Enhancing Stress Detection Systems Using Real-World Data and Deep Neural Networks

by

Tara Paranjpe

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2023 by the
Graduate Supervisory Committee:

Ming Zhao, Co-Chair
Nicole Roberts, Co-Chair
Nicholas Duran
Huan Liu

ARIZONA STATE UNIVERSITY

May 2023

ABSTRACT

As threats emerge and change, the life of a police officer continues to intensify. To better support police training curriculums and police cadets through this critical career juncture, this thesis proposes a state-of-the-art framework for stress detection using real-world data and deep neural networks. As an integral step of a larger study, this thesis investigates data processing techniques to handle the ambiguity of data collected in naturalistic contexts and leverages data structuring approaches to train deep neural networks. The analysis used data collected from 37 police training cadets in five different training cohorts at the Phoenix Police Regional Training Academy. The data was collected at different intervals during the cadets' rigorous six-month training course. In total, data were collected over 11 months from all the cohorts combined. All cadets were equipped with a Fitbit wearable device with a custom-built application to collect biometric data, including heart rate and self-reported stress levels. Throughout the data collection period, the cadets were asked to wear the Fitbit device and respond to stress level prompts to capture real-time responses.

To manage this naturalistic data, this thesis leveraged heart rate filtering algorithms, including Hampel, Median, Savitzky-Golay, and Wiener, to remove potentially noisy data. After data processing and noise removal, the heart rate data and corresponding stress level labels are processed into two different dataset sizes. The data is then fed into a Deep ECGNet (created by Prajod et al.), a simple Feed Forward network (created by Sim et al.), and a Multilayer Perceptron (MLP) network for binary classification. Experimental results show that the Feed Forward network achieves the highest accuracy (90.66%) for data from a single cohort, while the MLP model performs best on data across cohorts, achieving an 85.92% accuracy. These findings

suggest that stress detection is feasible on a variate set of real-world data using deep neural networks.

ACKNOWLEDGMENTS

I'd like to extend a huge thank you to Dr. Zhao, Dr. Roberts, Dr. Duran, and Dr. Liu for supporting me through my thesis and research endeavors. I'd also like to thank Sang-Hun Sim (M.S), Allen Lin, Aishwariya Ranjan, Kaiqi Zhao, the VISA lab, and the ASU Emotion lab. I couldn't have completed this work without their help.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

1.1 Motivation

Stress is a common, undeniable human experience which emerges when we respond to challenges or demands that overwhelm us. Though manifesting differently across individuals, stress is a shared experience impacting behavior, decision-making, social interactions, and health. According to the American Psychological Association [29], some common correlates of stress in adults include getting angry very quickly, unexpected mood swings, and screaming or yelling at a loved one. Stress and other intense physical and emotional states are generally associates with increased cardiological activation, such as heart rate [12, 23].

When investigated in a deeper context, stress is detrimental in workplace situations. According to The American Institute of Stress [33], 80% of workers feel stress on the job and nearly half of them requested stress management techniques. Police work, more specifically, yields overwhelming amounts of stress. The deeply unpredictable situations, coupled with high expectations from the public, have shown to increase feelings of stress, burnout, and mental health. Further, studies [21] have found correlations between health and years of service, observing that officers with more years of service had more evidence of mental health decline.

Such negative consequences of police work have impacted far more than just the officers. Families of the officers report increased stress and tension on family dynamics, noting that 30% of surveyed family members agreed that their spouse releases work

stress on the family [14]. Negative perceptions and repercussions of the job can also impact recruitment and retention rates in police academies and units. Police agencies face enormous costs when officers leave their jobs due to stress-related incidents, becoming responsible for covering long-term disability or early retirement costs [20].

This thesis aims to address the significant and relevant issue of stress among police cadets and the detrimental impact it can have on health, well-being, family, and job performance. The proposed stress detection framework leverages advanced technology, including wearable devices and robust deep neural networks, to detect instances of potential stress. The system uses data collected in realistic stressful situations to identify biometric patterns and provide early intervention strategies to prevent negative consequences of stress-controlled situations. By leveraging naturalistic data, we circumvent the induced nature of traditional data collection, ensuring that the stress experiences are authentic. However, accurate stress detection in naturalistic contexts is difficult to achieve. Very few studies have proposed effective solutions, attributing its challenges to the noisy, variate data and a lack of ground truth. Further, even fewer studies leverage deep neural networks for detection. This thesis explores data processing and filtering techniques and various deep neural network architectures to enhance stress detection systems and provide more accurate intervention steps.

1.2 Key Objectives

This thesis focuses on 2 key research objectives to design a robust stress detection system using real-world data and deep neural networks. The research goals are outlined below.

- What data processing techniques and methodologies create the most robust

and best-performing dataset for data collected in real-world contexts using a wearable device? Do those techniques address the challenges of unconstrained data collection?

- How do existing deep neural networks perform on real-world data for stress detection tasks? Are there specific data processing approaches that perform more optimally?

1.3 Methodology

This thesis uses data collected in real-world contexts. In order to gather the data, our lab (the Virtualized Infrastructures, Systems, and Applications Lab - VISA) and the The Emotion, Culture, & Psychophysiology Lab at Arizona State University collaborated closely with the Phoenix Regional Police Training Academy. We worked with 5 training cohorts, equipping consenting cadets with Fitbit Versa 3 devices. Each Fitbit device came with a custom-built Fitbit application. The application, detailed in Section 4, uses the Fitbit heart rate sensor to display 4 stress level buttons when the recorded heart rate exceeds a chosen threshold value (called a 'Stress Prompt'). Data collection periods for each cohort ranged from about 2-4 months. During the data collection period, cadets were instructed to wear their Fitbit watch both on-academy and off-academy hours. The Fitbit API was responsible for gathering biometric measurements, while the custom built application performed handshakes with the study's cloud instance to back up stress level responses. To manage and monitor each Fitbit device during the collection period, a web-interface dashboard was built.

Following the completion of data collection for each cohort, the raw heart rate data and stress level responses were extracted from the Fitbit API and the custom

cloud instance. The heart rate data were processed into 5-second granularities before being associated around stress level values to ensure accurate analysis. Various windows of heart rate were extracted to capture heart fluctuations prior to the stress prompt. Then, to understand the characteristics of the collected data, we evaluated the frequency of absolute change between heart beats and the range frequency over a 5-minute heart rate window. Through this analysis, we identified some outlier values, indicating potential noise. Such noise could pose a challenge to pattern detection in the deep learning models.

To address this potential challenge, we employed four different noise filtering techniques designed for heart rate artifact detection and smoothing: Hampel, Median, Savitzky-Golay, and Wiener. We then used two different structuring approaches the dataset: 1) contiguous samples of heart rate standalone and 2) features extracted from contiguous samples of heart rate. To further evaluate the effectiveness of the models, we divided the dataset into two groups: data from a single cohort and data from all cohorts combined.

These datasets were then fed into three deep neural network architectures, the Deep ECGNet, created by Prajod et al. [24], a Feed Forward network, created by Sim et al. [28], and a Multilayer Perceptron model. The data processing techniques and the feed forward neural network proposed by Sim et al. were leveraged as they were proposed and created to help establish baseline. Before evaluating their effectiveness, we carried out hyperparameter tuning to improve the performance of these models.

Results from the experiments on different techniques are compared across the three different deep neural network architectures. Results achieved on the same techniques and feed forward network as Sim et al. are compared to demonstrate the feasibility of stress detection on the large, multi-user dataset compared with the smaller subset of

data. The experimental results give us a better look at the temporal dependencies and features of the heart rate data.

1.3.1 Results and Contributions

In an effort to address the long and short-term impacts of stress in police officers and the overall law enforcement domain, this thesis contributes to a growing domain of stress management interventions. This work proposes a stress management system that uses data collected from police cadets at the local Phoenix Regional Police Training Academy on different neural networks. The proposed system suggests a preliminary approach to stress detection in police cadets using finely-tuned heart rate data and complementary self-reported stress scores collected by Fitbit wearable devices unconstrained and real-world contexts to highlight instances of potential stress through neural network architectures.

The most significant results from the experiment are as follows:

- Extracting samples of contiguous heart rate prior to stress level prompts is the most effective approach for capturing relevant trends. This approach outperforms the use of extracted heart rate features as input to deep neural networks.
- Hampel and Savitzky-Golay filters are the most successful at removing outliers in the heart rate data and preserving the characteristics of the data majority. Datasets with these filters applied perform higher on deep neural networks compared to other traditional filtering techniques.
- Smaller, cohort-specific datasets perform better on the deep neural networks than the full dataset with all participants. The Feed Forward network achieved the highest accuracy (90.66%) on a smaller dataset that was treated with the

Hampel filter. The MLP network achieved an 85.92% accuracy on the larger Hampel-filtered dataset. Both best-performing results leveraged the Approach 1 data structuring technique which used contigious heart rate samples prior to the stress prompt. These results suggest that deep neural networks may be able to generalize well.

1.4 Outline

The rest of the thesis is organized as follows: Section 2 provides background information about algorithms and technologies used; Section 3 investigates related works pertaining to real-world data collection, artifact removal, and deep neural networks for stress detection; Section 4 presents our *StressManager* framework for stress detection; Section 5 details our data processing strategies; Section 6 analyzes the collected data in terms of compliance and biometrics; Section 7 presents our approach to deep learning for stress detection; and Section 8 concludes the thesis.

Chapter 2

BACKGROUND

2.1 Wearable Devices as a Health Management Tool

Wearable devices are marketed toward the everyday, busy consumer. Affordable and lightweight, these devices allow users to stay connected via Bluetooth pairing to their smartphone, use common applications like timers or alarms, and view trends of their sleep, exercise, and other biometrics without disrupting their day. Their popularity is booming in recent years. According to Insider Intelligence [13], 23.3% of the US population owned a wearable device in 2021. This percentage is expected to increase to 25.5% in 2023 and nearly 27.2% in 2025.

With such a boom in this domain, its Internet of Things (IOT) features and capabilities have also grown substantially. Improvements to battery life, processing capabilities, and monitoring (such as blood oxygenation, sleep quality, crash detection, and more) have made the wearable device market more competitive and more attractive.

More specifically, wearable devices as a health management tool are on the rise. These tools are able to collect biometric data, such as heart rate, steps per day, movement intensity, and calories burned. With such raw metrics, these devices come with processing software to turn that data into trends over time to indicate overall health and quality of health. Additionally, some wearable devices, like the Apple Watch [27], allow users to send their data to their healthcare providers to increase health visibility.

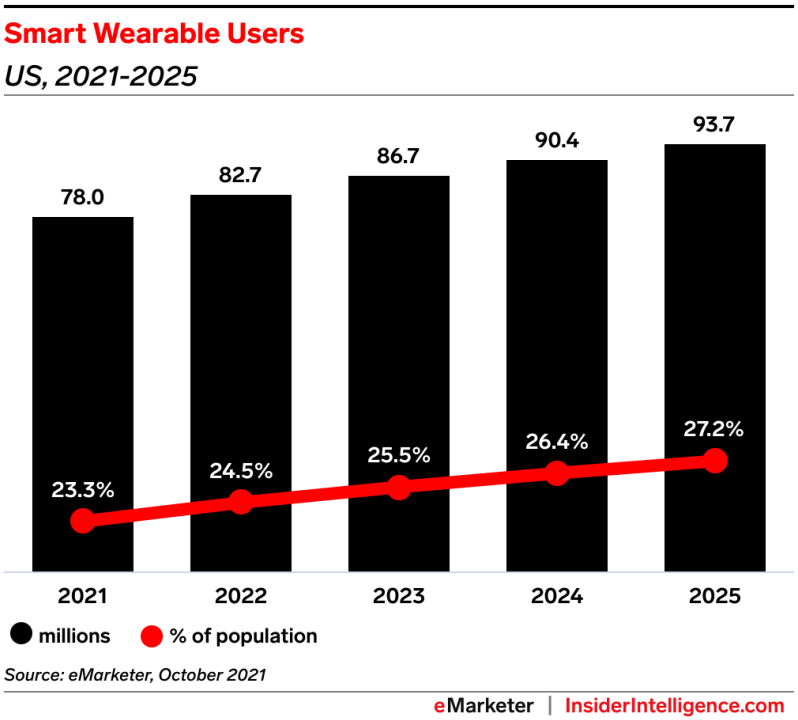

Figure 1. Figure Provided by Insider Intelligence [13] Visualizing the Projected Trend of Wearable Devices from 2021-2025

Further research suggests the use of data collected from these wearables to conduct predictive healthcare-related analysis and research. A study conducted by Beniczky et al. [3] investigated the use of wearable devices to detect and predict seizures for patients with epilepsy. The researchers collected electrocardiogram (ECG), heart rate variability (HRV), and accelerometer data from wearable devices. These data were then fed into machine learning models to create predictive models for generalized tonic-clonic seizures. A similar study, conducted by Rykov et al. [26], leveraged wearable devices in the same capacity to screen for depression-related biomarkers. The researchers collected sleep patterns, physical activity, and psychological measurements from wearable devices to evaluate mental states of their study participants. There is precedent in using wearable devices to manage health through machine learning systems.

2.2 Validity of Wearable Device Data

As the monitoring capabilities of wearable devices increase, evaluating the validity of the collected data becomes important. Several studies compare commercially available devices and their performance in terms of accuracy of the metrics collected.

A study by Bai et al. [2] runs tests on wearable devices, including Fitbit, Garmin, and Apple Watch. The authors test the quality of data collection across these three devices in a "free-living setting" compared to traditional sensors for steps, heart rate, and moderate-to-vigorous minutes (MVPA). Over the 24-hour data collection period, the authors find that all three devices measure steps accurately and both the Fitbit and Apple devices provide reasonable heart rate calculations. The authors also find that all three underestimate the MVPA metric.

Cumulative research shows that no watch is particularly better than the other. In this competitive space, new hardware and software capabilities are bound to improve performance and increase accuracy in biometric data collection.

2.3 Machine Learning Fundamentals

Machine learning is an integral part of daily life. From recommendations from streaming services to spam detection of emails, machine learning is responsible for daily functions beyond what we can see. These systems learn from input data and its features. Over time and with more relevant data, the model learns patterns and adjusts its parameters to become more accurate. Models are "trained" on training data, which consists of the data and their classification, and are evaluated on testing

data, which omits the correct classification and measures the model's success based on its prediction compared to its expected label.

There are roughly four integral pieces in a machine learning pipeline [1]: 1) data acquisition, 2) data processing, 3) model development and training, and 4) model evaluation. Each phase of this pipeline is dependent on the domain of the problem, but often requires fine-tuning and adjustment to improve the overall accuracy and robustness of the model.

2.4 Data Acquisition

The first piece in the machine learning pipeline is data acquisition. This phase focuses on collecting data and labels that will eventually contribute to the training of the machine learning model. This phase is important and needs to be done properly. This means that the data collected should be relevant, not missing or repeating values, has a strong representation of each class, and has accurate labels. In most cases, it is better to collect a wide variety of data points that are relevant rather than a few cases to help complement the data and enable the model to learn on wider patterns.

2.5 Data Processing

The next phase of the pipeline is data processing. There are three key pieces to this stage: 1) detecting and handling data outliers, 2) transformation of raw data into data that is more usable by machine learning models, and 3) removal of non-essential noise and redundant data from the dataset.

2.5.1 Outlier Detection

In real-world data collection, noise and outliers in data are expected. With wearable devices still being improved for extraneous noise detection, improper wrist placement, tattooed skin, and sudden movements often introduce noisy data that are not consistent with actual physiology. It becomes more challenging when researchers do not have the exact context behind when the data was collected. Machine learning models can struggle to learn patterns with noisy data and empirically perform better with invariant, pure data [11].

Noise detection itself is, however, a difficult task. There are different statistical approaches that can be used to identify specific instances of noise. In the case of heart rate, which is calculated in beats per minute, heart rate is highly fluctuating. According to a study by Falcone et al. [7], the heart rate of a normal, healthy adult can change nearly 75 bpm in 1 minute of exercise. Moreover, heart rates varies between adults depending on age and overall health, proving more so that heart rate anomaly detection is not always constrained by min-max heart rate values. To mitigate this issue, there are several outlier detection filters that are designed to detect anomalies and smooth them over.

The **Hampel Filter** is designed for outlier detection in time series data [22]. This algorithm leverages a configurable sliding window, as seen in Figure 2.

