generations of both the BVP and BGC. The last two sections *Model Performance Analysis BVP* and *Model Performance Analysis BCG* present our results for each of the waveforms along with an analysis on their performance. After the final models have been optimally selected, the window size is chosen, and the models are trained for 100 epochs, using the first four segments of the first session, testing is the final step to be done. There is a total of three sessions, and the first session is data collected from the participants on day one, the second session is data collected on day 15 and the last session contains data collected 30 days after the initial collection day. Other than the 12 participants shown for whom models were generated, there were another 10 participants whose data was used to form the external sets. The purpose behind that is to introduce some outliers to the system in order to more accurately describe the authentication ability. Using both the external and validation sets we generated the results which are described below.

6.1 Metrics

In a Receiver Operating Characteristic **(ROC)** curve the true accept rate (TAR/ Sensitivity) is plotted in function of the false accept rate (FAR/Specificity) for different cut-off points. Each point on the ROC curve represents FAR/TAR pair corresponding to a particular decision threshold. A test with perfect accuracy has a ROC curve that passes through the upper left corner (100% TAR/Sensitivity, 100% FAR/Specificity). Therefore the closer the ROC curve is to the upper left corner, the higher the overall accuracy of the experiment will be [**?**]. In order to consider the accuracy in a more rational way we will be comparing the area-under-the-curve **(AUC)**. Thus the higher the AUC the higher the accuracy of the model.

Finally for the performance analysis of the model we will be comparing the the validation set with the external set for both signals. The validation set refers to the set of data that is used for training the neural network as opposed to the external set which is the set of data that is never seen by the neural network until the very end, and it is then used to determine the final unbiased accuracy of the model. Thus the comparison between the performances of the two sets is critical to determine a reasonable performance of the model. This means that the accuracy of the external set has to be slightly lower than the accuracy of the validation set. When the accuracy of the external set is significantly less than the accuracy of the validation set, then we have overfitted, meaning that the model is biased and will not be behaving as it was intended to. Whereas if the external set has a higher accuracy than the validation set, that is simply irrational, and in such a case there must be a mistake in our metric methods.

6.2 Model Performance Analysis BVP

In this section we will be testing the accuracy of the model that is using the BVP signal. For that we are using all three sessions of data. For each test we have the validation set which is testing the 12 models using the first four segments of all the 12 "known" users, that is data that the models were trained with. Whereas the external set consists of the data from the 5th segment of every known user as well as all the "unknown" external users, that is data never seen by the models.

As seen in **Figures 9, 10 and 15** below, the ROC curves are produced based on the performance of the 12 different participants. Some specific observations are that participants 7 and 8 are outliers in sessions 2 and 3 respectively, as can be seen in **Figures 9 and 10** below, therefore we infer that the participants did not meet the criteria of the data collection during their respective data collection sessions.

Specifically in **Figure 14** we can observe the ROC curves for the BVP Validation and External Sets for the first session, the AUC of the external set is very high and also expectedly close to the AUC validation set, since there is no overfitting. We can also observe that participant 16 is outlying, thus we can conclude he or she did not meet the criteria of the data collection.

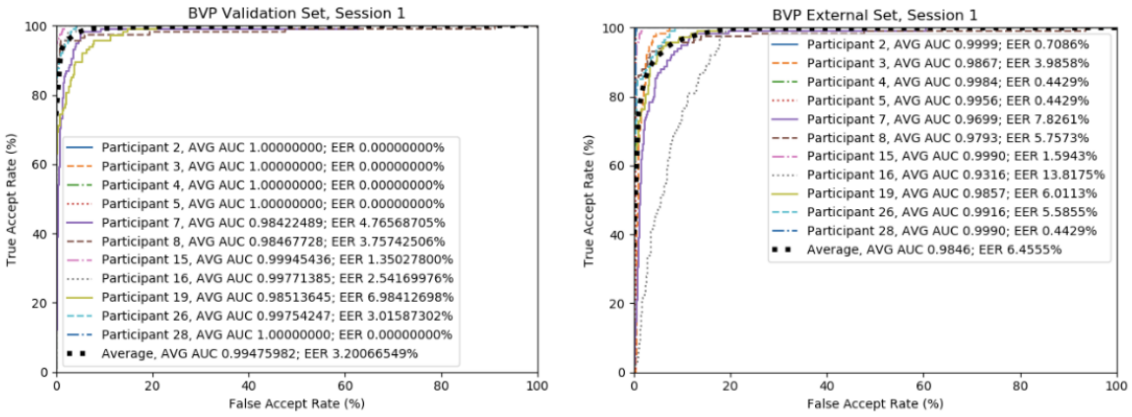

Figure 9: BVP Session 1 Validation (left) and External (right) Sets

In **Figure 9** above we can observe that there is no overfitting as the accuracy of the external set is not lower than the accuracy of the validation set. We can also observe that now there are many participants that are outlying such as participant 7, 8, and 16 which are represented by the purple line, the maroon dashed line, and finally the extremely dashed blue line. As we mentioned above, outliers do not meet the criteria for many different reasons such as the days in between each session, and if the participant was in a different mood than their later data collection sessions.

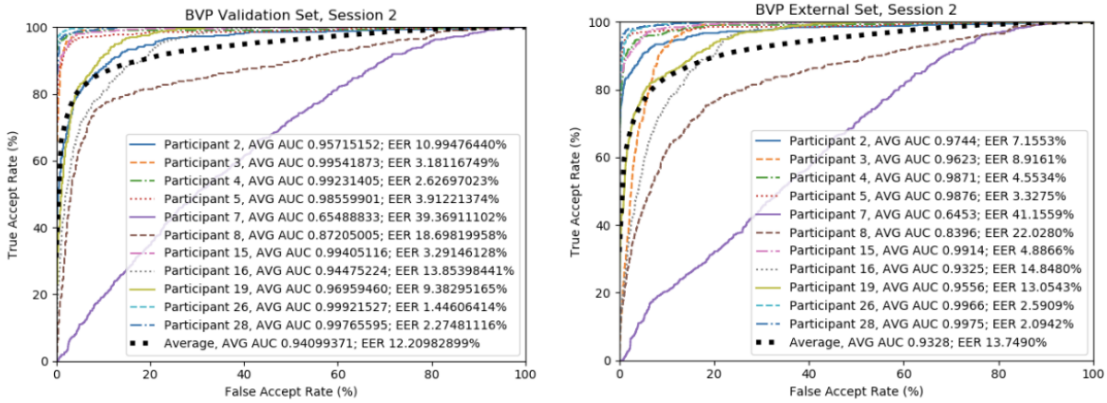

Figure 10: BVP Session 2 Validation (left) and External (right) Sets

Figure 10 shows that there is no overfitting similarly to **Figure 9**. We can also observe that participant 8 is outlying in both validation and external set, thus judging from **Figure 11** we can hypothesize that the BCG signal extraction could not possibly work for participant 8 as they are an extreme outlier and the authentication system would not correctly recognize them every time

We can see overall how the average AUC does not significantly drop from the first session up until the third, displaying how such an authentication system can perform well over time.

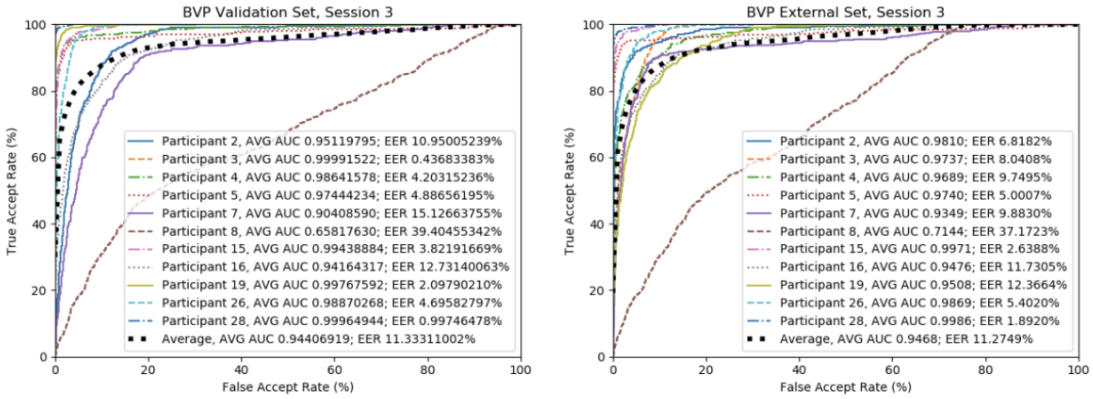

Figure 11: BVP Session 3 Validation (left) and External (right) Sets

6.3 Model Performance Analysis BCG

Similarly as seen in the BVP results the average accuracy starts very high in the 1st session and decreases as we move to the third session. Also the AUC of every user decreases from the validation set to the external set (except for participant 19, who behaves abnormally in sessions 2 and 3 below). Finally we observe that the BCG results from session 1 have a higher AUC by 4% for the BVP vs BCG results.