X_s represents the observation value. $k = \frac{len_{window}-1}{2} + 1$ indicates the number of samples before the observation and after the observation in the window.

For each window, the median, median absolute deviation (mad), and standard deviation are calculated. The median is calculated traditionally (m_{window}). The mad

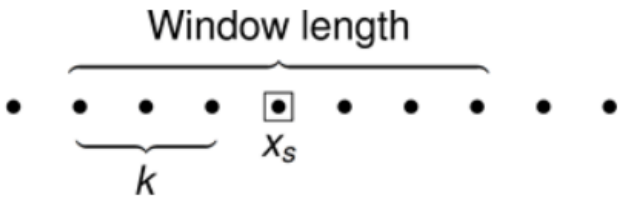

Figure 2. Visualization of the Hampel Filtering Window [10]

is calculated as the median of all the absolute values of the observation minus the median of the complete window.

j is the Gaussian coefficient and the standard deviation is calculated as follows:

$$\sigma = j * mad_{window}$$

Each value is then analyzed. 3 is used as the default threshold:

$$|X_s - m_{window}| > 3 * \sigma$$

If the condition is satisfied, the conditions for outlier have been met, and the value is replaced by the median over the window, m_{window}.

The **Median Filter** is a common algorithm used in digital image processing to preserve edges of images and remove noise. This filter uses a configurable window, similar to the Hampel Filter above.

An example of the algorithm is shown below[19]:

Assume input vector is as follows: X = (1, 2, 3, 6, 10), window size = 3.

1. $y_1 = median(1,\ 2,\ 3) = 2$

   $y_{1result} = (2)$
2. $y_2 = median(2,\ 3,\ 6) = 3$

   $y_{2result} = (2,\ 3)$
3. $y_3 = median(3,\ 6,\ 10) = 6$

   $y_{3result} = (2,\ 3,\ 6)$

4. $y_{result} = (2, 3, 6)$

The **Savitzky-Golay Filter** is built to smooth sequenced data. For each value in the input sequence, the algorithm takes its N neighbors and tries to fit a polynomial to the data. Each polynomial is set to the same degree, but is fit to different values in the window. To ensure the polynomials don't fall into Runge's phenomenon, large oscillations in the data that don't match the integrity of the data, the Savitzky-Golay filter finds the lower order polynomial that fits the sequence in terms of least-squares [25]. Figure 3 shows how the filter works on the sequential data.

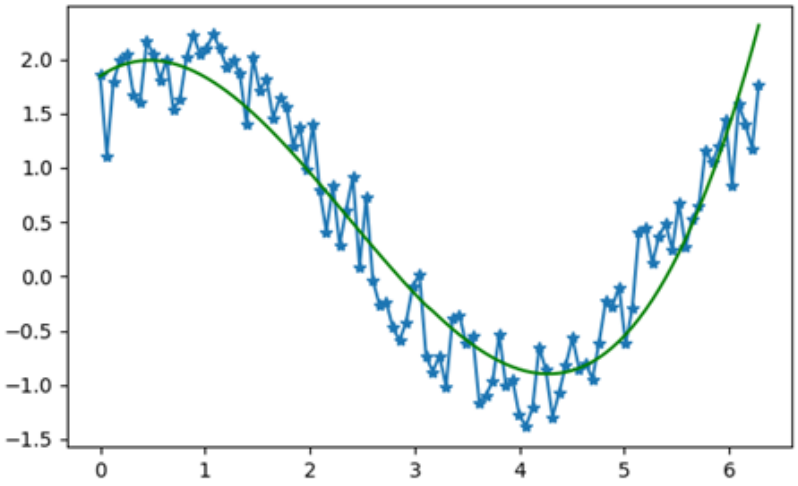

Figure 3. Example of Savitzky-Golay Filtering - the Blue Line Indicates the Original Data, the Green Line Indicates the Data Smoothed by the Filter [25]

The **Wiener Filter** is often used to enhance signals from data. The main goal of this filter is to minimize the mean square error and the average squared distance between the output and the desired signal. It depends on the power of degree of the signal and the noise.

2.5.2 Data Transformation

Another phase of data processing is transforming the data so it is understandable by the machine learning model. Models learn by pulling patterns from input data and correlating them to their given label. It is important that the input data is normalized in a way that enables pattern identification.

Firstly, data contiguity is important. That means that the number of data points must be consistent across individual data samples. For example, if a model is given 10 data points to indicate a specific label, it will be trained to find patterns across the next input of 10 data points. This also means in the case of time series data, the variability across data points is consistent - i.e. 1 minute variability from point to point.

Another common technique used by machine learning scientists is data normalization. These include min-max normalization, z-score normalization, and decimal scaling, among many others. The goal of normalization is to maintain relationships between the data points and improve generalization. If there are values at different scales, the model may become biased toward a specific value.

Min-max normalization is very common [17]. All features in a given vector are scaled to values between 0 and 1. The observation value, $Xobs$, in a given vector, X, is scaled using the following formula:

$$X_{normalized} = \frac{X_{obs} - X_{min}}{X_{max} - X_{min}}$$

where $Xmin$ and $Xmax$ represent the min and max values from the input vector.

Z-score normalization scales the input vector so the vector has a mean of 0 and

a standard deviation of 1. Each value in the input vector is scaled using the following formula:

$$X_{normalized} = \frac{X_{obs} - \mu}{\sigma}$$

where μ represents the mean of the input vector, while σ represents the standard deviation of the data.

Decimal scaling scales the data from an input vector by dividing values by a power of 10. Each value in the input vector is scaled using the following formula:

$$X_{normalized} = \frac{X_{obs}}{10^j}$$

where j represents a configurable exponent.

2.5.3 Data Redundancy

The final stage of data processing is removing non-essential and redundant data. This step reduces the complexity of the models and ensures that models learn on a wide variety of relevant data. Removing redundancy in data can also ensure that the models don't learn with bias. In order to avoid training bias, it is important to make sure that the number of samples representing each class label in the dataset is balanced. If unbalanced, the model will see more data leaning toward the majority class and tend to predict closer to the majority class.

In real-world situations, however, balanced data is often not feasible. To combat that, scientists have come up with re-sampling techniques: undersampling and oversampling (see Figure 4).

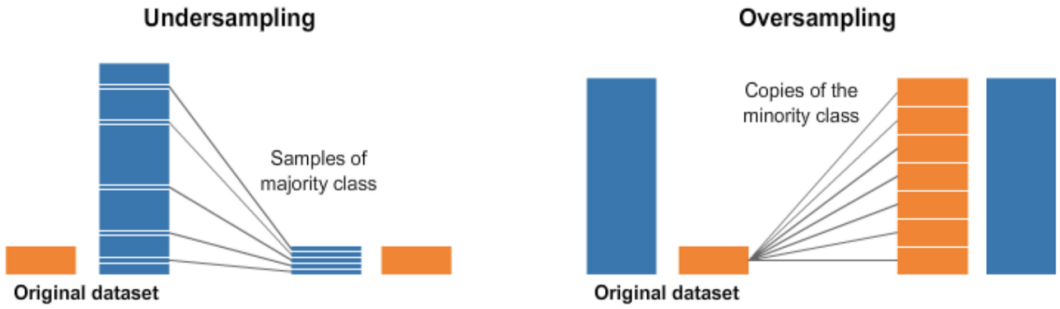

Figure 4. Visualization of Undersampling and Oversampling Techniques [6]

Undersampling is a technique that balances datasets by keeping data from the minority class, and reducing samples from other classes, including the majority class, to balance. This approach will preserve features of the minority class which are harder to predict since they are less frequent. However, this approach will also shrink the overall dataset and could lead to loss of information.

Oversampling, on the other hand, balances datasets by creating synthetic data learned from the minority class to increase the size of the minority class to match the majority class. Some common over-sampling algorithms include SMOTE: Synthetic Minority Oversampling Technique, which is integrated into Python for easy re-sampling.

2.6 Model Development and Training

Deep neural networks typically have an intricate architecture with at least two layers. These models are typically more complex than classification models and are built for multi-faceted data. Within deep neural networks are convolutional neural

Layer Type	Function of the Layer
Convolutional	This layer generates a feature map from an image by sliding a filter to detect patterns.
Activation	This layer defines how the input's weighted sum is transformed.
Pooling	This layer is responsible for down-sampling the feature map which reduces the problem of overfitting.
Batch Normalization	This layer normalizes the data to fix the mean and variances of the data.
Dropout	This layer is responsible for randomly setting units to zero during training to prevent the problem of overfitting.
LSTM	This stands for "Long Short-Term Memory Network". It is an RNN layer that recognizes patterns in time series or sequence data.

Table 1. Common Layers in a Deep Neural Network and Their Function

networks (CNNs) that are commonly used in image classification tasks, artificial neural networks (ANNs) used for classification or clustering, and recurrent neural networks (RNNs) for time series prediction. Models are usually developed with different layer types based on the task at hand. Table 1 shows some common layers and their function.

Once the architecture of the model is created, the next phase is training the model. The conventional data split is 80% for training and 20% for testing. The training data is what will tune the weights in the model, while the testing data is a set of data that the model will not have seen before. The testing data plays a key part in the evaluation phase described in the following subsection.

Models are trained with finely tuned hyperparameters. These include the optimizer function, the loss function, the learning rate, batch size, and number of epochs. It is very important to test and experiment with these hyperparameters, as they can impact how the model's weights and training processes are updated. These parameters specify how the model learns from the training data. Table 2 defines some key hyperparameters and their function and importance.

Hyperparameter	Function of Hyperparameter
Optimizer Function	The optimizer function is an algorithm that defines how the attributes of the neural nework should be updated. Some common optimizers are Adam or SGD (stochastic gradient descent).
Loss Function	This hyperparameter is a function that compares how different the target value of the data is from the predicted value from the model. Some loss functions include the binary cross entropy function and the mean squared error.
Learning Rate	The learning rate defines the rate at which the network updates its parameters. Usually a low learning rate means slower learning but smoother convergence, while a high learning rate is faster for learning but doesn't always converge.
Batch Size	Batch size is how many samples are processed before the model updates its parameters.
Epochs	This number specifies how many times the training data is fed to the model in the training stage.

Table 2. Hyperparameters and Their Function

2.7 Model Evaluation

Once the model is trained, the next step is to separate the testing dataset from the main dataset and feed the values through the model. The testing dataset is the best way to evaluate the model since these are values that the model has not seen before. The output from the model is compared to the expected label to evaluate model success. Typically, machine learning engineers use accuracy as a measurement of model success. Accuracy is usually calculated as follows:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Number\ of\ Total\ Predictions\ Made}$$

Accuracy, however, is not always a helpful metric to determine model performance. Accuracy can be biased to the majority labels in a given class; higher occurrence labels may be more correctly predicted while the lower occurrence labels are misclassified and ignored. To correct this bias, engineers also leverage precision, recall, F1-Score, and ROC curve.

Precision and recall are calculated using values found in a confusion matrix (see

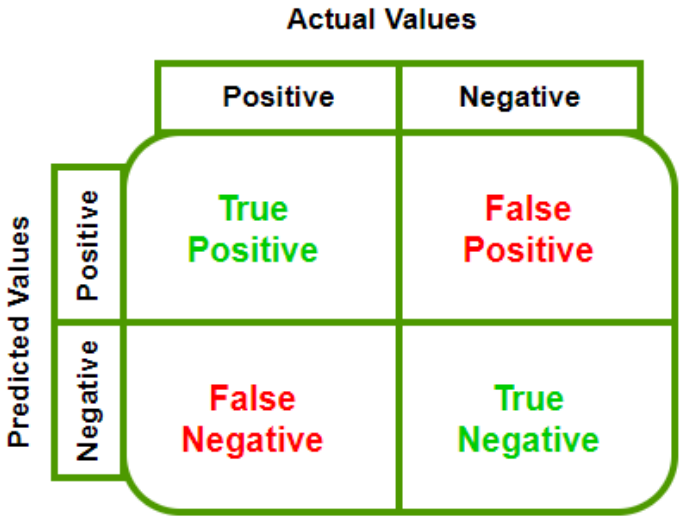

Figure 5. Visual Representation of a Confusion Matrix for Model Evaluation [32]

Figure 5) for binary classification tasks. These values describe the relationship between the predicted values and the actual values [4].

- True Positive (TP): Positive classes correctly predicted as positive
- True Negative (TN): Negative classes correctly predicted as negative
- False Negative (FN): Positive classes incorrectly predicted as negative
- False Positive (FP): Negative classes incorrectly predicted as positive

Precision tells us the percentage of correct predictions for a given class given all predictions for that class. In other words, what percentage of identifications for the positive class were actually right?

$$Precision = \frac{TP}{TP + FP}$$

Recall tells us the percentage of correct predictions compared to the total number of occurrences of the class. In other words, what percentage of actually positive labels were identified correctly?

$$Recall = \frac{TP}{TP + FN}$$

The F1-score is a combination of the precision and recall, and is calculated as follows:

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

Chapter 3

RELATED WORKS

3.1 Data Collection in Real-World Contexts

Research in real-life contexts is rather rare. Simply put, as researchers, we do not have much context when we gather data in daily-life contexts and rely on self-reported information as ground truth. Larradet et al. [15] researches the importance, advantages, and disadvantages of data collection in the wild. The authors emphasize that data collected in laboratory settings are often done through highly controlled induction techniques, which don't always induce the desired emotion or feeling. They state that some advantages of real-world data collection include maintaining ethical constraints (not having to induce a feeling or experience in someone) and natural context awareness. However, the authors also underscore the challenges of such data collection, stating that the absence of ground truth, introduction of noisy data without context, and necessity for long-term experiments limit the overall data collection process. Real-world data collection is difficult to implement and execute.

A handful of studies focus on stress management *in situ*, collecting data from participants without any intervention using wearable devices. Wearable devices, as highlighted in prior sections, tend to capture data trends, but also integrate noisy data collected from daily wear and movement. Martinez et al. [18] does an investigation into the efficacy of the HRV metric on stress in the real-world. The authors note that in laboratory settings, where the stress is induced in an isolated environment, HRV alone is a reliable metric for stress detection. They note that its reliability comes from

the specific stress events and research-grade sensors. The authors deploy a long-term study where they collect HRV in daily life contexts of different participants. Through their analysis, they find that HRV and stress have a small relationship in the data collected *in situ*. They conclude that while HRV is enough to deduce experiences of stress in controlled laboratory settings, HRV collected in real-world contexts is not enough for stress prediction and should be considered a piece of a more complex phenomenon. This tells us the importance of real-world data collection and the difficulty associated with phenomenon conclusions.

The ability to make diagnoses and predictions like these in real-world contexts rather than typical evaluations in controlled environments demonstrate the feasibility of using wearable devices to extend clinical research. It becomes clear that wearable devices have capabilities that far surpass traditional data collection techniques, allowing for data collection in broader, naturalistic, and more extensive contexts.

3.2 Artifact Removal in Wearable Device Data

As mentioned, real-time data collections inherently comes with noisy data or noise artifacts that misrepresent the true values of the data collected. Noisy data is common in any type of data collection, but becomes essential to manage when wearable devices are used.

Veeravalli et al. [31] proposes a real-time heart rate monitoring system for patients that can identify normal heart fluctuations and call out abnormal or noisy heart rate artifacts. In this paper, the authors use filtering techniques to weed out any abnormal or high-frequency ECG data in real-time current beat analysis, specifically using the Savitzky-Golay filter which simultaneously preserves high order characteristics

corresponding to the ECG peaks. To capture anomalies in uni-variate data, the authors highlight the benefits of the Hampel filter, which proves to be most effective and robust. The research also leverages a median filter to smooth the data, which is also a linear-time complexity algorithm. All of these algorithms present a low complexity, lightweight solution for real-time detection. The authors find high accuracy results for their detection algorithm, contributing the reduced complexity and high accuracy to filtering techniques to remove abnormalities.

Another paper by Zhang et al. [35] proposes a motion artifact detection experiment across supervised and unsupervised models integrating accelerometer data. The authors leverage data collected in laboratory settings and data collected in real-world settings and use 5 different supervised models and 3 unsupervised models. The models are able to determine abnormalities in ECG data with 94.1% accuracy, but the authors find that complementing ECG data with accelerometer data does not improve classification performance and in fact drops it substantially. They find that detecting abnormalities in real-world data is still difficult, but can be done with well-trained models and extracted features.

Both these papers show that artifact detection from wearable device data is essential, improves overall accuracy compared to non-filtered data, and is simultaneously difficult. The second study investigates using other metrics to complement the heart rate data but doesn't get promising results. Several papers emphasize the importance of some type of filtering to smooth data abnormalities.