In **Figure 12** below we observe the ROC curves for the BCP Validation and External Sets for the first session, the AUC of the external set is very high and also expectedly close to the AUC validation set, since there is no overfitting.

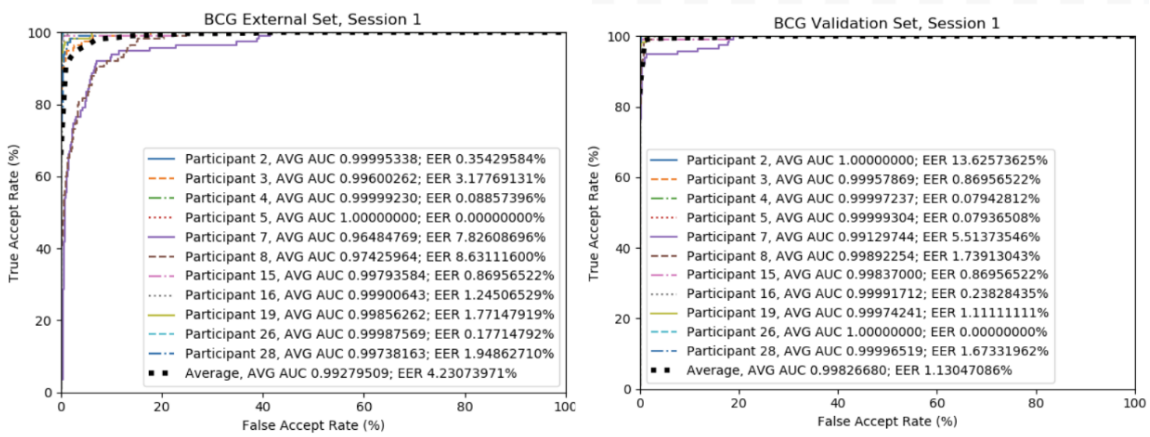

Figure 12: BCG Session 1 Validation (right) and External (left) Sets

In **Figure 13** below we observe again that there is no overfitting. We can also observe that now there are many participants that are outlying such as participant 7 with the purple line, and participant 19 represented as the lime-green line.

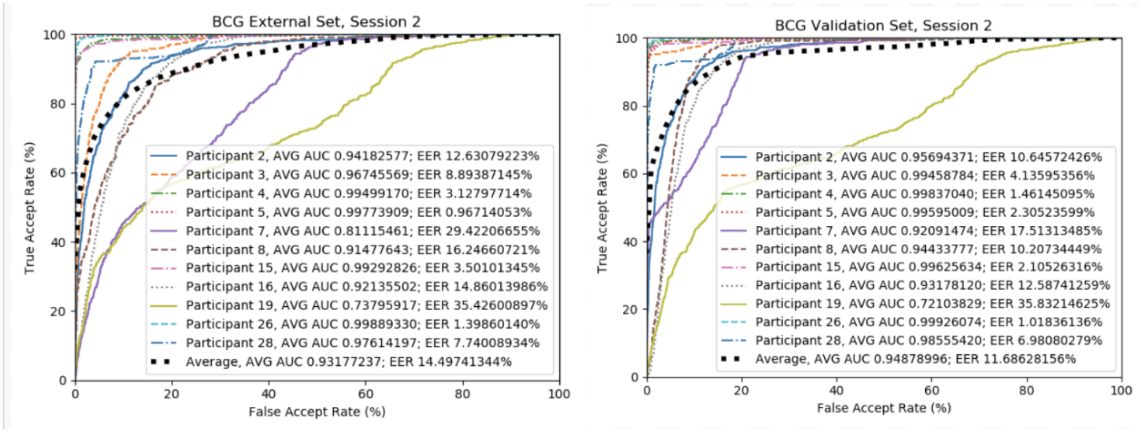

Figure 13: BCG Session 2 Validation (right) and External (left) Sets

We observe in **Figure 14** below again that there is no overfitting. Also there is a higher variance between the users which is reasonable since the model is tested with Session 3 data which was collected a lot later that the data the model was trained with.

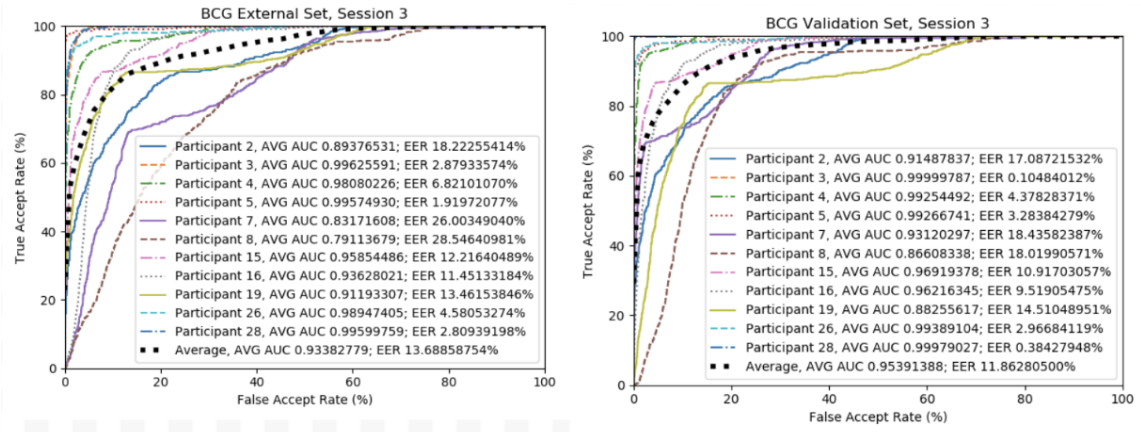

Figure 14: BCG Session 3 Validation (right) and External (left) Sets

We can see overall how the average AUC does not significantly drop from the first session up until the third, displaying how such an authentication system can perform well over time. Also in comparison to the BVP model, the BCG model has higher accuracy and fewer outliers in general.

7 Discussion

In this section we discuss the validity of obtaining each signal (BVP and BCG) from accelerometer and gyroscope data based on our results. This discussion is broken down into two parts (1) *Empirical Mode Decomposition* which talks about the methods we used to process the raw data into more complete signals, and (2) *BVP Results* in which we discuss why the BVP results were not fit for continuing use whereas BCG was.

7.1 Empirical Mode Decomposition

In order to analyze the signal and ensure that the filtering produced the desired waveform (BVP or BCG) we made empirical comparisons to a 'typical' example of the desired waveform. To accomplish this, we first had to filter out all the noise in raw data. For noise filtering we used a technique called Empirical Mode Decomposition **(EMD)**. EMD is a method for breaking down a waveform into its component functions, while remaining in the time domain [**?**]. The constituent functions known as Intrinsic Mode Functions **(IMF)** represent filters applied at different frequencies, thus by combining all the IMFs together, the original signal is formed. After running the EMD on the signals, we manually combined IMFs with similar frequencies to produce new signals. We then qualitatively compared the newly produced signal to the characteristic desired signal to find the most promising combination of IMFs. The matlab code we used to perform the EMD was provided by Rice University.

To produce the BVP IMFs for each axis and each sensor we decomposed the signal into its constituent IMFs and an observation we made was that the lower frequency IMFs (IMF 4 - 9) represented mostly noise as the frequency bands they occupied were well below that of a normal heart rate. These IMFs can be seen in the second and third rows of **Figure 15** below.

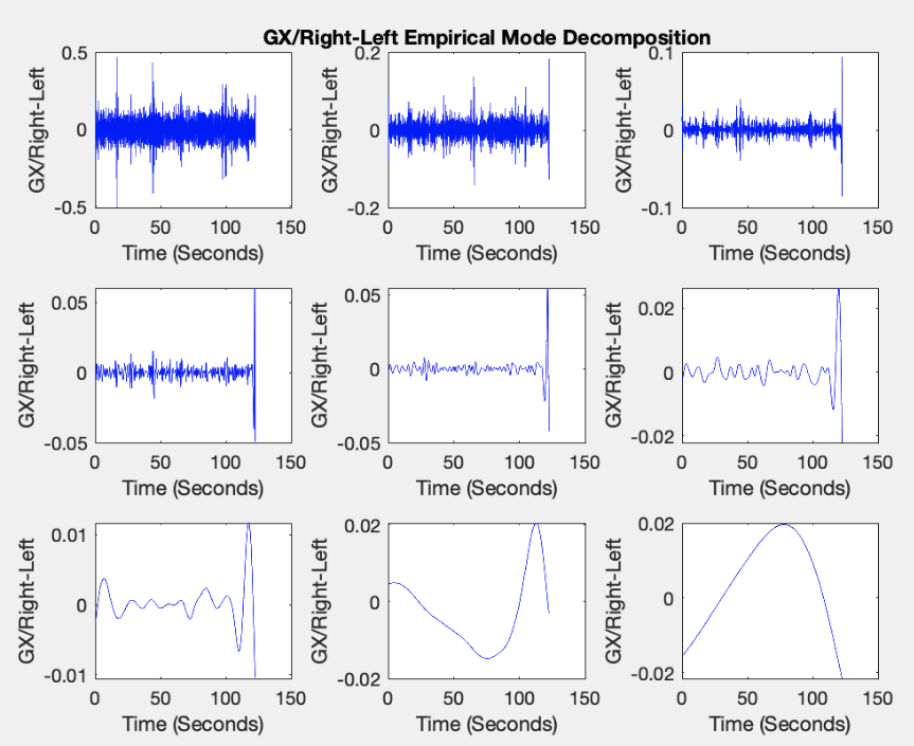

Figure 15: BVP IMFs for the Gyroscope X-Axis

Due to this observation we were able to determine that only the first three IMFs (the first row of IMFs in **Figure 15**) were relevant to the noise filtering of our signal. To find the best combination of IMFs we empirically compared the combinations to that of a heartbeat. Below in **Figure 16** we

can see IMF 1, and the combinations of IMFs 1-2, and IMFs 1-3. We found that the combination of IMFs 1-3 produced the closest waveform to BVP with some portions of the graph imitating a typical BVP signal. We found that the gyroscope X-axis best represents the desired signal.

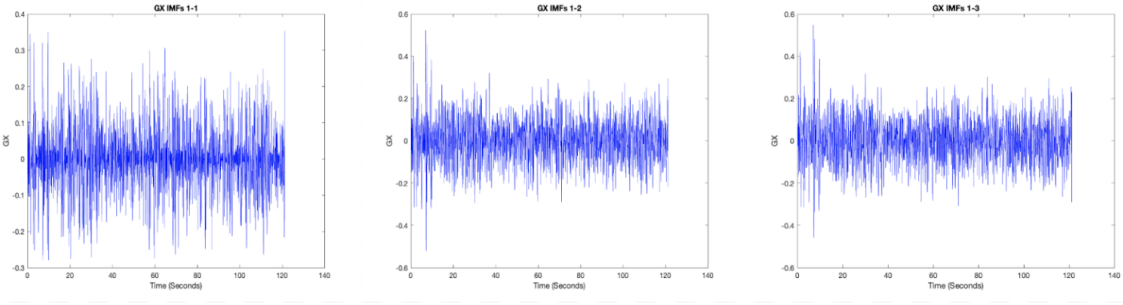

Figure 16: BVP IMF Combinations for Gyroscope X-Axis

Although this best exhibits BVP from the data we collected, as can be seen in the above graphs there is not a clear heartbeat-like pattern that would discern these results as strictly BVP. Although the basic shape of the signal is correct, the intervals at which they occur do not make sense given that BVP is meant to model a heartbeat. The complete set of decomposed and recombined signals can be found in Appendix 10.

Similarly to BVP, for each axis for each sensor we decomposed the signal into its constituent IMFs for the BCG signal. Additionally, just like BVP, the best IMFs for use in creation of a cleaner signal were the lower frequency IMFs (IMF 4 - 9). These IMFs can be seen in the second and third rows of **Figure 17** below.

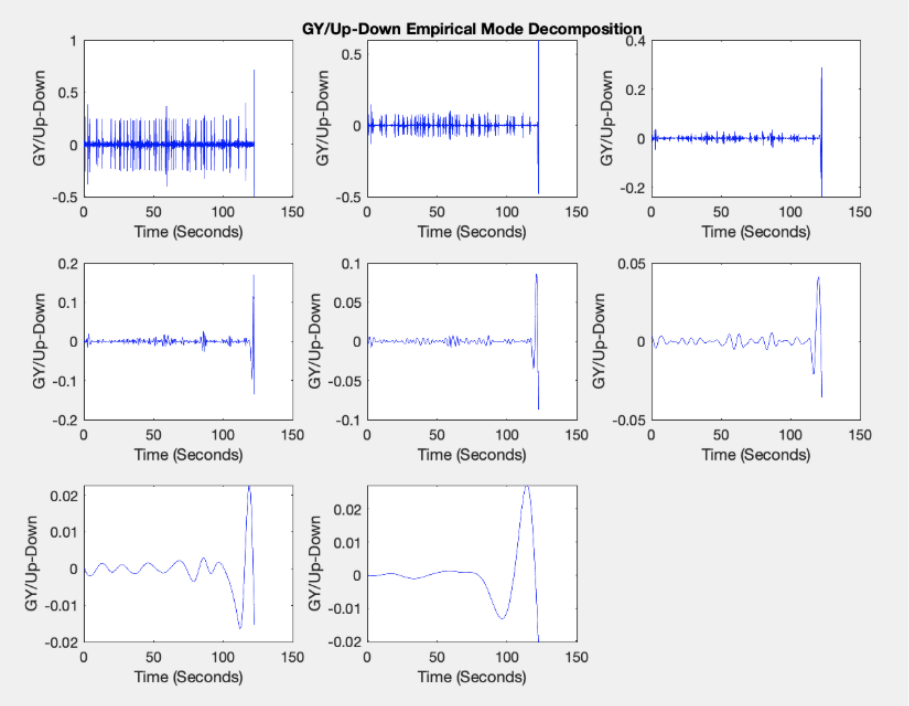

Figure 17: BCG IMFs for the Gyroscope Y-Axis

Due to this observation we were able to determine that only the first three IMFs (the first row of IMFs in **Figure 17**) were relevant to the noise filtering of our signal. To find the best combination of IMFs we empirically compared the combinations to that of a heartbeat. Below in **Figure 18** you

can see IMF 1, and the combinations of IMFs 1-2, and IMFs 1-3. We found that the combination of IMFs 1-3 produced the closest waveform to BCG with large portions of the graph imitating a typical BCG signal. We found that the gyroscope Y-axis best represents the desired signal.

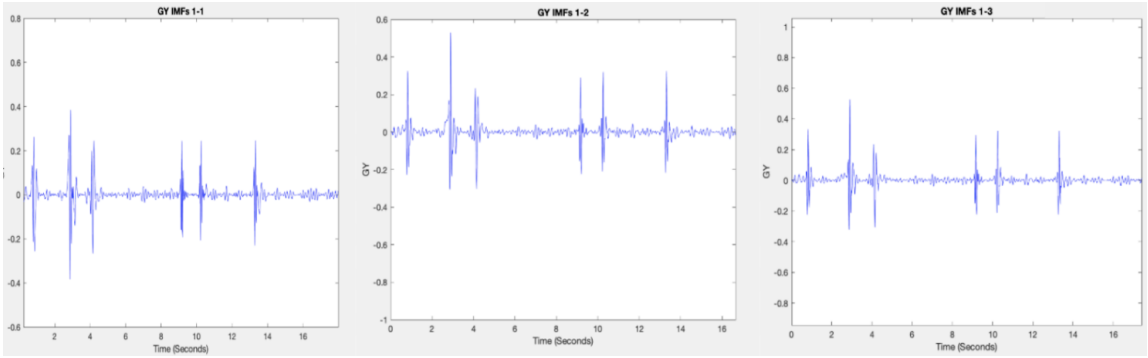

Figure 18: BCG IMF Combinations for Gyroscope Y-Axis

This best exhibits BCG from the data we collected, as can be seen above in **Figure 18**there is a clear heartbeat-like pattern that would discern these results as strictly BCG. The basic shape of the signal is correct, and they occur at approximately one second intervals corresponding to a heartbeat of 60 BPM which is relatively standard for a resting heart rate. These results combined with the BVP results allowed us to determine that using this method of filtering and cleaning of the signals, only BCG will be viable for our model going forward. While the BVP data acquired us fairly good results for the CNN, the results from this section show that the neural network is in fact fitting the specific noise produced by the Google Glass and each individual using it. The goal of this paper is not to fit noise but rather a unique cardiac related signal, therefore BCG is the appropriate signal to use going forward.

7.2 BVP Results

As seen in our results section and in **Figure 19** below, our BVP waveform does not reflect a typical BVP signal. The derived BVP waveform is seen in blue, and a typical BVP waveform is represented with the red overlapping signal. As seen above, the BVP signal that we derived very loosely matches the patterns represented in the typical BVP waveform. We deduce that this could be due to the data that was collected in the previous study using the Google Glass device. We also used a filtering technique that was gathered from a previous study which might not have been the optimal way to filter the signal to produce a BVP waveform. After implementing the filtering we also empirically decided on the axis that we would be using, and such studies have never been presented before. Also, the EMDs and IMFs produced by the raw BVP data did not give us appropriate results to derive a typical BVP waveform which further helps us deduce that we did not acquire a BVP waveform.