3.3 Stress Detection using Deep Neural Networks

Stress detection using deep neural networks is a growing area of research. Over time, different studies have worked with different channels of data to complement stressful situations and different deep neural network architectures to capture relationships with that data.

A recent study by Li et al. [16] proposes stress detection using various physiological measurements calculated from traditional sensors on a 1-dimensional convolutional network and a multilayer perceptron (MLP) neural network. The data were collected from participants performing specific tasks, creating a binary distinction between stress and non-stressed states. The datasets were processed into training and testing using subject-independent cross-validation. Their binary approach on the deep convolutional neural network reached 99.80% accuracy, while the MLP model reached 99.65% for the same classification task.

Prajod et al. [24] proposes a new deep neural network architecture, called the *Deep ECGNet*, to predict stress using publicly available datasets, WESAD and SWELL. The *Deep ECGNet* is a CNN-LSTM model that uses the CNN layer to extract features and the LSTM layer to learn the patterns from the extracted features. They used the ECG data from both datasets as input to the model. For the ECG data, the authors leveraged a second-order Butterworth band-pass filter and Min-Max normalization. Similar to the previous paper, the datasets were processed into training and testing using subject-independent cross-validation. The model achieved 90.8% accuracy on the WESAD dataset and 75.5% on the SWELL dataset using the leave-one-subject-out (LOSO) approach.

More recently, Sim et al. [28] proposed the use of simple classification models and

a simple feed forward neural network on data collected from police training cadets in real-world contexts. The Sim et al. study is a preliminary phase of the larger work conducted in this thesis. The authors take heart rate data, presented in beats per minute, and extract features including the mean, standard deviation, minimum, maximum, and other resting heart rate specific features. The participants in the study provide self-reported stress level scores to complement instances of possible stress. The authors then feed the features into a simple feed forward network, designed with six hidden layers activated by the ReLu function. They achieve 95.98% accuracy using the heart rate features on the feed forward network.

Numerous studies have been done on stress detection using deep neural networks using different models and channels of data. The results achieved by these studies demonstrate the potential of stress detection using deep learning. However, very few studies leverage deep learning on data collected in real-world contexts. Such research can expand the models' ability to generalize to different populations and contexts, proving important in stress management across the board. This thesis proposes a unique approach by investigating data taken in real-world contexts, processing the data for potentially noisy artifacts, and leveraging deep neural networks to learn from the naturalistic data to generalize trends between heart rate and stress.

Chapter 4

STRESSMANAGER: DATA COLLECTION FRAMEWORK

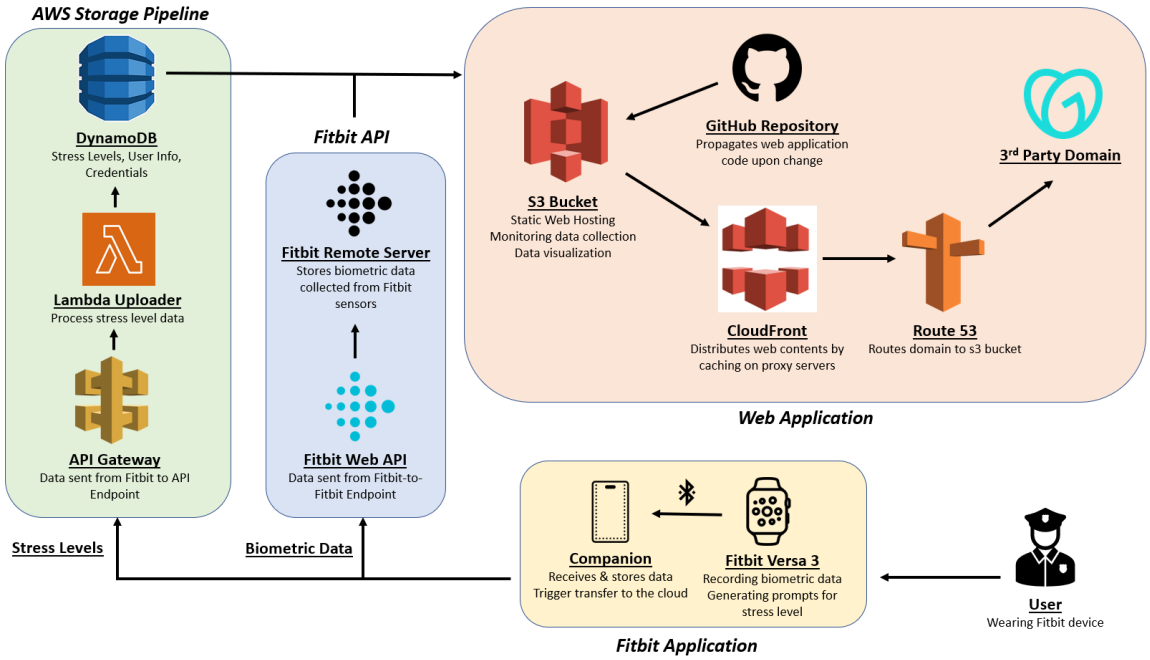

Figure 6. *StressManager* Architecture showing the Fitbit Application, Storage, and Web Application components

The data collection framework, called *StressManager*, is a study initiative led by two interdisciplinary teams focusing on machine learning and stress psychology. The goal of *StressManager* is to design and gather biometric and self-reported data in daily, unconstrained contexts of police training cadets. This study was conducted over the course of 11 months at the local Phoenix Regional Police Training Academy. The study alternates between various cohorts with different cadets and varying forms of intensity based on their training interval. Participants in the study are each equipped with a Fitbit Versa 3 device and a companion mobile application component, forming the larger Fitbit architecture. The participants are asked to wear their device both on and

off academy hours, while sleeping, and on weekends. Each device has a custom-built Fitbit application that is responsible for gathering all relevant information about the user while worn. The data is then propagated to relevant database storage buckets, where they are then retrieved and visualized through the custom-built web application. This section describes the key elements of the data collection framework, *StressManager*, including the services and software designs behind the study. Figure 6 shows the architecture diagram of the *StressManager* framework.

4.1 Participants

The participants of the study are training cadets at the Phoenix Police Regional Training Academy. The cadets range between the age of 20-50, originating from various backgrounds, prior experiences, and demographics. The batch of cadets, called cohorts, undergo rigorous training as part of their initiation into the police force. Their training program lasts about 26 weeks and consists of varying periods of intensity: orientation, lecture work, field training, and physical conditioning. As part of this study, we worked with 5 unique cohorts to gather insights and data during different phases of their training. Table 3 shows the number of consenting participants and duration of data collection for each cohort.

Cohort	# of Participants	# of Males	# of Females	Weeks of Collection
Original Cohort	15	12	3	13
Cohort C	10	8	2	23
Cohort D	4	3	1	12
Cohort E	6	6	0	18
Cohort F	2	2	0	13

Table 3. Information about Participating Cohorts

4.1.1 Participant Responsibilities

The participants are asked to participate in personal study components. Using their Fitbits, the cadets are asked to individually rank their stress, when prompted, on a set scale: 1 (Not Stressed), 2 (A Bit Stressed), 3 (Moderately Stressed), 4 (A Lot of Stress), 5 (Extremely Stressed). Over time and progression of the study, this scale was reduced from 1-4 to increase clarity due to feedback from the cadets. This scale is individual to the user, and the study emphasizes that there is no right or wrong answer. Psychologically speaking, this range will be interpreted differently for each cadet as some people can be naturally more reactive to stress than others. The cadets are also asked to wear their Fitbit throughout the day and night in order to collect biometric data in a naturalistic context, but are encouraged to take "wrist breaks" to charge their devices. Daily surveys are also administered to each cadet. This survey consists of about 4 questions with a few subsections and focuses on gauging emotional and decision-making perception scores. Occasionally, consenting participants were asked to wear their Fitbit as they endure training drills and scenarios as part of their academy curriculum.

4.1.2 Participant Consent and Anonymity

This study, approved by the Institutional Review Board (IRB), allows our team to consent willing and able participants to our research. Each cohort went through an in-depth introduction to the study, what it would entail for them, and how it would help them address their own stress. Participation was strictly voluntary, and any participant, at any point, could halt participation.

Figure 7. Default and Prompted Face of the *StressManager* Clock Face Application

Due to the confidential and sensitive nature of the data collected, the identity of each participant is strictly and carefully protected. Each cohort of focus was assigned a unique letter to help distinguish between collected data. Each individual cadet within each cohort was also assigned a unique, unidentifiable username. Personal data, like height, weight, age, and gender were gathered and entered into the Fitbit application for each individual device to ensure the data collected was accurate to the user's physiology.

Each user, upon entering into the study, was required to sign a consent form indicating desire to particpate and an equipment loaner form for the study administrators to manage. The consent form details the extent of participation, commonly asked questions, and above all, guarantees that the data collected will not be misused or associated with the specific cadet.

4.2 Fitbit Application

The Fitbit interface and device are simple and easy to use, securely store data for easy retrieval from the Fitbit API, and connect to a Fitbit Developer [8] interface for custom-built applications with a readily-available developer support community. Fitbit Developer provides a robust pipeline for creating Fitbit applications and deploying them onto any consenting device. Additionally, the developer toolkit provides simple queries to the Fitbit API, allowing developers to store and retrieve Fitbit data and provides read and write methods for data retrieval.

Our team created a StressManager Fitbit application, developed with JavaScript, HTML, and CSS constructs to gather and monitor heart rate with a dynamic and scaling threshold, determined by the Fitbit-calculated resting heart rate. The application was designed to be functional and intelligent. To make the watch convenient to use, we included a simplistic clock face, which includes the time, date, current heart rate, calories burned, steps, and watch battery life.

Along with this functionality, our application's main purpose was to collect data. The Fitbit Versa 3 device will automatically collect heart rate (raw and processed into features), steps, calories, intensity (measured in METs), and sleep using the sensors on the device. This data, upon a "sync" from the user's companion smartphone application, will automatically be stored by Fitbit. This data can be retrieved through the Fitbit API.

4.2.1 Application Logic

To complement this biometric data with real-time stress perception scores, our application listened to the heart rate sensor as the "prompting" trigger. A potentially "stressful" episode is classified using the prompting algorithm outlined in Algorithm 1. Our application classifies a stressful episode as the user heart rate increasing 35% above the resting heart rate. The resting heart rate is calculated by Fitbit once a day. This value is re-queried every time the heart rate sensor is turned on. Once the prompting trigger has been satisfied, the application prompts for stress level information to help correlate how the user's heart rate fluctuations impact their anxiety and stress.

Each stress level choice is represented using a button component in CSS. The options are: 1 (Not Stressed), 2 (A Little Stressed), 3 (Moderately Stressed), and 4 (A Lot of Stress). If a user does not respond to a prompt, which is displayed on the Fitbit face for a total of 7 minutes, the response is marked as a '0' in the database. Figure 7 shows visualization of the default and prompted faces of the application. On click of any button, the application is responsible for writing the corresponding stress level and timestamp of the click to a CBOR (Concise Binary Object Representation) file. The CBOR file with the data is stored to the file transfer queue and waits for steady bluetooth and mobile phone connection to transfer the file to the companion mobile app.

4.2.2 Storage Architecture

Once the data is stored to the mobile companion application, the file is deleted from the Fitbit to allow for storage maintenance and is then queued for transfer to

```
restingHR ← API(user.CurrentRestingHR);
percentageIncrease ← 1.35;
threshold ← restingHR * percentageIncrease;
currentHR ← HeartRateSensor.CurrentHR;
if currentHR ≥ threshold then
    sleep(60);                                   /* Wait for 1 minute */
    if currentHR > threshold then
        Display prompt and vibrate;
        answered ← False;
        timer ← CurrentTime;
        if response then
            answered ← True;
            Hide prompt;
            Record as 'entered' prompt;
        else
            if timer is 7 minutes & answered is False then
                Hide prompt;
                Record as 'missed' prompt;
            end
        end
    end
end
else
    currentHR ← HeartRateSensor.CurrentValue;
end
```

Algorithm 1: *StressManager* Application Prompting Algorithm

the cloud. Using **AWS API Gateway**, our companion application logic sends data to the API endpoint. The **Lambda Uploader**, responsible for building functions to handle requests and responses from the user, takes in the stress data and processes it for storage. Finally, the data is stored in **DynamoDB**, a key-value store, sorted by unique id (for each user) and containing the stress level and corresponding timestamp.

We chose to use DynamoDB to store the collected stress level data. The key-value data is efficient to query by the partition key or sort key, like extracting heart rate values for a specific user given start and end dates. It is also a non-relational database

User Type	Web Application Capabilities
Individual Participant	• **View Device Information** – Battery Life – Time and Date of Last Sync • **View Data** – Heart Rate Visualization over Specified Date Range – Stress Level Values Overlaid on Heart Rate Visualization over Specified Date Range – Download Specific User Data (Heart Rate or Stress Level) • **Take Daily Survey** – Perception Scores – Stress Correlation Activities
Researcher	• **View Device Information for Every Study Participant** – Battery Life – Time and Date of Last Sync • **View Data for Every Study Participant** – Heart Rate Visualization over Specified Date Range – Stress Level Values Overlaid on Heart Rate Visualization over Specified Date Range – Download Specific User Data (Heart Rate or Stress Level) • **Batch Download Data for Every User over Specified Date Range** – Stress Levels – Heart Rate – Daily Survey
Root	• **View Device Information for Every Study Participant** – Battery Life – Time and Date of Last Sync • **View Data for Every Study Participant** – Heart Rate Visualization over Specified Date Range – Stress Level Values Overlaid on Heart Rate Visualization over Specified Date Range – Download Specific User Data (Heart Rate or Stress Level) • **Register New Participants** – Add New Participants via Fitbit Authorization Flow

Table 4. Capabilities of Each Authorization Role in the Web Application

that allows scaling both vertically and horizontally, which is important to support data generated continuously from many wearable devices.

4.3 Web Application

As the number of participants grew, we needed a dashboard to manage the participants in the study and the data inflow from each Fitbit device. The static web

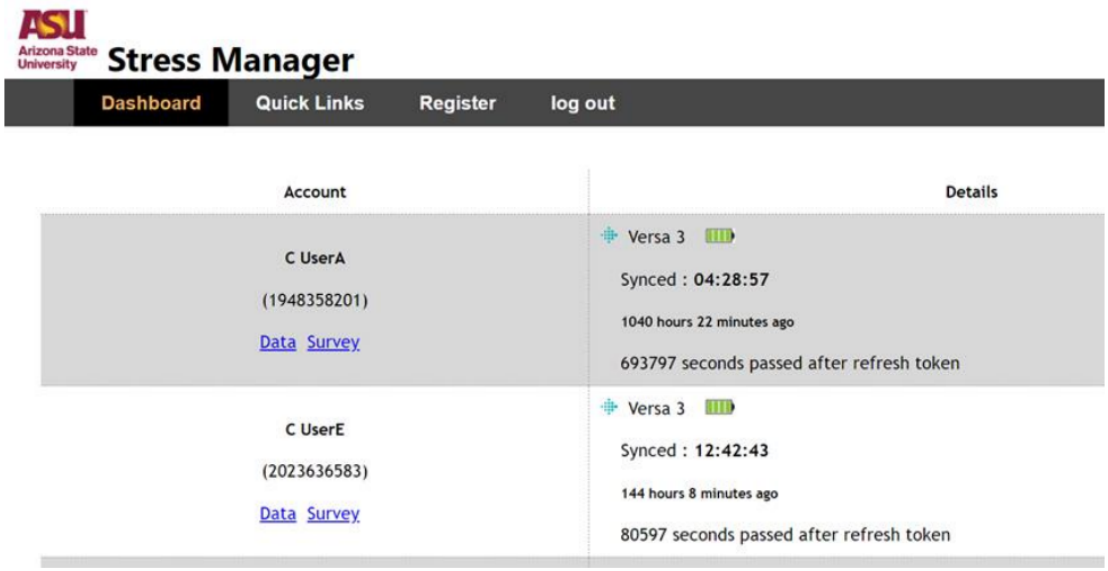

Figure 8. Snapshot of the *StressWatch* Web Application Showing Fitbit Device Information and Visualizations of Heart Rate Data from the Root Account

application, termed *StressWatch*, was created using Amazon Web Services (AWS) components, including the **Simple Storage Service** (S3), which holds the HTML, CSS, and JavaScript files. Through this web application, we can ensure that data is getting propagated as expected and can identify when specific users may be having trouble using their device.

4.3.1 Functionalities

The web application provides several functionalities depending on the logged-in user. Each participant is equipped with their own credentials to take their daily survey and view their heart rate and stress level trends. The researcher account is created for the research team to batch download data for each user and manage all active devices.

The root account registers new Fitbit devices but cannot batch download data like the research account. See Table 4 for more detailed information.

4.3.2 Architecture

Fitbit requires HTTPS protocol for outside communication, so we purchased a private domain name and connected the domain to the S3 bucket using **Route53**, **CloudFront**, and **Certificate Manager**. Most importantly, we utilized AWS Lambda functions to allow our static web application to act as a standard server. AWS Lambda is a server-less computing service that executes code without establishing a server. It is event-driven, executed only when the service is requested. Lambda helps the application build data processing functions by accessing other resources provided by AWS such as DynamoDB and S3. It can handle up to 250MB of code, and the execution time cannot be more than 15 minutes, which is sufficient for our needs. In the case of our application, we built lambda functions for creating and logging in user accounts and uploading and retrieving recorded data to the web interface.

We also utilized the **API Gateway** to build APIs. Our application runs in RESTful APIs, which requests and responds in JSON format with four methods, including CREATE(post), READ (get), UPDATE (put), and DELETE (delete). AWS's API Gateway provides users in the backend with an endpoint to access data in other services, such as DynamoDB. It also controls authentication to filter out unidentified requests. In our application, we built endpoints and mapped them to Lambda functions so that end-users can access and upload data. More specifically, the companion uploads the stress level data through an API gateway endpoint to DynamoDB.

Chapter 5

DATASET PROCESSING

After the data collection period, it became important to understand the characteristics of the data. Each cadet was left relatively un-monitored during the data collection window. This however required us to process the data at several large scale steps to learn trends about the cadets as a whole and prepare the data for machine learning training and inference. This phase consisted of data extraction, data contiguity and correlation, and noise removal techniques.

5.1 Data Extraction

The web application is responsible for making requests for stress level and Fitbit data. As specified in the sections above, Fitbit data is managed and stored by Fitbit itself. The stress data, since it is collected through the custom-built application, is stored in our personal DynamoDB instance.

5.1.1 Fitbit Data

Through the use of an OAuth Client Identifier, Secret and special Client Intraday heart rate permissions (see Figure 9), each user's collected Fitbit data could be accessed by our team. The Intraday service pulls any queried biometric value collected by Fitbit for a 24 hour period using an API request. Heart rate is sampled by the Fitbit heart rate sensor every 5-15 seconds when it detects "OnWrist" presence. As

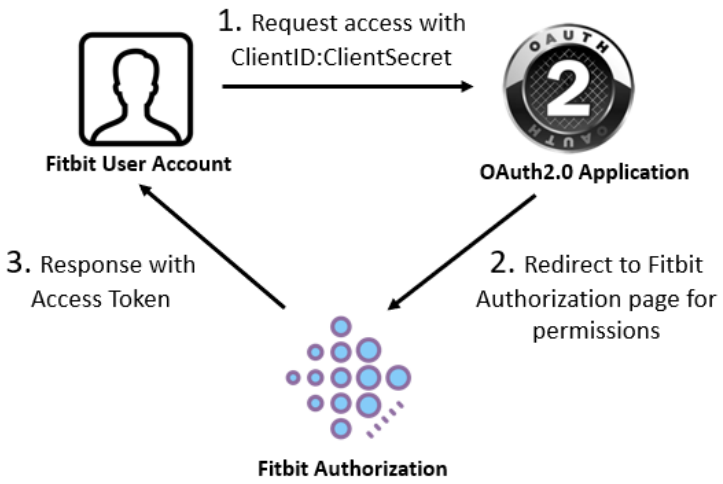

Figure 9. Authorization Flow for Fitbit Code Grant using OAuth2.0

part of our data acquisition process, we batch downloaded CSV files with this 5 to 15-second separated heart rate data and the corresponding timestamp of recording.

Other supplemental data were also downloaded. These data included steps, calories, resting heart rate, and METs. METs, a somewhat unfamiliar metric in the traditional physiology sense, stands for "metabolic equivalent of task". In other words, it is the rate of energy expended per unit. This information was collected and stored by Fitbit in a 1 minute interval, as it is not a linear growth but a measurement of change over time. These raw values and their corresponding timestamp of collection were also batch downloaded into formatted CSV files. The Fitbit API queries were made for each day in the specified date range and were sent and processed in the web application backend.

5.1.2 Stress Level Data

Each stress level recorded by each participant is stored with its timestamp in the Stress Level table in the key-value store. The database schema is as follows:

$$deviceId : stressTime : heartRateTime : stressLevel$$

Each Fitbit device was assigned a unique device identifier. This identifier serves as the partition key in the database. API requests were made to the stress level table in DynamoDB and the results were downloaded into formatted CSV files. Query parameters included the device id and the start and end date range chosen by the user on the web application.

5.2 Data Contiguity and Mapping

We began by exploring the features of the biometric data downloaded from Fitbit. In particular, we were interested in the granularity of the heart rate data itself. Fitbit provides multiple heart rate data granularities: 1-second, 1, 5, and 15-minutes. Due to the intricacy of heart rate and how it quickly is impacted by feelings of anxiety and stress, we chose to specifically leverage the second-by-second heart rate data in our processing. Fitbit documentation specifies that heart rate is sampled every 5-15 seconds, so the 1-second granularity provided a heart rate value in beats per minute (bpm) between 5-15-second intervals. To normalize this data and ensure that the beat-by-beat variation stayed consistent, we insert heart rate values if two samples are either 10 seconds or 15 seconds apart. We inserted values using a data contiguity algorithm, illustrated in Figure 10. After this normalization approach, we maintained a 5-second variance between samples that were collected either 5, 10, or 15 seconds

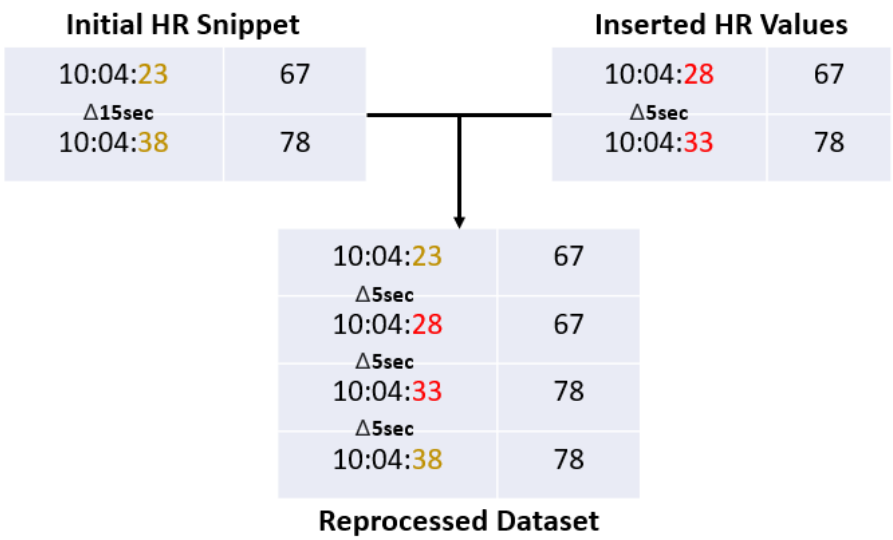

Figure 10. An Example of the Data Contiguity Processing Done on Heart Rate Files to Maintain 5-second Intervals from Beat-to-beat

apart. Rows sampled more than 15 seconds apart were kept as is, since it implies that the heart rate sensor may not have been continuously worn.

With nearly 194MB of collected data, the next step was the correlate biometric data to the target classified label and reduce the data complexity. The label in our case was the stress level provided by the user. This was the ground truth, specified by each participant, to explain the biometric patterns. This thesis mainly focuses on the relationship of stress and heart rate. As a result, heart rate was the only biometric measure correlated to the stress level. Each stress level also has its timestamp of creation. This is when the heart rate met the threshold trigger logic in the Fitbit application and created the prompt for the user. Since the threshold was based on the resting heart rate value, the resting heart rate (which was collected by Fitbit in day granularity) was also extracted around the stress level window.

To provide the most number of data points and history of heart rate fluctuation to each stress level, we chose to extract the 5 minutes of heart rate prior to the stress

prompt. 5 minutes were chosen to give the most amount of heart rate history in hopes that patterns could be deduced from heart rate fluctuations leading up to a potentially stressful episode. In our re-processed heart rate dataset, 5 minutes of heart rate data would translate to 60 samples (12 samples per minute * 5 minutes). As explained in a following subsection, other history samples (ie. 1 - 4 minutes : 12 - 48 samples before the stress prompt) were also extracted to view the optimal prediction window. 60 samples was marked as the upper bound of the data extraction so all 60 samples were used in the noise detection and removal phases.

For each stress level and timestamp of prompt creation from a given cadet, our algorithm leverages binary search on their heart rate data to find the closest timestamp before or equal to the target timestamp. The logic then backs out from that timestamp by 60 samples (the upper bound of the detection range). The first timestamp (the timestamp that is 59 samples away from the closest timestamp) must be within 5 minutes of the closest timestamp, and the last timestamp (the closest timestamp from the target timestamp) must be within 5 seconds of the target timestamp. Figure 11 shows an illustration of the data correlation task with bounds checking. The bounds are changed depending on the number of samples extracted for each dataset.

At this point, our dataset consisted of stress level values and their 60 samples of heart rate. The frequency of stress level responses was largely biased. As mentioned in the prior section, users were initially given stress level options, 1-5. This range was condensed to 1-4 to simplify choices for the users. If a user did not respond to a prompt, which was displayed for 7 minutes, the response to the stress level was marked as a '0' in the database. As seen in Table 5, our dataset is heavily biased toward missed responses, and the frequency of the stress level intensity decreases exponentially.

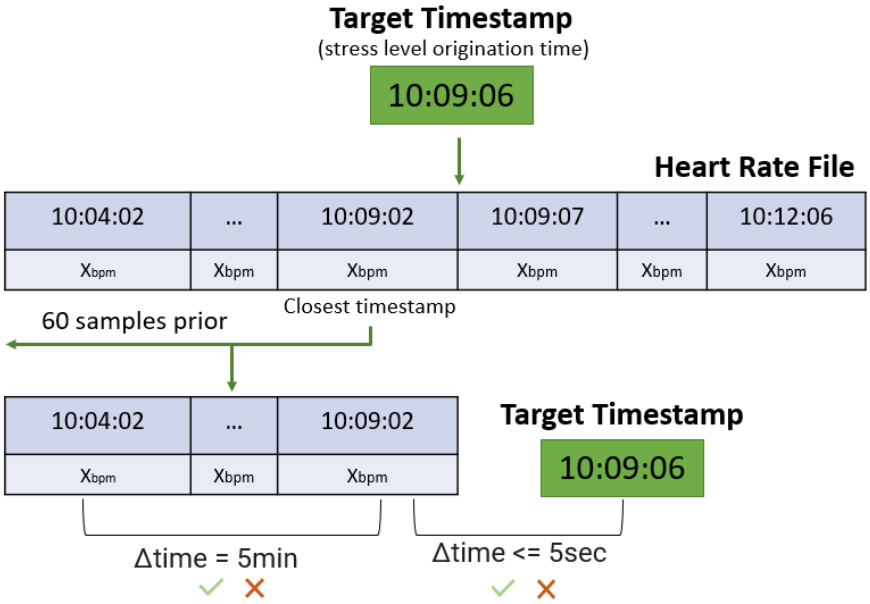

Figure 11. Illustration of the Data Correlation Process for Stress Levels to the 60 Contiguous Samples of Heart Rate Prior to the Stress Level Timestamp

Stress Level	Number of Responses in Dataset
0 (“Missed Prompt”)	9397
1 (“Not Stressed”)	3060
2 (“A Little Stressed”)	485
3 (“Moderately Stressed”)	123
4 (“A Lot of Stress”)	31
Removed 5 (“Extremely Stressed”)	2

Table 5. Frequencies of Responses to Each Stress Level Option in the Dataset

To manage the imbalanced nature of the dataset, the stress levels and corresponding heart rate data were passed through an oversampling technique, called SMOTE, to add samples to the minority class. Further, to simplify the classification task, we decided to leverage a binary classification problem. For all datasets created in the following subsections, we grouped stress level ‘1’ into the “Not Stressed” class and levels ‘2-4’ values in the ‘Stressed’ class.

5.3 Noise Detection

Data collected in unconstrained contexts inherently have noise. Fitbit devices, which are still being developed and improved upon, do not have the ability to collect data without any environmental bias. As explored in Section 2, Fitbit devices tend to underestimate heart rate. Studies have also found that improper wrist placement, tattooed skin, and sudden movements can contribute to wildly fluctuating, inaccurate data. At this point in time, such limitations are common in any wearable device.

All the Fitbit data collected was done in unconstrained contexts. Because of this characteristic, we are unclear if/how much noisy data exists in our dataset. If noise does exist in the dataset to some capacity, noise removal techniques can be leveraged to handle the outliers. In the context of this investigation, we define 'noise' as any heart rate characteristics that are much higher or much lower than the other frequent characteristics in the dataset.

We begin by removing any samples of heart rate that correspond to a '0' stress level. These values are removed because they were missed responses, indicating that we do not have a ground truth label for that period of heart rate data. Additionally, missed responses tended originate from infrequent watch wearing, leading to a higher probability unnatural heart rate fluctuations.

5.3.1 Beat Variance Frequency

We decided to extract the absolute change across beats for each sample of heart rate data. Since each beat value should be 5-seconds apart from the next, this analysis can help us identify sharp increases and decreases in heart rate (bpm). Some

characteristics of noise from the Fitbit heart rate sensor includes random drops or spikes in recorded data. Each absolute change value was plotted on a histogram to see the most frequent beat variation values. We figured the more common values would naturally fall together since we assume noise will be present, but not largely frequent.

Figure 12 shows the frequency distribution for this variance characteristic. It becomes immediately apparent that the most common beat to beat variance falls from -10 to 10. The next common range is -20 to -10 and 10 to 20. We can see that on both ends of the histograms, there are very few occurrences of larger variance. We classify these larger variants as "noise" or outliers in the dataset.

If we look closer at these outlier values, we see large increases across some beat values. In the window below, we see a sample of heart rate values:

87	88	88	**88**	**141**	141	155	156

From value **88** to value **141**, we see a 53bpm absolute change in a 5-second span. When we look at the histogram (Figure 12), we can see that 53 bpm absolute change is in the outlier range. Another example of a possible outlier is the window below:

127	127	130	**120**	**86**	87	84	81

Here, we see a -34bpm absolute change from **120** to **86** in a 5-second span. The left end of the histogram (Figure 12) shows that -34 falls in the outlier range. It is unclear whether we should count these windows as noise or if this is normal in highly stressed individuals. To manage these potential outliers, filtering techniques were leveraged, as outlined in the next section.