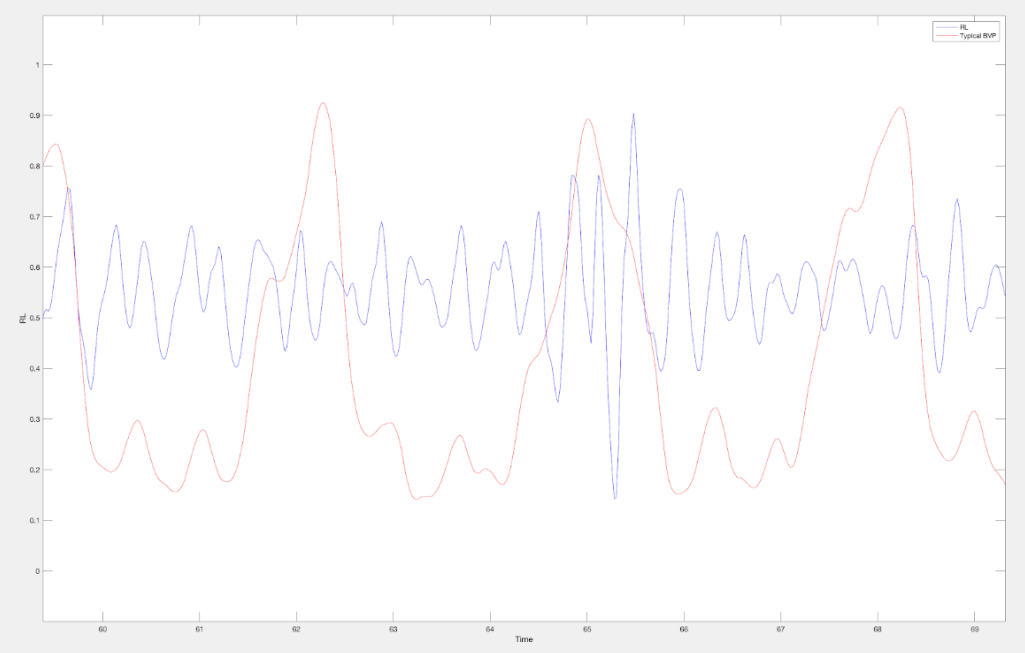

Figure 19: Typical BVP vs. Derived BVP

8 Related Work

Head mounted authentication methods using head-motion derived biometric signals have already been attempted. In [**?**] Hernandez et. al show a new method of collecting real-time cardiac biometrics from a head mounted device, Google Glass in their case. Their study shows the high accuracy of such methods and details the methods with which they are able to filter the signal into useful biometric signals such as BVP and BCG. This is closely related to our work as we use their method for filtering signals to produce BVP and BCG. However, while their work proves this method is possible and lays out usages such as monitoring biometrics in healthcare, our study addresses the possibility of using biometric signals to authenticate individuals in a passive real time and persistent manner. [**?**] by Schneegass et. al describes a method of authentication that uses bone conduction speakers to authenticate users. While this method is accurate, it takes 23 seconds to authenticate, which is unrealistically long to be a useful authentication method. Additionally, some users find the experience of bone conduction speakers playing white noise into their skull uncomfortable. In [**?**], the authors develop a method for authentication that involves tracking a users eye motions as they watch a video. This authentication method is accurate and reliable, however it requires 34 seconds of data to complete the authentication process making it cumbersome when authentication should be as close to a real time process as possible. In [**?**] the anonymous authors develop a method for authenticating a user using accelerometer and gyroscope data collected from a google glass to produce a BCG signal and train a model to authenticate users with this model. While the work done here is superior to its competitors in terms of speed of authentication (2-5 seconds) and obtrusiveness, it limits the authentication signal to BCG and does not implement the authentication model on a wearable device. Our work expands on the potential available authentication biometrics by looking at BVP as well as BCG. Additionally, we verified BCG results translate to a wearable device that we developed that is much cheaper than the Google Glass.

9 Conclusion

In this paper we showed that deriving a user's BVP signal from a head mounted accelerometer and gyroscope does not produce the desired results. The resulting signal is not BVP but rather just erroneous noise that our CNN managed to fit. The fact that this is not a BVP signal invalidates the results despite the fact that the CNN was performing well with low EERs of 6.5% and 11% after 30 days. Once we discovered the inviability of BVP as an authentication signal, we used results from [**?**] to move forward with BCG as an authentication method. We showed that the accelerometer and gyroscope data can be successfully filtered into a characteristic BCG signal and that the models trained from this signal have good EERs of 1% and 12% after 30 days. Additionally, we developed a wearable device to perform the accelerometer and gyroscope data collection and user authentication. Finally, we showed that the data collected on this device can be filtered into a characteristic BCG signal. If the EERs on the models trained on this data are acceptably low, our approach shows promise for a low-cost method to authenticate users using a wearable device such as the one we developed. In the future we intend to continue this work by using the device we developed to: (1) collect more data use that data to train a CNN authentication model, (2) develop a method to store the trained model directly on the wearable device so only the device is necessary for authentication, and (3) test the accuracy of this device, especially with relation to how its performance changes over time.

10 Future Work

Due to time constraints on our project, we were unable to complete training for our hardware setup. Regardless, we continued collecting data to in order to plot the results to see if we obtain some significant observations in order to move forward with our setup. We will continue fine tuning the hardware even after the completion of our original project. For the hardware setup, we configured a Raspberry Pi Zero W microprocessor running the Raspbian operating system due to its reduced size and ease of programming. Also, the Raspberry Pi allowed us to run our existing python code without much change. In order to record changes in acceleration and angular velocity, we used the SunFounder MPU6050 sensor which includes an accelerometer (16 g's) and gyroscope (2000 degree/second). The primary reason from choosing this sensor was the size which is 4.10mm X 4.10mm X 0.95mm and the ranges that it supports for acceleration and velocity, which is adequate to capture miniscule head movements on the user. The sensor is connected to the Raspberry Pi through a I2C serial interface. This connection is made up mostly by four connections which are a 3V VCC, ground, SCL, and SDA. Once they were all connected to the microprocessor we soldered the connections to make them as stable and reliable as possible, hence avoiding potential noise that could be caused from a weak connection. Once the hardware was configured and ready to read data, we proceeded by setting up the sensor in a container that can be placed on a person's head. To record data we wrapped the head mountable device around a headband to secure it with the wire on the head, as can be seen in **Figure 20** below. We then placed this device onto a users head with the wire connected to the Raspberry Pi on one end, and a general homeplug on the other, this is also shown in **Figure 20**.

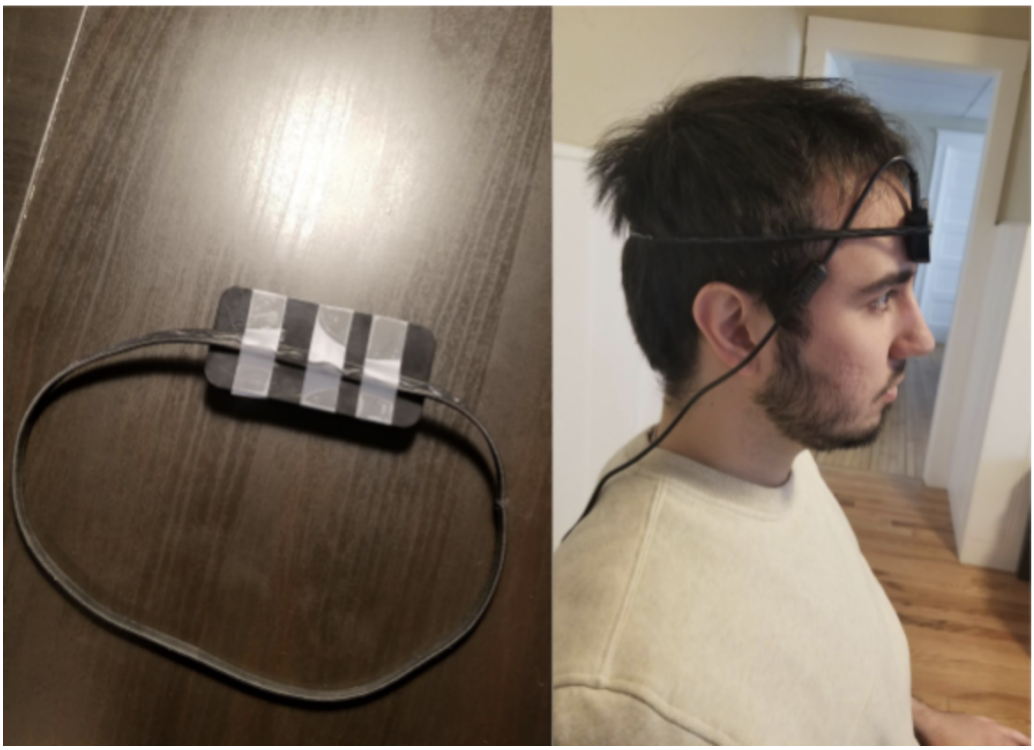

Figure 20: Headmountable Device (left) and Side Profile of user wearing device (right)

Once the device was secure on the head, we asked the user to remain still while we initially collected one session worth data of 2 minutes. We used the 'ssh' command to connect to the sensor to record the data. One person was handling the stopwatch, and another one was recording the data on a laptop while the third person sat still for data collection. We followed the same protocol of placing hands on lap, feet planted on the ground, and user staring ahead at a red dot placed on the wall for each of the data collection sessions. After the two minute segment was over, we saved

the data into a comma-separated values format (**CSV** file) in order to use this data for training.