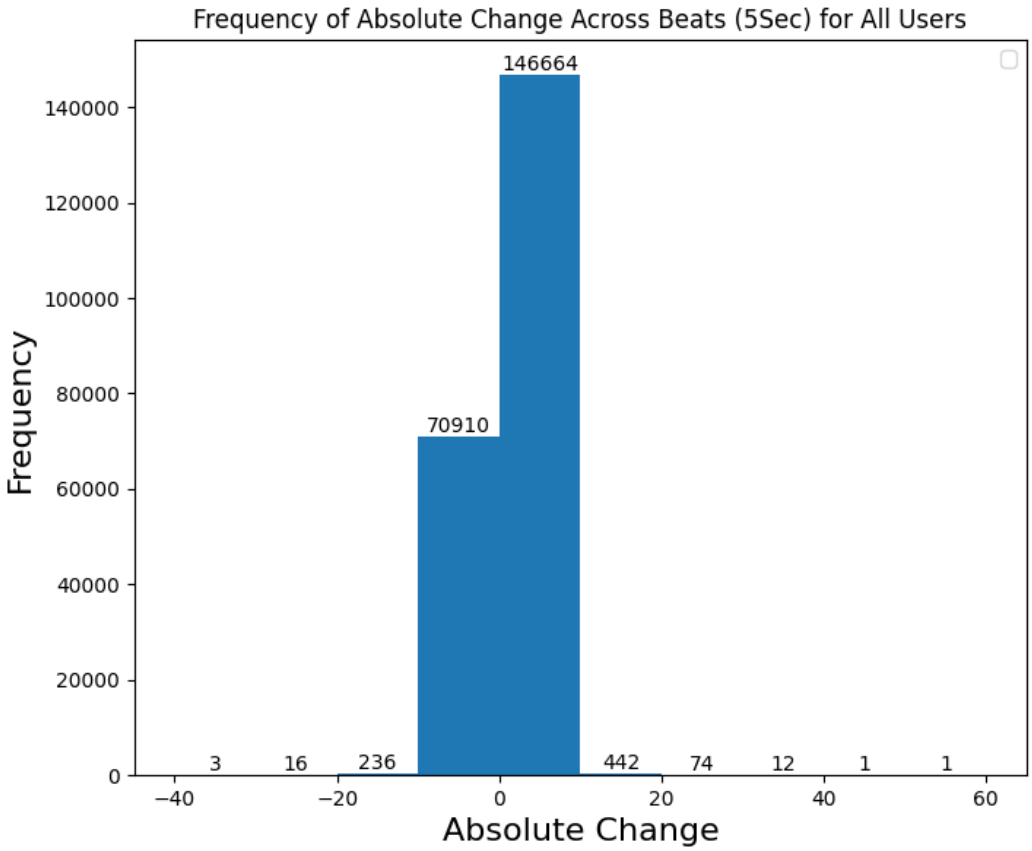

Figure 12. Histogram Representation of the Absolute Change Beat Variance of the Dataset

5.3.2 Range Frequency

We also decided to investigate the range value of the 60 samples of heart rate. This means that we will find the minimum and maximum value in the 60 sample window, and take the difference. This will show us how much the heart rate changed over the course of the 5 minutes of heart rate data. Similar to the previous characteristic, we will take the data to a histogram to see possible outliers the end of the spectrum.

Figure 13 shows the histogram of the range characteristic. We can see from the

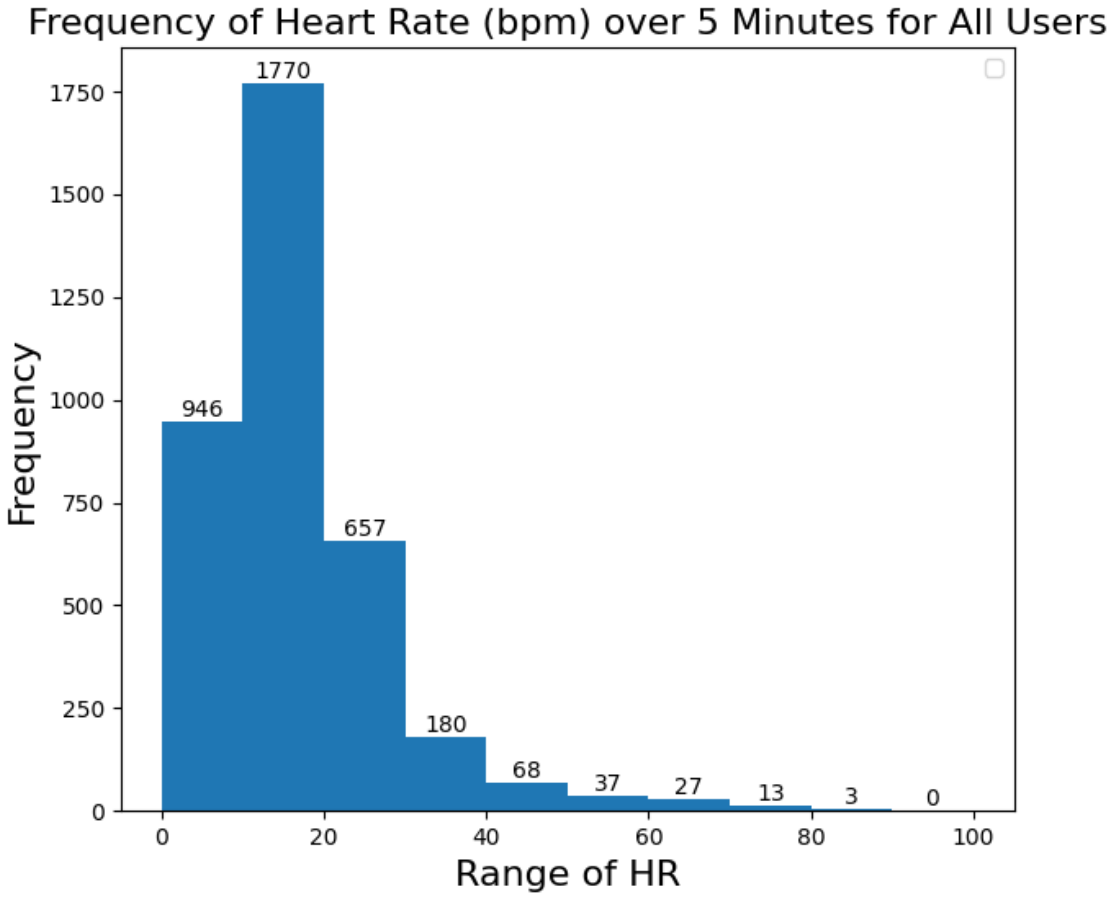

Figure 13. Histogram Representation of the Range Variation of Heart Rate over 60 Samples (5 Minutes)

histogram that the most common range of heart rate over the 5 minute interval sits from 0-30bpm. Outliers in this data characteristic based on the histogram appear to be values above 70bpm. It is unclear if this is a normal heart rate range over 5 minutes, since we do not know the user that it belongs to or have their heart health and trends.

Some outliers from this investigation are shown in the table below. The maximum

value and minimum value columns are the respective values taken from the 60-sample window of data.

Maximum Value	Minimum Value	Range Variance
145	59	**86**

Maximum Value	Minimum Value	Range Variance
163	82	**81**

These samples and range values are on the higher end of the most common range variance values in the histogram. This indicates to us that over the 5 minutes, the users heart rate changed more than 80bpm. While this falls out of the most common range of heart rate variance for other values in this dataset, we are unsure if we can attribute this sample to noise or simply a biological reaction to an event or feeling. Again, to manage these potential outliers, filtering techniques were leveraged, as outlined in the next section.

Through this noise detection investigation, it becomes apparent that there is no clear bounds or values to determine noise. This investigation does show us the variability of our dataset, and how on a beat-by-beat level, the patterns of change are important. Due to the real-world data collection strategy used in this study, we can deduce that some outliers observed will be unnatural.

5.4 Noise Removal

From a machine learning perspective, abnormalities or outliers in the data can impact how the model draws trends and patterns. To address any finer grained beat-by-beat characteristics, we decided to run our processed dataset through different

noise removal filters designed for outlier detection and smoothing. These filters are described in detail in Section 2. We tested the four filters, Hampel, Median, Savitzky-Golay, and Wiener on the datasets to see which was most effective on the deep neural networks.

5.4.1 Hampel Filter

This filter is designed for outlier detection in time series. A study by Ghaleb et al. [9] is one of several that leverage the Hampel filter to detect outliers in their ECG (Electrocardiogram) data. ECG data is another measure of heart health. It measures how often and how regular the heart beats. The Hampel filter has not been used in heart rate data measure in beats per minute in any publicly available studies. The Hampel filter was implemented using the Python Heart Rate Analysis Toolkit (*heartpy*). The filter size was set to 12, meaning 6 data points were taken from the left side of the current heart rate sample, and 6 data points were taken from the right side of the current heart rate sample for the analysis. 12 was chosen arbitrarily based on the generalized idea that there are 12 samples for every 1 minute of data in the 60 sample range.

5.4.2 Median Filter

This filter is a common algorithm used in digital image processing to preserve edges of images and smooth values. This filter is used very commonly in heart rate estimation and artifact removal, as seen in a study conducted by Yang et al. [34]. They found that this filtering technique was able to successfully remove artifacts, and

proved to be generally intuitive and computationally simple. The Median filter was implemented using the Scipy Signal library (*medfilt*). The kernel size, indicating the local window size was set to the default, 3.

5.4.3 Savitzky-Golay Filter

The Savitzky-Golay filter is a type of low-pass filter, meaning that it gets rid of signals that are below a selected cutoff. It is often used to clean signals and smooth data. It has been used in heart rate studies, including a study by Chatterjee et al. [5], which uses PPG heart rate data - data collected using a photoplethysmograph. This filter was implemented using the Scipy Signal library (*savgol_filter*). The parameters given to this filter include window_length = 5 (default) and the order of the polynomial, which is set to 2. These are arbitrarily chosen.

5.4.4 Wiener Filter

The Wiener filter is a filter that takes the linear estimation of the original data using a stochastic framework. It is commonly used to remove any additive noise and to invert blurring in data. Temko et al. [30] explores the use of a Wiener filter to estimate heart rate from photoplethysmography. This filter was implemented using the Scipy Signal library (*wiener*).

Table 6 shows the distributions of absolute beat change on each filtering technique. Compared to the non-filtered frequencies, the Hampel and Median filters tend to move more samples toward the most frequent distributions. They are not able to smooth the outliers on both ends of the range. The Savitzky-Golay and Wiener filters seem

to do a better job of clearing out the outliers on both ends of the range while moving more samples to the most common ranges.

Ranges (bpm)	None	Hampel	Median	Savitzky-Golay	Wiener
-40 ↔ -30	3	3	2	0	0
-30 ↔ -20	16	16	10	2	6
-20 ↔ -10	236	229	158	79	39
-10 ↔ 0	70910	64368	48056	103426	88210
0 ↔ 10	146664	153266	169759	114626	129992
10 ↔ 20	442	392	311	217	98
20 ↔ 30	74	70	52	9	11
30 ↔ 40	12	13	9	0	2
40 ↔ 50	1	1	1	0	1
50 ↔ 60	1	1	1	0	0

Table 6. Frequencies of Each Absolute Beat Change Based on BPM Range Using the Different Filtering Techniques

Table 7 shows the distributions of the range variance over the 60 samples (5 minutes) leading up to the stress prompt of heart rate data. This, as a refresher, shows how much the heart rate changed over the course of the sample window. The analysis of the frequencies here are not as clear-cut as the previous characteristic comparison. Each filter pushes the samples toward the 0-10bpm range, but do not thin out the outlier ranges as much. This characteristic may not be as important as the beat variance characteristic since heart rate can fluctuate a lot in the course of 5 minutes.

Ranges (bpm)	None	Hampel	Median	Savitzky-Golay	Wiener
0 ↔ 10	946	1092	1231	1162	1285
10 ↔ 20	1770	1684	1706	1713	1648
20 ↔ 30	657	611	507	531	503
30 ↔ 40	180	172	138	169	145
40 ↔ 50	68	63	54	54	53
50 ↔ 60	37	36	30	33	33
60 ↔ 70	27	28	26	22	24
70 ↔ 80	13	12	9	14	9
80 ↔ 90	3	3	0	3	1
90 ↔ 100	0	0	0	0	0

Table 7. Frequencies of Range Variation Based on Bpm Range over 60 Samples (5 Minutes) Leading up to the Stress Prompt Using the Different Filtering Techniques

5.5 Dataset Variations

5.5.1 Approach 1 - Contiguous Heart Rate Samples

In the 5 individualized filtered datasets created in the noise removal stage, 60 samples (12 samples * 5 minutes) representing 5 minutes of heart rate data were extracted. We considered this as the upper bound of the extraction window due to the prompting logic of our application. Noise detection and noise removal were conducted on this larger sample range to encapsulate the larger variation of data. To experiment with variate duration of heart rate history, in addition to the 60 samples of heart rate (5 minutes), we also extracted data using the following sample history prior to the stress prompt creation:

- 48 samples of heart rate (4 minutes)
- 36 samples of heart rate (3 minutes)
- 24 samples of heart rate (2 minutes)
- 12 samples of heart rate (1 minute)

5.5.2 Approach 2 - Heart Rate Features

As proposed in Sim et al. [28], which conducted an initial experiment into the unconstrained police officer data and stress detection, our second approach leverages feature extraction using heart rate and resting heart rate. In the study, the authors extract 70 samples of heart rate, chosen as a potential upper bound, leading up to the stress prompt. With the heart rate data, which is not normalized or made contiguous, the authors extract the following features from the data to feed into the ML models.

1. Resting heart rate (as calculated by Fitbit)
2. Mean of 70 heart rate samples
3. Standard deviation of 70 heart rate samples
4. Maximum sample value of 70 heart rate samples
5. Minimum sample value of 70 heart rate samples
6. Difference in resting heart rate:
   $(mean(HR) - resting(HR))/resting(HR) * 100$
7. Root mean square of successive differences: $\sqrt{mean((HR)^2)}$

Each dataset was passed through the filtering techniques mentioned above for each approach. To capture the performance differences between one cohort and every cadet throughout the study, the datasets are also created in two batches, "Full Dataset" and "One Cohort". All datasets used for neural network training were processed using SMOTE to add samples to minority class. Additionally, as specified above, the stress levels were converted into binary classes: 'Stressed' and 'Not Stressed'. Table 8 shows how the classes were created using the different stress level values. The experimental results achieved in the thesis from these dataset variations are conducted and explained in detail in the following sections.

Class	Stress Levels Used
"Not Stressed"	Level 1
"Not Stressed"	Levels 2 - 4

Table 8. Stress Levels Used for Each Binary Class - "Not Stressed" and "Stressed".

Chapter 6

DATA ANALYSIS

This section provides a deeper look into the data collected, specifically related to study compliance and high-level biometric patterns from the extracted data.

6.1 Compliance Analysis

This subsection focuses on the compliance rates and takeaways from the study as a whole. Compliance is an important aspect of conducting a successful study, directly impacting the validity and reliability of study results. Over the course of the entire 9 month data collection period, we collected thousands of raw data points and stress level values. Due to the large scale of incoming data, we weren't entirely clear on the full extent of our data and the overall feasibility of the study. This analysis focuses on the prompt generation and response rates across the entire study and specific cohorts to learn overall compliance trends from participants.

Figure 14 shows the percentages of missed prompts to responded prompts for all users throughout the study. This pie chart corresponds to Table 5 in the previous section. We see immediately that approximately 3/4 of all prompts created were missed or not responded to. There are two possible conclusions we can make from this data. One possibility could be that the prompting threshold (35% increase over resting heart rate for 1 minute) was too sensitive. Users could have been getting prompted too frequently and didn't respond to the prompts. Another possible conclusion is that users were too consumed by their stress to reply to the prompt. The stress, or the

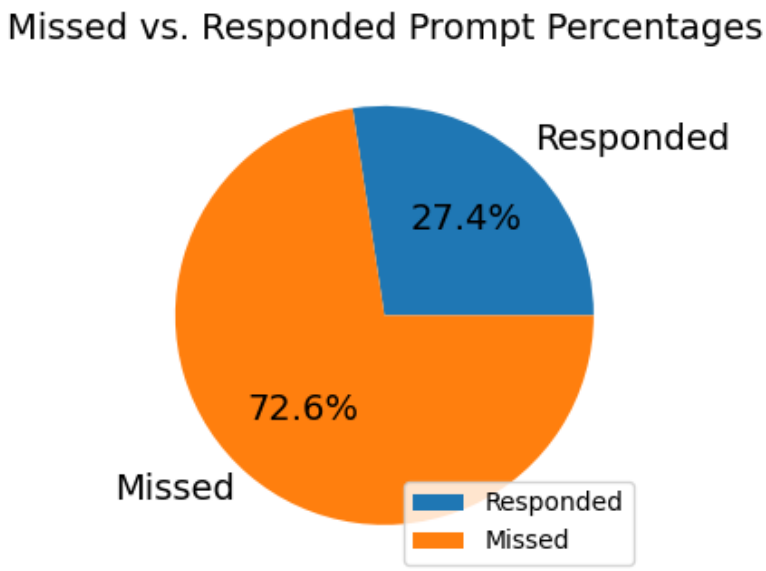

Figure 14. Response Frequencies Across the Full Study Dataset

scenario itself, had the potential to be consuming and users couldn't break away from their current situation to provide a response. To remedy this in the future, we can address the prompting threshold to make it less sensitive, and provide a way for users to give stress scores after the moment may have passed to make up for lost data.

Figure 15 looks at response percentages across cohort-specific data. The distribution across cohorts shows us that in almost every cohort, more than half of the prompts created were missed. Cohort F, which only consisted of 2 users, was the best in terms of response. The heart rate threshold could be too sensitive, causing the application to prompt when there is no stress or when it is off the wrist. The threshold was set to 135% above the resting heart rate, which may be too low. Additionally, we have found that the "OnWrist" presence feature for the Fitbit is at times inaccurate in determining wrist presence. When the watch is not being worn and the "OnWrist" presence is still

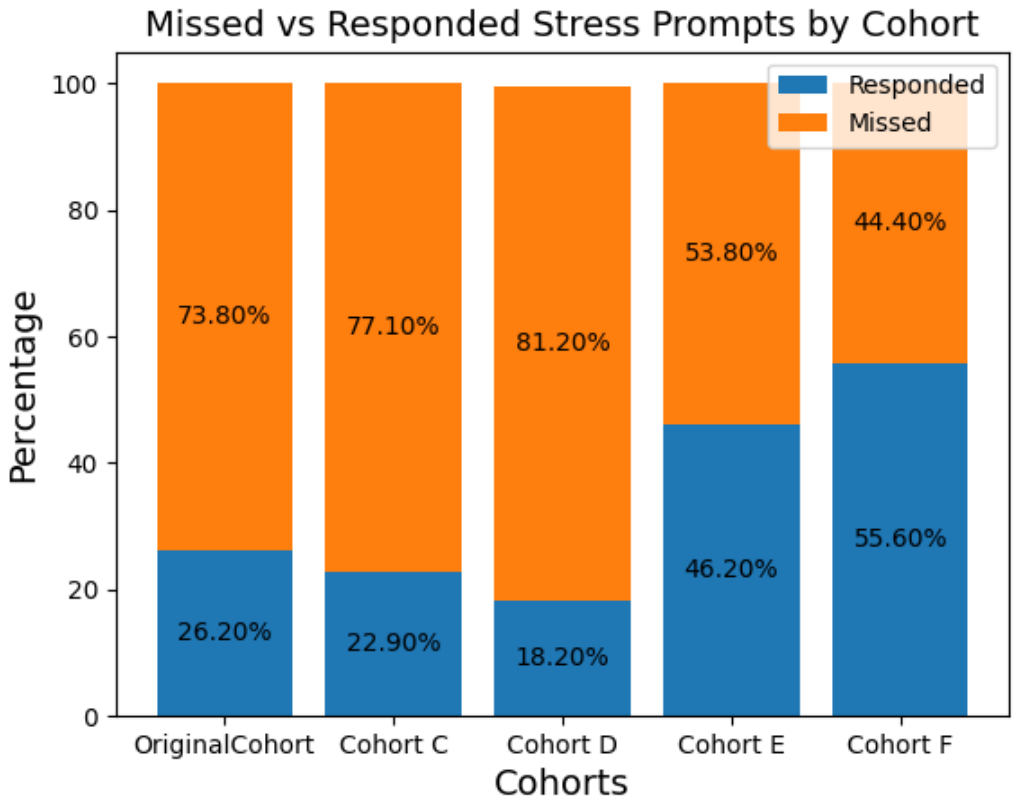

Figure 15. Response Frequencies Visualized as a Percentage Sorted by Cohort. Original Cohort: N = 15, Total Prompts = 12976, Cohort C: N = 10, Total Prompts = 2309, Cohort D: N = 4, Total Prompts = 890, Cohort E: N = 6, Total Prompts = 1204, Cohort F: N = 2, Total Prompts = 410

true, the heart rate sensor continues to pick up noisy or faulty heart rate values. Our application, dependent on the heart rate sensor and not looking at wrist presence, will continue to prompt users, leading to missed prompts. Over the course of the academy, the demand for high psychical and mental attention also increases. Users probably did not have time or the ability to respond to the prompts immediately.

Figure 16 shows the percentage distribution of missed versus responded prompts by gender. This statistic takes the total number of prompts for each gender and finds the response and non-response rates. We can see in the figure that female cadets

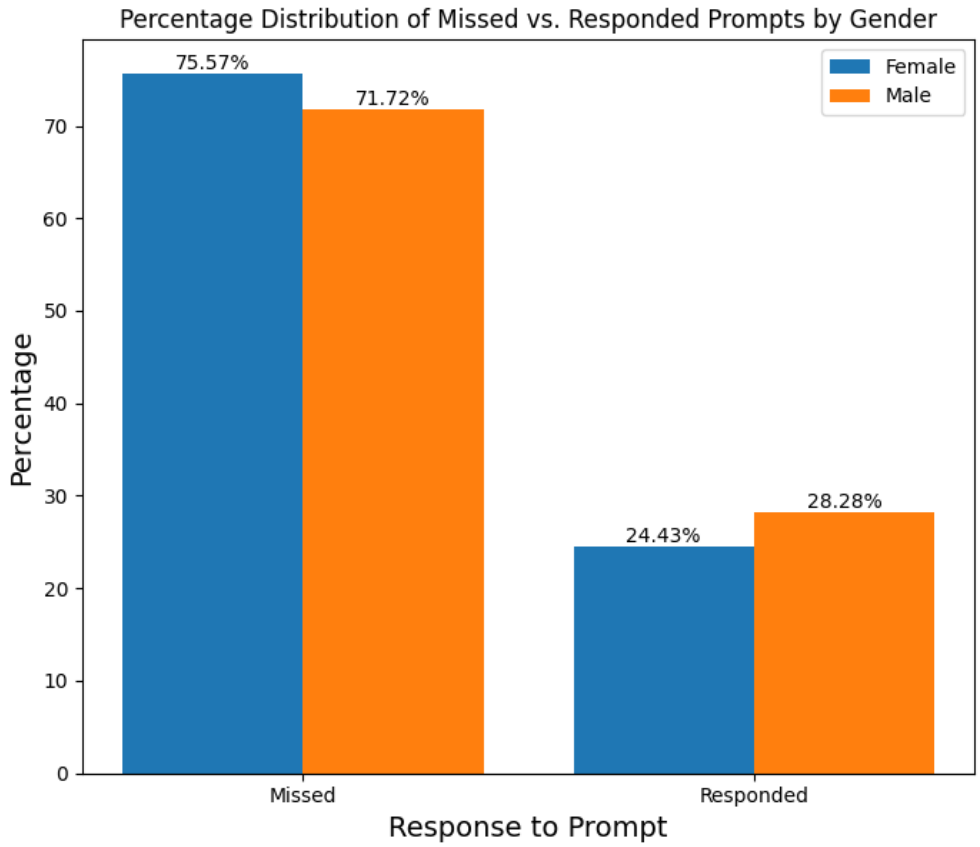

Figure 16. Percentage of Prompts Missed vs. Responded by Gender. Female: N = 6, Total Prompts: 3982, Male: N = 31, Total Prompts: 13803

tended to miss more prompts than male cadets. This could be paired with some of the emotion regulation data collected from the pre- and post-academy questionnaires to see how women managed their emotions and feelings.

Figure 17 shows the average number of prompts generated each week during the study period by each cohort. The number of prompts per week for each cadet in each cohort were averaged. We specifically focus on the number of prompts generated to understand how much the Fitbit was worn. We assume here that if prompts were

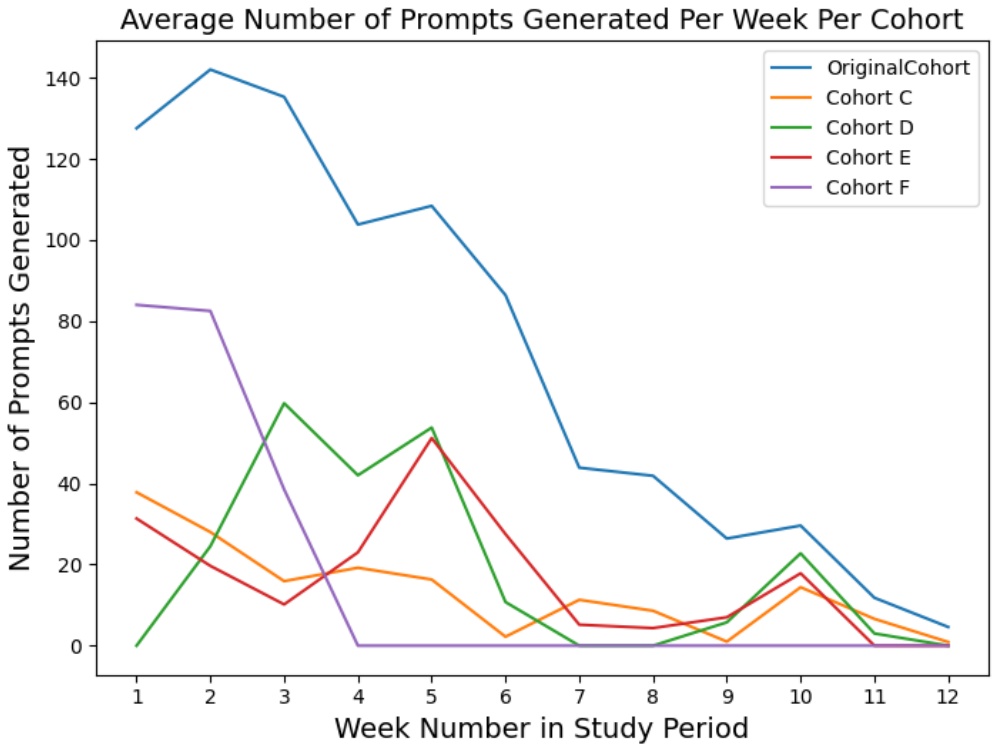

Figure 17. Average Number of Prompts Generated from Each Cohort Across Weeks in the Study

generated, the Fitbit was on and there was a high probability that the Fitbit was on the wrist (or recently on the wrist). The figure shows that over time, the number of responses generated decreased. This tells us that compliance was higher when the study began than later in the study. Overall study perception and attitude could play a large part in these trends. Frequent prompting could have caused disruptions during high-focus academy tasks, which typically appeared later in the academy curriculum, leading to negative feelings toward the device.

6.2 Biometric Analysis

Analyzing biometric trends from heart rate data can provide insights into a person's stress response and help identify triggers that lead to that stress. This subsection looks into the heart rate data collected and its relationship to prompting and the stress level scale. Learning generalized trends can guide psychologists to understanding what factors and patterns indicate higher stress.

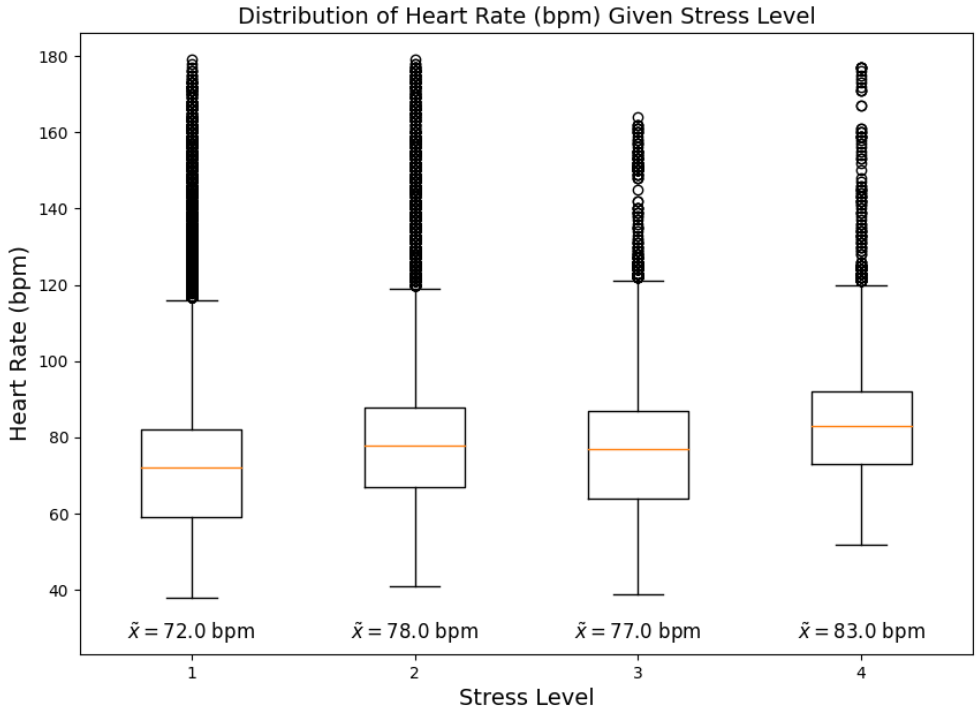

Figure 18. Box Plot Indicating Median, 25th, and 75th Percentile for Heart Rate Values 5 Minutes Prior to Stress Prompt Creation for Each Stress Level Value

Figure 18 shows the distribution of heart rate values for each stress level metric. The boxplot shows the median values for each stress level, along with the 25th and 75th percentile values. We can see from the figure that the median value slightly

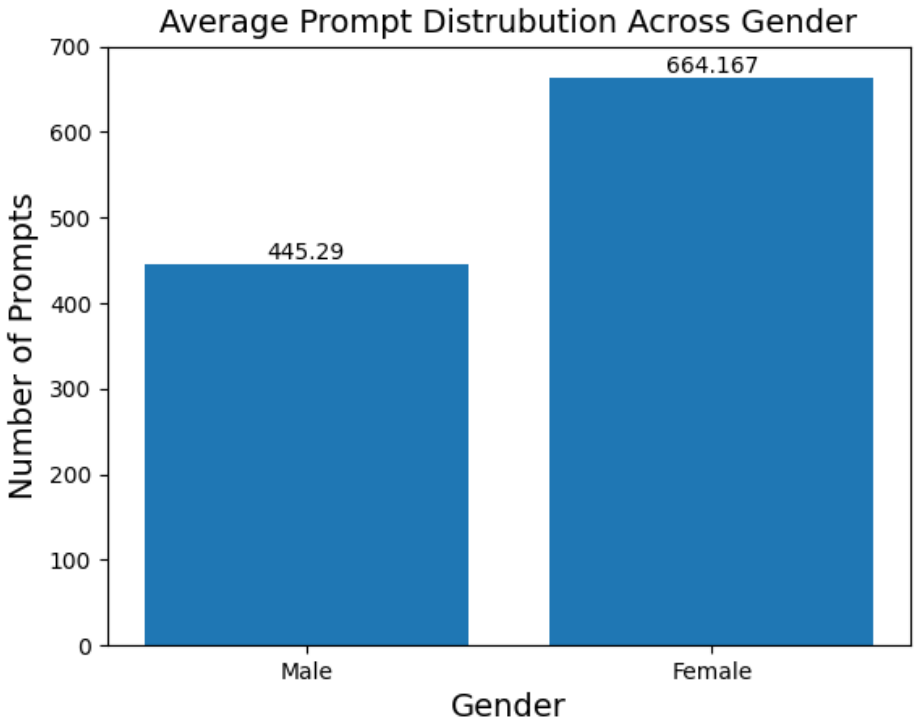

Figure 19. This Graph Shows the Average Prompts Created Across Males and Females in the Study

increases across increased stress levels, which could support the theory that higher heart rate yields higher stress. This boxplot also shows the outliers outside of the range. We can see that other stress levels have data well past the maximum range, indicating that even though heart rate is high, stress levels may not always follow suit, contradicting the earlier assumption. These outliers could also be from high-intensity exercise, which naturally increases heart rate.

Figure 19 shows the average number of prompts generated across genders. Here, the assumption is that if a prompt was generated, the watch was worn and the current heart rate of the user at the time of the prompt was higher than the threshold. This figure corrects for proportion, calculating the average number of prompts generated for

each gender group. Assuming the threshold is appropriately sensitive, this figure shows that women had an average of 664 prompts during the course of the study and men had an average of 445 prompts during the study. These values suggest that women had higher heart rate and were consequently prompted more often. These values could also suggest that women were more compliant overall in the study, yielding more Fitbit wear time and more generated prompts.