In the future, we aim to finish configuring the hardware by training models based on users and perform the authentication process. The aim of this hardware setup is to allow users to use this physical device in order to authenticate themselves by training the CNN. In order to optimize the hardware we also will need to add a battery with the board as this will allow us to remove the cable wire that is attached to the container, as seen in **Figure 20**. The final step would be to load the final neural network model that will be used in order to authenticate a user into a system. After that, we will need to train and test the model until we have successful authentication.

Appendix A: IMF Decompositions of the BVP Signal

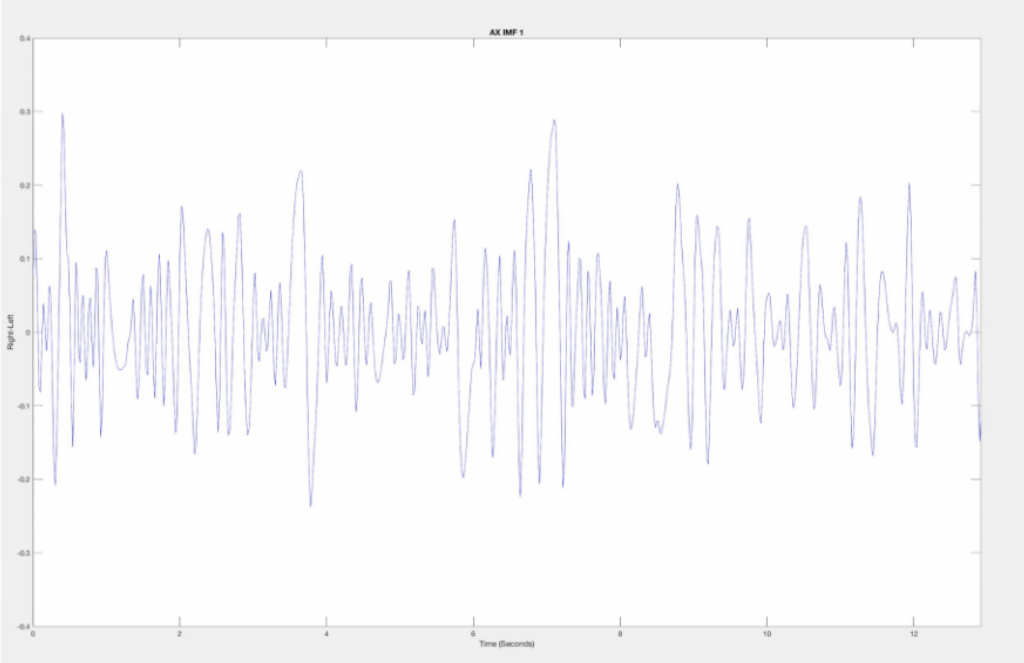

Figure 21: AX IMF 1

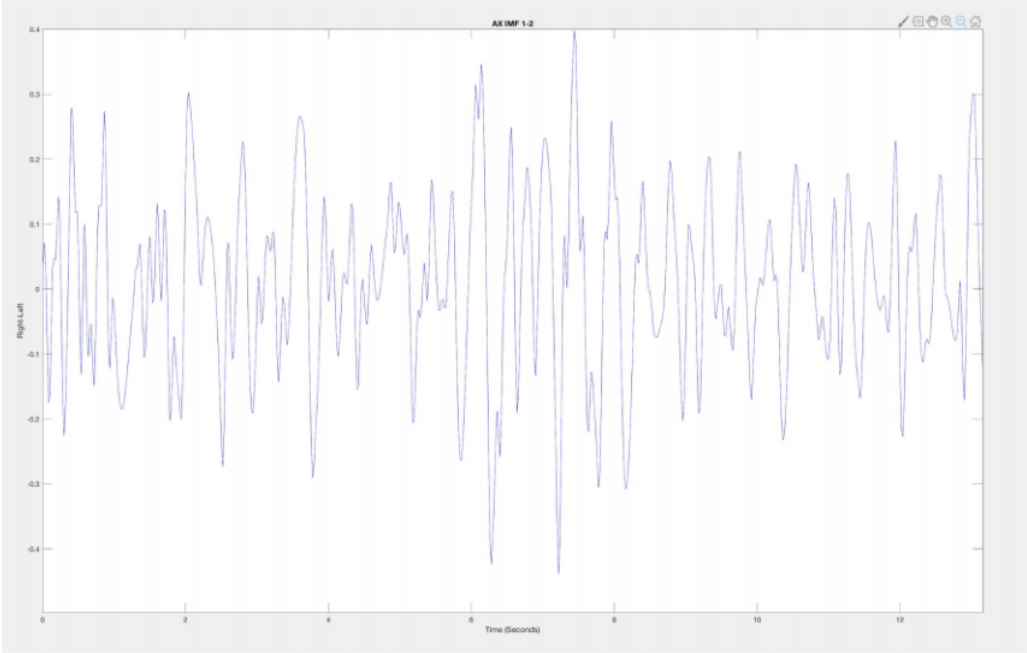