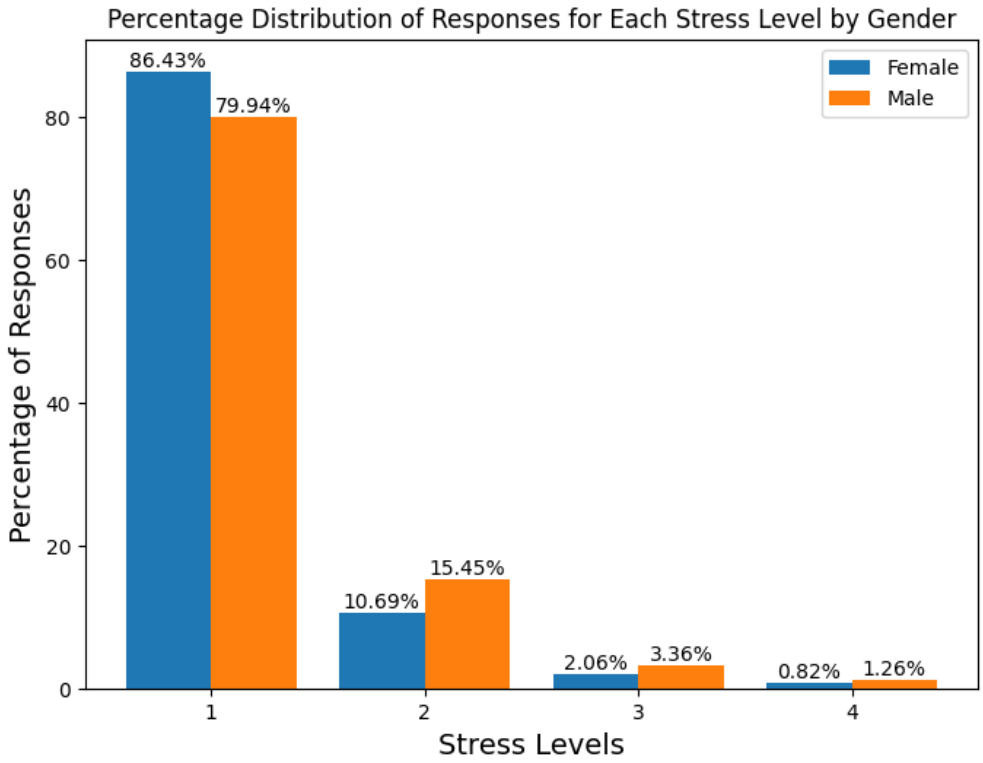

Figure 20. This Graph Shows the Percentage of Responded Prompts for Each Stress Level Across Males and Females in the Study. Females: N = 6, Total Prompts = 973, Male: N = 31, Total Prompts = 3903

Figure 20 illustrates the percentage of responses for men and women split by stress level values. Figure 18 showed that a higher heart rates tended to correspond to higher amounts of stress. With more average prompts from women, we can possibly

deduce that women had higher heart rate values overall. Higher heart rate could have implied higher levels of stress, per Figure 18. This figure, however, shows that on average, men tended to respond with higher stress levels than women.

This section looks into different characteristics of the collected data to draw some trends and conclusions about compliance and biometrics. The takeaways are as follows:

- Approximately 3/4 of all prompts during the whole study were not responded to. More specifically, on a cohort level, most cohorts had nearly half of the generated prompts missed.
- Over the course of the study, the number of prompts generated decreased. The number of prompts indicate Fitbit wear time, since prompts were only generated when the heart rate sensor was marked as active.
- According to 18, the corresponding heart rate values tended to increase as stress levels increased.
- Female cadets had more generated prompts on average than the male cadets, indicating higher compliance and higher heart rate values.
- Male cadets responded to higher stress levels (2 - 4) more than women.

Chapter 7

STRESS DETECTION ON DEEP NEURAL NETWORKS

With these carefully processed and curated datasets, the next step is to design deep neural networks to learn features of stress and heart rate to detect situations of stress. With meticulously crafted neural networks, our eventual goal would be to catch moments of intense stress in an accurate and non-disruptive manner. Several papers have already begun investigating stress detection using deep neural networks (see 'Related Works' section). This section goes into the training and evaluation results of several different deep neural networks using the two approaches on dataset formulation, 1) heart rate contiguity and 2) heart rate feature extraction. The deep neural networks leveraged in the experiments have different capabilities, strengths and weaknesses. They are highlighted in subsequent subsections. These datasets are initially comprised of user's heart rate (in bpm) and their relevant features leading up to a stress prompt, and are then processed as follows to eliminate any potential noise pollution:

1. Raw, Non-Filtered Heart Rate
2. Hampel Filtered Heart Rate
3. Median Filtered Heart Rate
4. Savitzky-Golay Filtered Heart Rate
5. Wiener Filtered Heart Rate

7.1 Models

7.1.1 Deep ECGNet

The Deep ECGNet was created by Prajod et al. [24] in their work on stress detection using ECG heart rate data. This study leverages a publicly available wearable device dataset, called WESAD, to test their created neural network. Through their experiments, they are able to achieve a 90.8% accuracy using the WESAD dataset and the Deep ECGNet. The architecture of the Deep ECGNet is seen in Figure 21. This model has one convolutional layer, two dropout layers, one max pooling layer, two batch normalization layers, and two LSTM layers. An LSTM layer is a recurrent neural network layer typically used for time series data to detect data dependence on time. This model uses its convolutional neural network layer to extract features from the input data. It then uses the two LSTM layers to extract temporal (time-based) features from the extracted features. This methodology makes this model effective on contiguous, sequential, time-series data. Table 9 shows the hyperparameters used for training.

Hyperparameter	Value
Loss	Binary Cross Entropy
Optimizer	Adam
Learning Rate	0.001
Batch Size	156

Table 9. Hyperparameters for Deep ECGNet

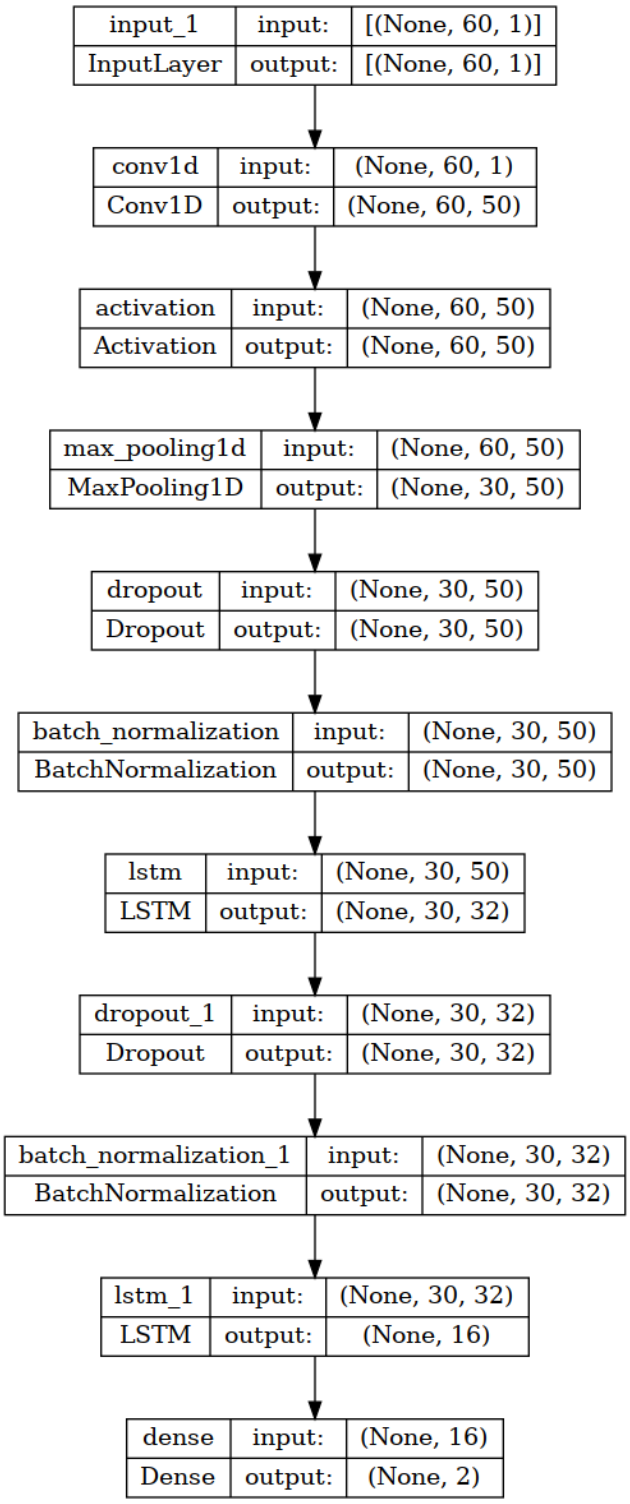

Figure 21. Architecture of the Deep ECGNet

7.1.2 Feed Forward Neural Network

This network was proposed in Sim et al. [28] for stress detection networks. It is proposed as a relatively simple network (see Figure 22), comprised of six hidden layers activated by the ReLu function and an output Sigmoid function layer. The output layer with the Sigmoid function outputs a probability between 0 and 1 which makes the model useful for binary classification tasks. This model has unidirectional layers without any feedback connections. This model works well with tabular data, which is often read from a CSV. From the study, the authors are able to achieve a 95.98% accuracy using a feature extracted dataset. This model is used as created to measure the baseline of the larger dataset created here compared to the smaller dataset leveraged in Sim et al.

Hyperparameter	Value
Loss	Binary Cross Entropy
Optimizer	Adam
Learning Rate	0.005
Batch Size	156

Table 10. Hyperparameters for Feed Forward

7.1.3 Multilayer Perceptron Network (MLP)

The Multilayer Perceptron Network is a type of artificial neural network that uses several layers of input nodes and backpropogation for training. A study by Li et al. [16] investigates the use of the MLP network on wrist-worn wearable device

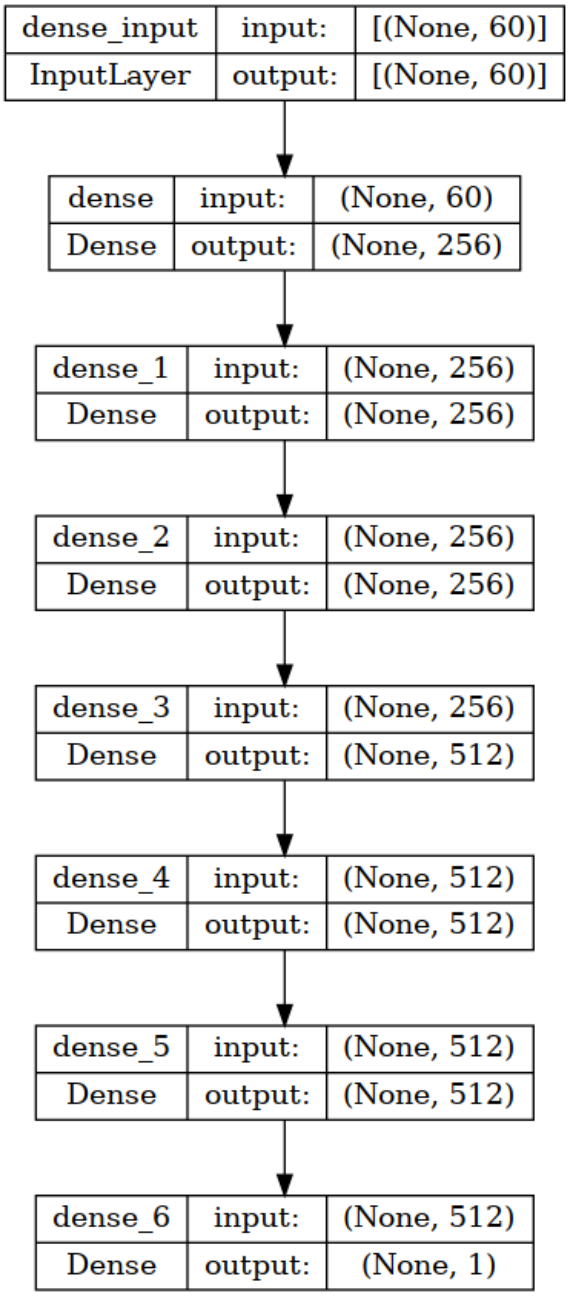

Figure 22. Architecture of the Feed Forward Neural Network

data. Through the use of publicly available datasets, they were able to achieve 99.65% accuracy for binary stress classification. This model is a type of feed forward neural network where each perceptron takes a weighted sum of its inputs, adds bias, and applies activations. This model works well with tabular data, which is often read from a CSV. The MLP model was taken from the sklearn.neural_network package. The first hidden layer had 250 neurons, the second layer had 100 neurons, and the last layer used 150 neurons.

7.2 Approaches

7.2.1 Approach 1 - Contiguous Heart Rate Samples

To determine the most optimal and best-performing dataset and sample window prior to stress prompt creation, we tested the non-filtered dataset on the different sample ranges to observe the best performance on the Deep ECGNet. The idea behind using the non-filtered dataset for evaluating sample window significance is due to the data variance across the non-filtered dataset. We assume that this dataset will be the most noisy and variate, so model results on this dataset will help establish a solid baseline. Once the best performing sample range dataset is found on this model, the same sampled range will be leveraged for the other filtered datasets on other models.

Each dataset has rows of heart rate data and the corresponding label. Each machine learning model is designed to take in a 3-dimensional input array. To satisfy this design choice, each row of data from the dataset was processed into a 2-dimensional array, called a "Bucket of Samples". The shape of the training data is (_, X, 1), where _ is the number of rows, X is the number of samples in the row (this is determined

through the investigation below), and 1 is the outer dimension, called a "channel". The train and test data is split into 80% train and 20% test.

Model	Number of Samples	Accuracy	F1-Score
Deep ECGNet	60 Samples	76.80%	77.57%
	48 Samples	82.20%	82.01%
	36 Samples	80.13%	81.00%
	24 Samples	**84.13%**	**84.66%**
	12 Samples	79.22%	78.77%

Table 11. Approach 1 Investigation - Accuracy and F1-scores from the Deep ECGNET on Different Windows of Heart Rate Samples

Table 11 shows the baseline results using the Deep ECGNet. From this experiment, it becomes clear that on the non-filtered, contiguous **24** sample heart rate dataset, we see the highest performance of 84.13% accuracy and 84.66% F1-Score. The datasets used for testing on the specified models will leverage 24 samples (2 minutes) of data leading up to the stress prompt and will contain various forms of filtering.

1. Non-Filtered: 24 samples of heart rate
2. Hampel Filtered: 24 samples of heart rate
3. Median Filtered: 24 samples of heart rate
4. Savitzky-Golay Filtered: 24 samples of heart rate
5. Wiener Filtered: 24 samples of heart rate

For this approach, the input data for all three deep neural networks consists of a 24-sample vector, where each sample in the window is an input feature.

7.2.2 Approach 2 - Heart Rate Features

This approach is inspired by the success achieved by Sim et al. [28]. The authors leverage the same data as this study, but on a smaller scale and process the data into

features which are then fed into a simple feed forward neural network. Approach 2 tests the impact of this data processed into 7 different features from the heart rate data. The study uses 70 samples of heart rate data (∼5-6 minutes without contiguous 5-second samples). To keep consistent with the previous work, we also leveraged 70 samples prior to the stress prompt. Since there is some possibility of noise in our dataset, we begin by first processing the data into the different filtering techniques presented in the above sections. After the data is processed, we take the filtered data and extract 7 features from the dataset. As mentioned, to maintain consistency across this thesis and the paper, we leveraged the same features as well as same window size. Future work could make this data

1. *Resting HR*: Resting heart rate value for the day as calculated by Fitbit
2. *Mean*: The average of all the 70 heart rate samples
3. *Standard Deviation*: The standard deviation of all the 70 heart rate samples
4. *Maximum*: The maximum value of all the 70 heart rate samples
5. *Minimum*: The minimum value of all the 70 heart rate samples
6. *DiffResting*: The change in the resting heart rate from the mean value
7. *RMSSD*: the root mean squared of successive differences value to show the beat-by-beat variance

As mentioned in the prior section, this testing on both approaches is done on two dataset sizes, the full cohort, which consists of all 38 cadets throughout different cohorts in the study period, and a smaller dataset that is only comprised of one cohort with 15 cadets. Sim et al. [28] leveraged a single cohort dataset to achieve high performance on the simple feed forward network. The data used in the single cohort dataset for this thesis leverages the same 15 cadets as the Sim et al., ensuring consistency when comparing results.

7.3 Experimental Results

Approach	Model	Filtered Dataset	Accuracy	F1-Score	Accuracy	F1-Score
			Full Dataset		Single Cohort	
Approach 1 24 Samples of Contiguous Heart Rate Before the Stress Prompt	Deep ECGNet [24]	None	84.13%	84.66%	85.49%	85.83%
		Hampel	**84.87%**	**85.60%**	85.98%	86.80%
		Median	74.94%	76.67%	85.39%	85.88%
		Savitzky-Golay	83.40%	84.38%	**86.38%**	**87.01%**
		Wiener	83.97%	84.88%	85.29%	86.17%
	Feed Forward [28]	None	81.77%	82.90%	89.86%	90.27%
		Hampel	**82.10%**	**83.43%**	**90.66%**	**90.94%**
		Median	76.81%	77.29%	85.19%	85.29%
		Savitzky-Golay	79.66%	81.40%	89.76%	90.20%
		Wiener	80.31%	81.78%	88.07%	88.44%
	MLP	None	85.35%	86.15%	**87.77%**	**88.49%**
		Hampel	**85.92%**	**86.88%**	85.59%	86.07%
		Median	81.53%	81.38%	86.08%	86.27%
		Savitzky-Golay	85.92%	86.49%	85.69%	86.07%
		Wiener	84.62%	85.79%	85.49%	85.63%
Approach 2 Features Extracted from 70 Samples of Contiguous Heart Rate Before the Stress Prompt	Deep ECGNet [24]	None	**80.70%**	**82.01%**	**84.42%**	**84.52%**
		Hampel	78.82%	79.97%	84.22%	84.29%
		Median	78.74%	79.75%	83.62%	84.11%
		Savitzky-Golay	80.38%	81.65%	80.82%	81.10%
		Wiener	80.21%	81.72%	83.12%	83.51%
	Feed Forward [28]	None	**83.73%**	**84.27%**	85.71%	85.69%
		Hampel	80.38%	80.23%	**86.81%**	**86.77%**
		Median	80.87%	81.10%	86.51%	85.98%
		Savitzky-Golay	81.68%	82.80%	84.52%	83.46%
		Wiener	81.85%	82.66%	84.52%	84.07%
	MLP	None	83.16%	83.76%	**81.02%**	**82.41%**
		Hampel	80.54%	80.99%	77.92%	78.10%
		Median	81.36%	81.58%	77.92%	77.74%
		Savitzky-Golay	**83.65%**	**84.35%**	74.43%	75.19%
		Wiener	82.26%	81.90%	79.52%	79.19%

Table 12. Accuracy and F1-score Results on Two Different Data Processing Approaches on the Deep ECGNET, Feed Forward Network, and the MLP. The Results Are Separated into "Full Dataset" And "Single Cohort" Results Specifying How Much Data Collected from the Study Is Fed into the Training and Testing.

The accuracy and F1-score from the different experiments are shown in Table12. Both approaches were run on the three neural networks described above; Deep ECGNet, Feed Forward, and Multilayer Perception Network.

- **Approach 1:** 24 Samples of Contiguous Heart Rate Before the Stress Prompt
- **Approach 2:** Features Extracted from 70 Samples of Contiguous Heart Rate Before the Stress Prompt

For each approach, the models are tested on two different processed datasets. The "Full Dataset" consists of every user's collected data during the entire course of the study. The "Single Cohort" focuses on only the first cohort in the study, as similarly leveraged by Sim et al. [28].

7.3.1 Approach 1 Results

Both approaches take a unique method of data processing to represent the heart rate variations prior to the stress prompt. Results from the Approach 1 structuring technique show the efficacy of using simple input data consisting of 24 samples of contiguous heart rate data, where each sample is passed in as a feature. Across the three models and five filtered datasets in Approach 1, the best accuracies for both the "Single Cohort" and "Full Dataset" are bolded. The best performing filtering technique appears to be the Hampel-filtered dataset. The best accuracy achieved from Approach 1 on the "Single Cohort" dataset is 90.66% on the Feed Forward neural network using the Hampel-filtered dataset, proving its efficacy on the deep neural networks as a whole. On the "Full Dataset", the best accuracy is 85.92% accuracy on the MLP network using the the Hampel-filtered dataset. In terms of overall generalizability, the accuracy achieved on the MLP network (85.92%) is the most promising. It demonstrates the models' ability to scale to a larger variation of data. This was also the best accuracy achieved on the "Full Dataset".

One thing to note is that the MLP network performs better than the Deep ECGNet on the "Full Dataset". The Deep ECGNet has a complex LSTM architecture. The lower accuracy results from the Deep ECGNet may indicate that the temporal features and dependencies of the Approach 1 dataset may not be strong enough to require the

complex LSTM architecture. However, since the accuracy difference is not large, the difference may be attributed to architecture, hyperparameters, and other factors.

7.3.2 Approach 2 Results

Results from the Approach 2 data structuring technique leverage hand-crafted features extracted from 70 samples of heart rate. Across the best performing accuracy results, the best filtering technique comes from the Non-Filtered dataset. The best accuracy achieved on the "Single Cohort" dataset is an 86.81% accuracy from the Feed Forward Neural network on the Hampel-filtered dataset. The best accuracy from the "Full Dataset" is 83.73% accuracy, which also comes from the Feed Forward network on the Non-Filtered dataset.

This accuracy result (83.73%) is less than the best performing accuracy from the "Full Dataset" (85.92%). Approach 1 performs better than Approach 2. This is an indication to us that the hand-crafted features fed in as input in Approach 2 may not have been the most optimal. Further, extracting these hand-crafted features required a lot of manual overhead. Prior to the training and inference tasks, these datasets needed to be processed with the different feature extractors, which is time-consuming. Such manual overhead did not yield any accuracy improvements, proving to us that Approach 1 may have been more effective. Overall, with the two approaches, we can deduce that deep neural networks perform better with simple input data, where the models themselves can extract relevant features and do the feature engineering without user overhead.

Additionally, the difference between the two approaches is probably due to the most amount of relevant history prior to the stress prompt and the higher probability

that the user was wearing the watch for the full 2 minutes (24 samples) compared to the full 5-6 minutes (70 samples) captured in Approach 2. Psychology and biology also suggest that stress can happen instantaneously, in which case Approach 2 would initially capture unstressed heart rate data in a later stressful episode.

7.3.3 Dataset Size Results

Results from the experiments show that the "Single Cohort" dataset performs much better than the "Full Dataset". In each model and approach type, excluding the MLP for Approach 2, most of the accuracy and F1-score values sit around the mid- to high-80%s. This is much higher than the "Full Dataset" performance, which barely broke 80% in the case of both approaches. These results are probably attributed to the strong correlation of Fitbit wear time, response rates, and overall perception of the cadets on a cohort-specific level. Cohorts operated with a strong sense of community and camaraderie and most probably discussed or shared experiences while wearing and recording self-reported stress values. Across the "Full Dataset", the cohorts each had different scenarios and different levels of environmental stress (from their lead sergeants or other peers) which could have impacted the biometric measurements and self-report rates, leading to lower levels of generalizability.

7.3.4 Filtering Results

Further, the experiments suggest that some of the filtering techniques prove to be more effective for the machine learning models than others. The Wiener filter, which is designed for blurred inversion and additive noise removal, and the median filter,

for image smoothing, did not perform the highest in any experiment. However, the Hampel filter performed the best, preserving the relationships between the heart rate values and smoothing the data across the 24-sample viewing window. The Savitzky-Golay filter was also successful in maintaining relationships across the data values and producing acceptable accuracy scores. From these experiments, we can also see that the non-filtered data was also high-performing, indicating the possibility of processing and changing the non-filtered data for higher accuracy.

Overall, the Hampel filter performed the best on both approaches, but specifically the best performing approach, Approach 1. This is should be investigated in other contexts to prove its overall efficacy.

Chapter 8

CONCLUSION AND FUTURE WORK

In summary, this thesis presents a framework that uses real-world data and deep neural networks to detect stress in police cadets, with the eventual goal of better supporting police officers and law enforcement agencies. Through the use of various data processing strategies, this study successfully handles the naturalistic and unconstrained characteristics of the collected data. The data was then fed into carefully designed deep neural networks. The results from those models prove that deep neural networks can learn from heart rate data and detect stress in real-world scenarios. Further, the results from the full dataset with various characteristics show the models' ability to generalize. Overall, the thesis contributes to a growing domain of research aimed at improving the lives of police officers and enhancing the overall effectiveness of law enforcement agencies.

To summarize, this thesis presented the following takeaways:

- Processing heart rate data into a 2-minute window prior to stress level prompts is more effective than using extracted heart rate features as input to different deep neural networks.
- Hampel and Savitzky-Golay filters proved most effective at removing outliers in heart rate data. The data processed with these filtered achieved higher deep neural network performance than traditional filtering techniques.
- Deep neural networks exhibit better performance when trained on smaller datasets that are cohort-specific compared to larger, generalized dataset with all participants. The Feed Forward network achieved the highest accuracy of 90.66%

when trained on the smaller dataset that was treated with the Hampel filter, while the MLP network achieved 85.92% accuracy on the larger Hampel-filtered dataset.

8.1 Limitations and Implications for Future Work

This work presents some limitations and areas for future improvement. Firstly, the method of data collection presents challenges in terms of the quality of the ground truth. The stress scale is relatively arbitrary - what one cadet may perceive as 'level 4 stress', another may perceive as a 'level 1 stress'. Such discrepancies may be making it difficult for the model to generalize with higher accuracy. To address this, a possible goal would be to develop individualized models for each user. Such models would be trained on data from a specific user so that stressed and unstressed characteristics are relevant to the user.

Additionally, compliance rates from the participants may have affected the quality of the data. Through the data analysis, we see that certain cadets participated more in the data collection process, and several stress prompts were not responded to. We also saw that as the intensity of the academy training increased, the Fitbit wear time (and thus, the data collected) rapidly decreased. If we can find a way to improve the data collection experience, we may be able to improve compliance rates and data integrity. Another way to improve compliance could be to revisit the prompting algorithm to ensure that the prompts are non-disruptive but also simultaneously capture moments of stress.

Finally, this thesis presented a binary classification approach to stress detection. Using this approach may have muted some essential characteristics of the data. In

the future, it may be beneficial to move towards a multi-class problem. Addressing these limitations and incorporating suggestions for future work could lead to a more reliable and comprehensive stress detection model for real-world applications.

REFERENCES

[1] Abid Ali Awan. *The Machine Learning Life Cycle Explained*. Oct. 2022. URL: https://www.datacamp.com/blog/machine-learning-lifecycle-explained.

[2] Yang Bai et al. 'Comprehensive comparison of Apple Watch and Fitbit monitors in a free-living setting'. In: *PLoS One* 16.5 (2021), e0251975.

[3] Sándor Beniczky et al. 'Machine learning and wearable devices of the future'. In: *Epilepsia* 62 (2021), S116–S124.

[4] Noam Bressler. *How to check the accuracy of your machine learning model*. Mar. 2023. URL: https://deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model/#:~:text=Accuracy%20score%20in%20machine%20learning,the%20total%20number%20of%20predictions..

[5] Ayan Chatterjee and Uttam Kumar Roy. 'PPG based heart rate algorithm improvement with Butterworth IIR Filter and Savitzky-Golay FIR Filter'. In: *2018 2nd International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech)*. IEEE. 2018, pp. 1–6.

[6] Denise Chen. *Statistical learning: Data sampling amp; Resampling*. Apr. 2020. URL: https://towardsdatascience.com/statistical-learning-ii-data-sampling-resampling-93a0208d6bb8.

[7] Colomba Falcone et al. 'Rapid heart rate increase at onset of exercise predicts adverse cardiac events in patients with coronary artery disease'. In: *Circulation* 112.13 (2005), pp. 1959–1964.

[8] *Fitbit Development: Fitbit SDK*. URL: https://dev.fitbit.com/.

[9] Fuad A Ghaleb et al. 'Two-stage motion artefact reduction algorithm for electrocardiogram using weighted adaptive noise cancelling and recursive Hampel filter'. In: *PloS one* 13.11 (2018), e0207176.

[10] *Hampel*. URL: https://www.mathworks.com/help/dsp/ref/hampelfilter.html.

[11] Victoria Hodge and Jim Austin. 'A survey of outlier detection methodologies'. In: *Artificial intelligence review* 22 (2004), pp. 85–126.

[12] Katie Hoemann et al. 'Context-aware experience sampling reveals the scale of variation in affective experience'. In: *Scientific reports* 10.1 (2020), pp. 1–16.

[13] Insider Intelligence. *Wearable tech in Healthcare: Smart Medical Devices amp; Trends in 2023*. Jan. 2023. URL: https://www.insiderintelligence.com/insights/wearable-technology-healthcare-medical-devices/.

[14] Kerry Karaffa et al. 'Perceived Impact of Police Work on Marital Relationships'. In: *The Family Journal* 23.2 (2015), pp. 120–131. DOI: 10.1177/1066480714564381. URL: https://doi.org/10.1177/1066480714564381.

[15] Fanny Larradet et al. 'Toward emotion recognition from physiological signals in the wild: approaching the methodological issues in real-life data collection'. In: *Frontiers in psychology* 11 (2020), p. 1111.

[16] Russell Li and Zhandong Liu. 'Stress detection using deep neural networks'. In: *BMC Medical Informatics and Decision Making* 20 (2020), pp. 1–10.

[17] Serafeim Loukas. *Everything you need to know about min-max normalization in Python*. Mar. 2023.

[18] Gonzalo J Martinez et al. 'Alignment Between Heart Rate Variability From Fitness Trackers and Perceived Stress: Perspectives From a Large-Scale In Situ Longitudinal Study of Information Workers'. In: *JMIR human factors* 9.3 (2022), e33754.

[19] *Median Filter*. Nov. 2022. URL: https://en.wikipedia.org/wiki/Median_filter.

[20] *On-the-Job Stress in Policing— Reducing It, Preventing It*. URL: https://www.ojp.gov/pdffiles1/jr000242d.pdf.

[21] H. Douglas Otto and Alysson Gatens. *Illinois Criminal Justice Information Authority*. URL: https://icjia.illinois.gov/researchhub/articles/understanding-police-officer-stress-a-review-of-the-literature.

[22] Ronald K Pearson et al. 'Generalized hampel filters'. In: *EURASIP Journal on Advances in Signal Processing* 2016 (2016), pp. 1–18.

[23] Suzanne Pieper and Jos F Brosschot. 'Prolonged stress-related cardiovascular activation: Is there any?' In: *Annals of Behavioral Medicine* 30.2 (2005), pp. 91–103.

[24] Pooja Prajod and Elisabeth André. 'On the Generalizability of ECG-based Stress Detection Models'. In: *arXiv preprint arXiv:2210.06225* (2022).

[25] Andrea Ridolfi. *Smoothing Your Data with the Savitzky-Golay Filter and Python*. URL: https://blog.finxter.com/smoothing-your-data-with-the-savitzky-golay-filter-and-python/.

[26] Yuri Rykov et al. 'Digital biomarkers for depression screening with wearable devices: cross-sectional study with machine learning modeling'. In: *JMIR mHealth and uHealth* 9.10 (2021), e24872.

[27] *Share your health data with healthcare providers*. Sept. 2022. URL: https://support.apple.com/en-us/HT213359.

[28] S Sim et al. 'Exploring Edge Machine Learning-based Stress Prediction using Wearable Devices'. In: *21st IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2022.

[29] *Stress in America™ 2020: A National Mental Health Crisis*. URL: https://www.apa.org/news/press/releases/stress/2020/report-october.

[30] Andriy Temko. 'Estimation of heart rate from photoplethysmography during physical exercise using Wiener filtering and the phase vocoder'. In: *2015 37th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*. IEEE. 2015, pp. 1500–1503.

[31] Bharadwaj Veeravalli, Chacko John Deepu, and DuyHoa Ngo. 'Real-time, personalized anomaly detection in streaming data for wearable healthcare devices'. In: *Handbook of large-scale distributed computing in smart healthcare* (2017), pp. 403–426.

[32] *Visualize confusion matrix using caret package in R*. Jan. 2023. URL: https://www.geeksforgeeks.org/visualize-confusion-matrix-using-caret-package-in-r/.

[33] *Workplace Stress*. Feb. 2023. URL: https://www.stress.org/workplace-stress.

[34] Ping Yang, Guy A Dumont, and J Mark Ansermino. 'Sensor fusion using a hybrid median filter for artifact removal in intraoperative heart rate monitoring'. In: *Journal of clinical monitoring and computing* 23 (2009), pp. 75–83.

[35] Yuning Zhang, Maysam Haghdan, and Kevin S Xu. 'Unsupervised motion artifact detection in wrist-measured electrodermal activity data'. In: *Proceedings of the 2017 ACM International Symposium on Wearable Computers*. 2017, pp. 54–57.