Figure 22: AX IMF 1-2

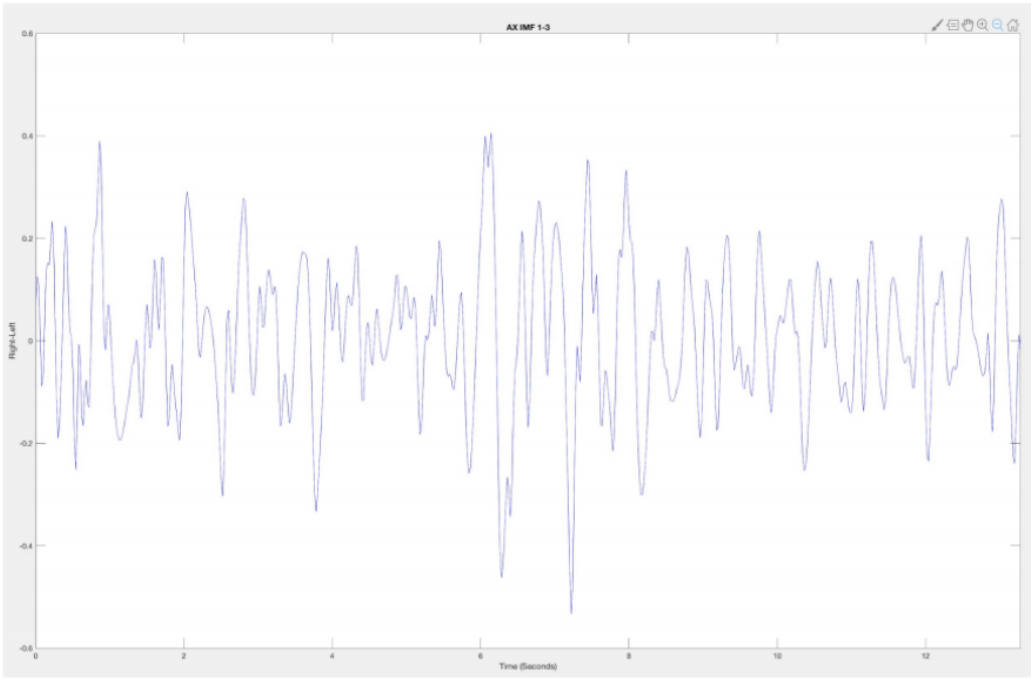

Figure 23: AX IMF 1-3

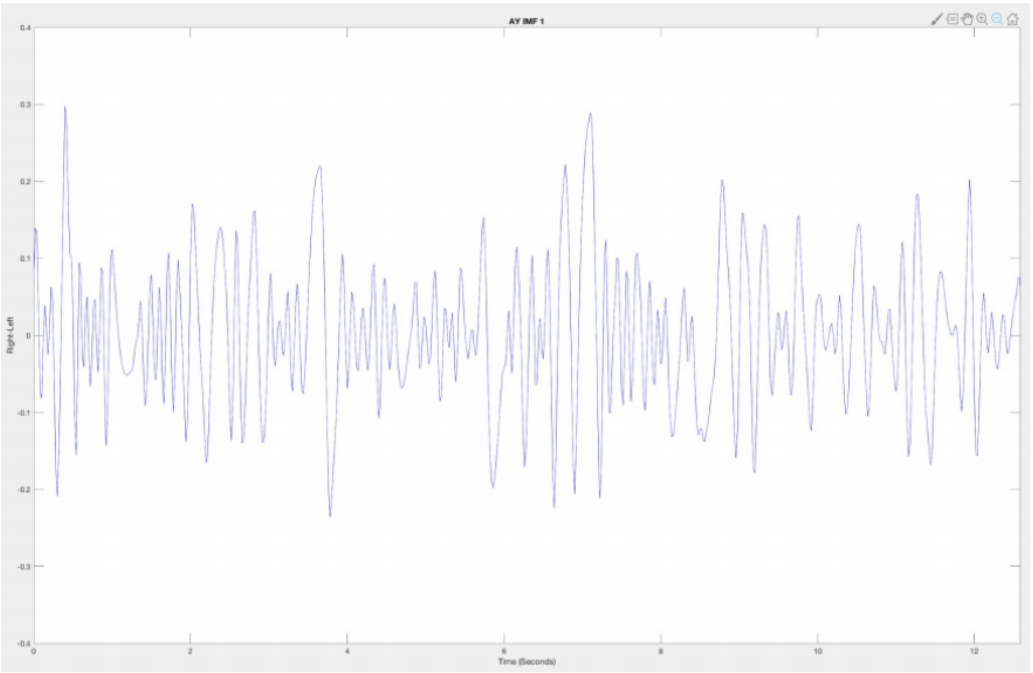

Figure 24: AY IMF 1

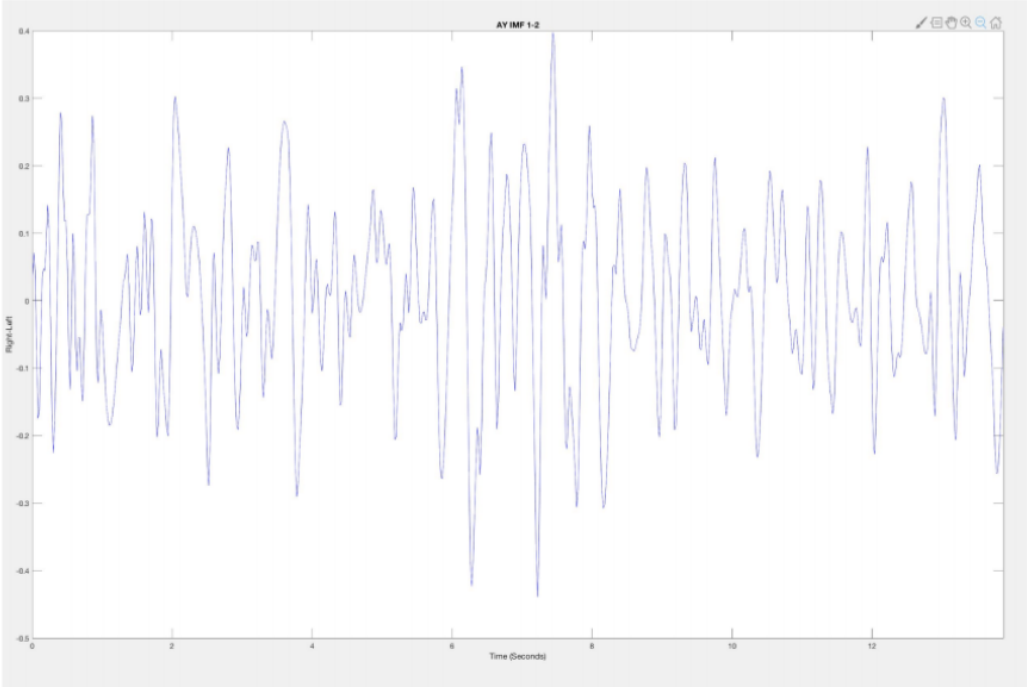

Figure 25: AY IMF 1-2

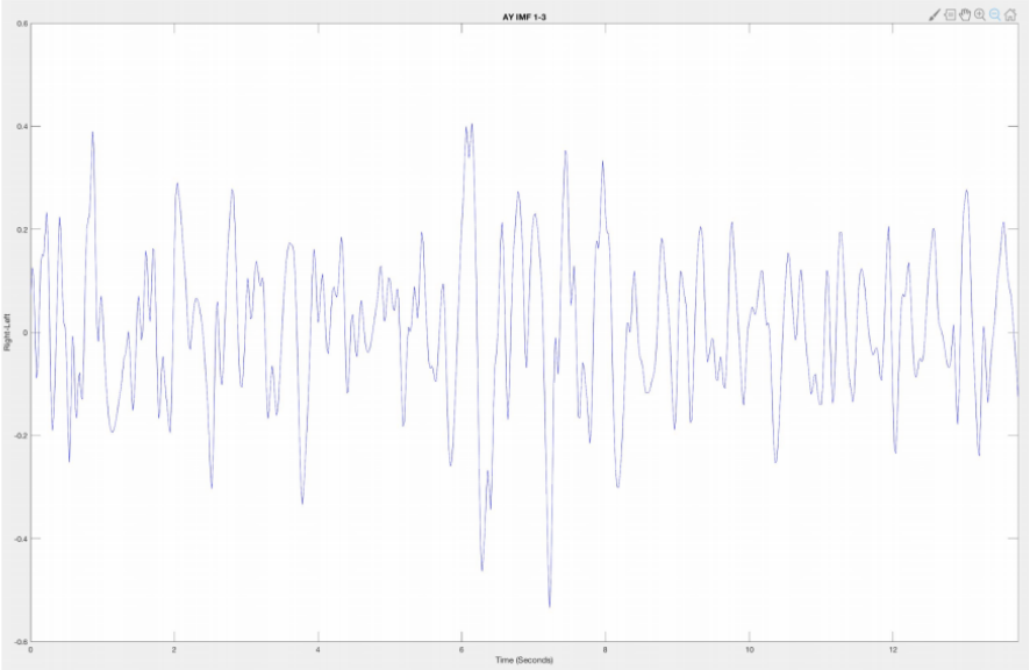

Figure 26: AY IMF 1-3

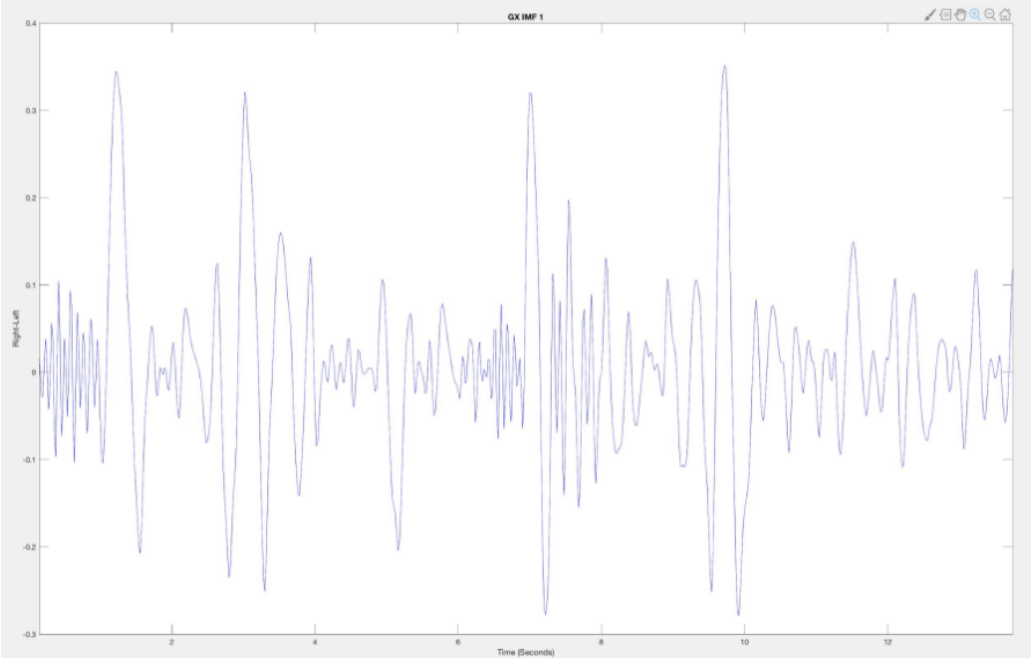

Figure 27: GX IMF 1

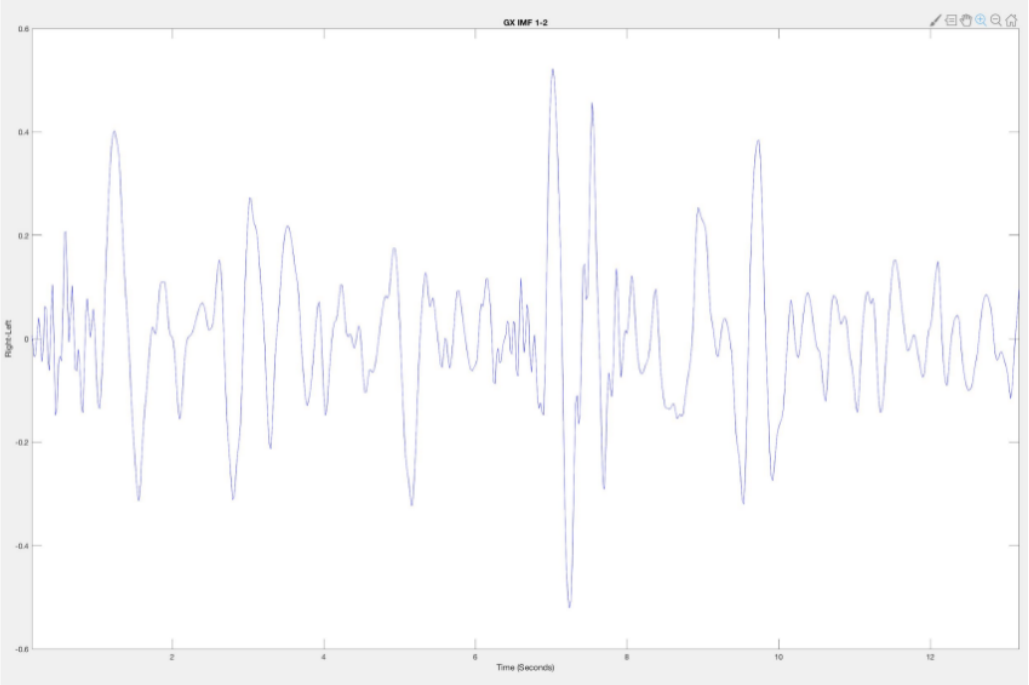

Figure 28: GX IMF 1-2

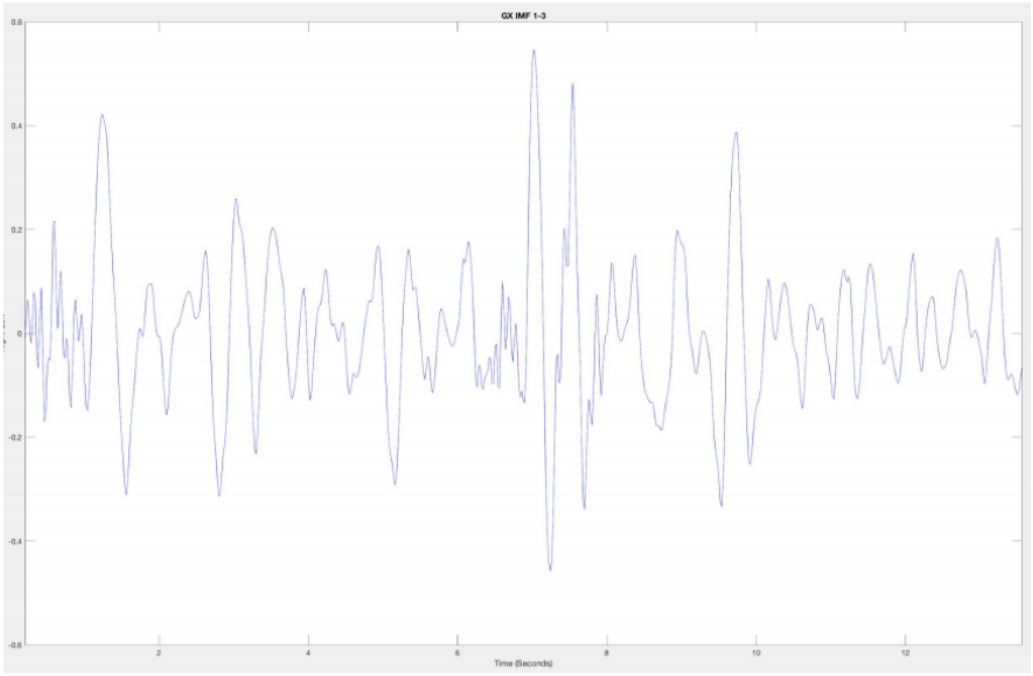

Figure 29: GX IMF 1-3

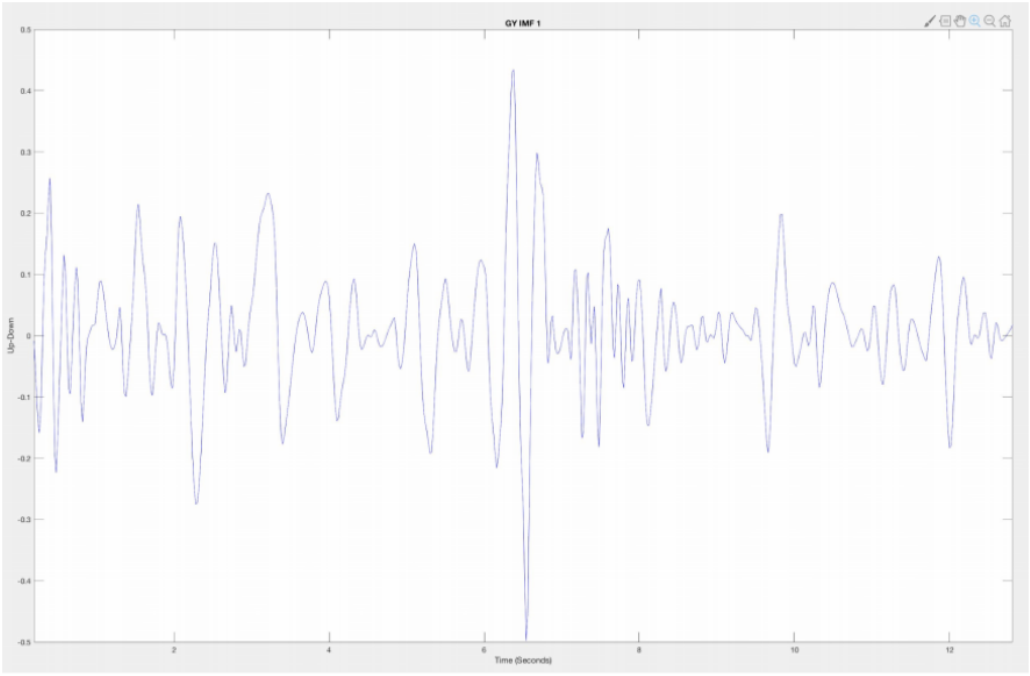

Figure 30: GY IMF 1

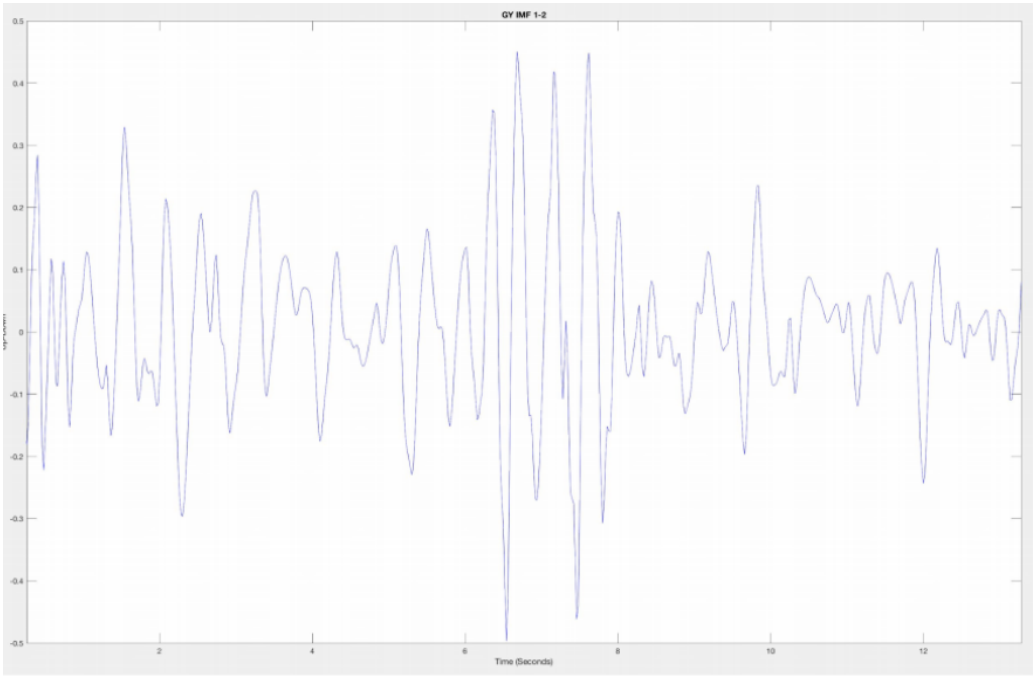

Figure 31: GY IMF 1-2

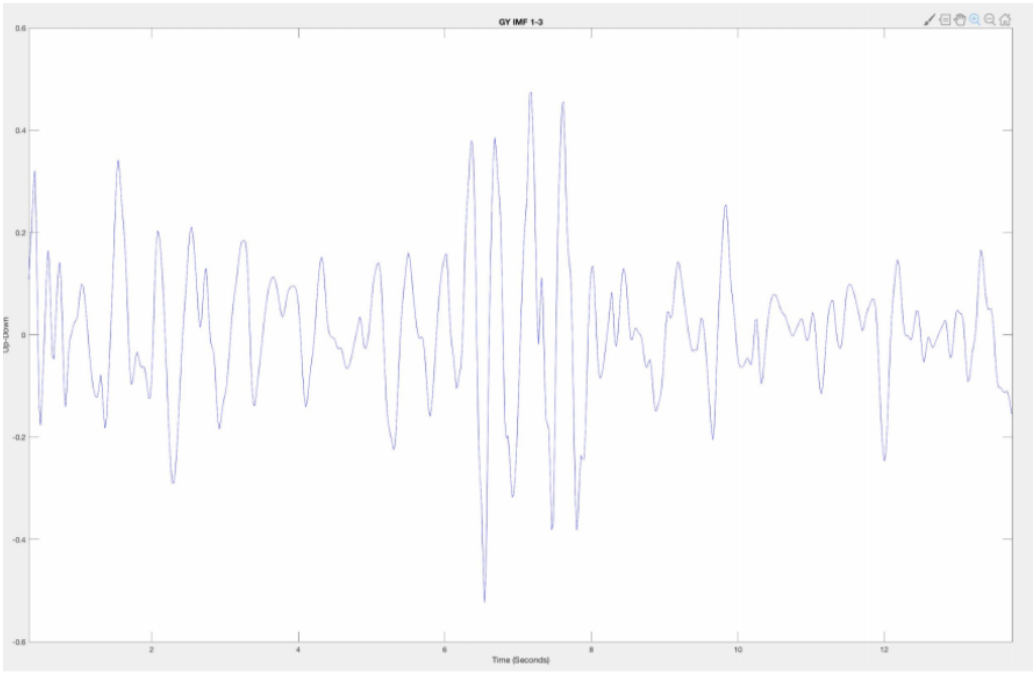

Figure 32: GY IMF 1-3

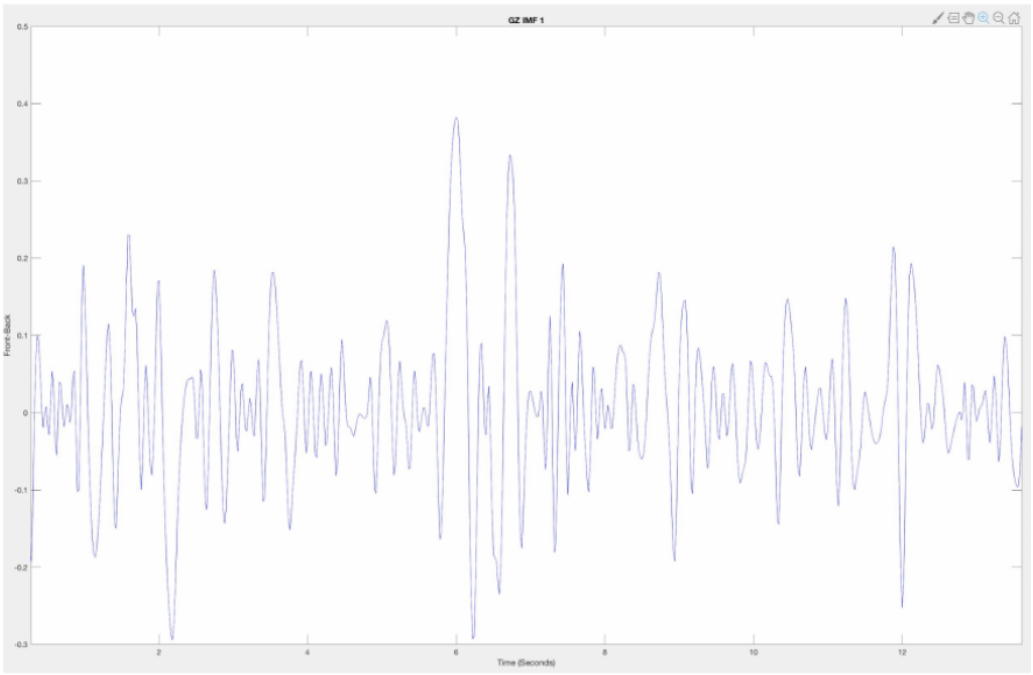

Figure 33: GZ IMF 1

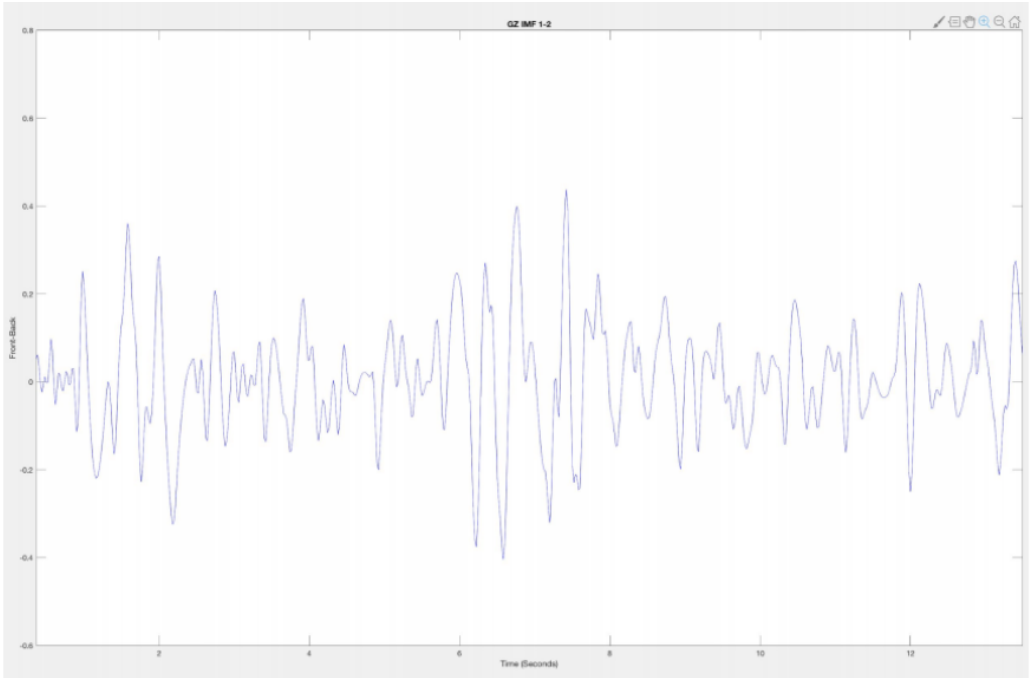

Figure 34: GZ IMF 1-2

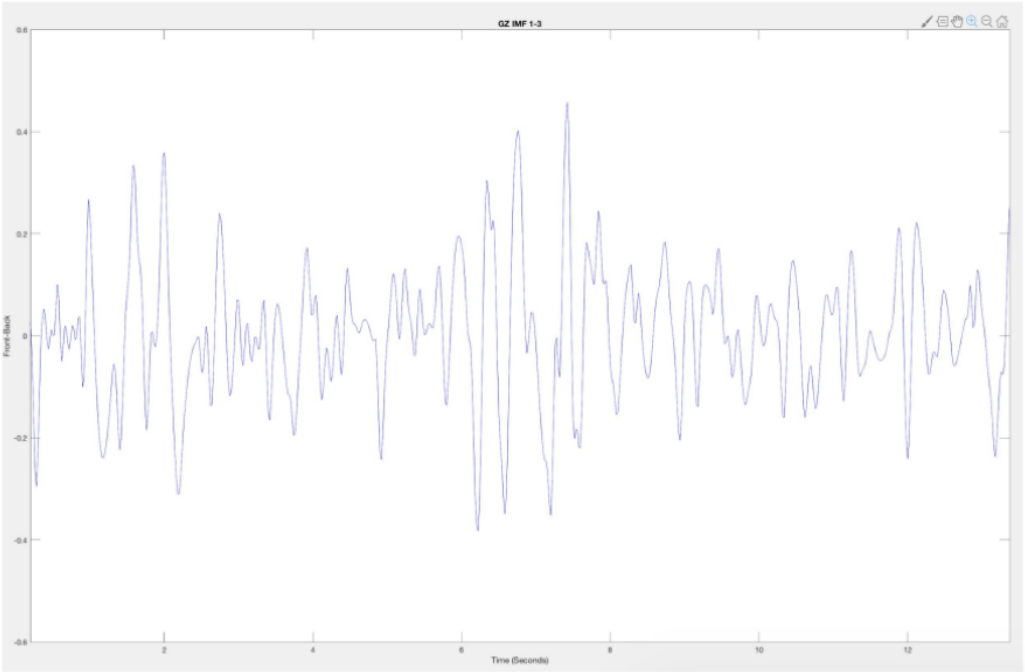

Figure 35: GZ IMF 1-3