PHOTOPLYTHESMOGRAM (PPG) SIGNAL RELIABILITY ANALYSIS IN A WEARABLE SENSOR-KIT

A Thesis

Submitted to the Faculty

of

Purdue University

by

Deena Alabed

In Partial Fulfillment of the

Requirements for the Degree

of

Masters of Science

May 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

Dr. Mireille Boutin, Chair

School of Electrical and Computer Engineering

Dr. Alexander L. Francis

School of Health and Human Sciences

Dr. Amy R. Reibman

School of Electrical and Computer Engineering

Dr. Michael D. Zoltowski

School of Electrical and Computer Engineering

Approved by:

Dr. Pedro Irazoqui

Head of School of Electrical and Computer Engineering

This thesis is dedicated to my parents; Mohammad and Nisreen,
who are, and always will be, my role models.

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my advisor, Prof. Mireille Boutin, for her constant support and guidance throughout my master's program. Thank you for your profound belief in my abilities and your valuable advice of both academic and personal natures. I am grateful to have had the chance to work with you and I could not have imagined having a better advisor.

I would also like to extend my deepest appreciation to my committee; Prof. Alex Francis, Prof. Mike Zoltowzki, and Prof. Amy Reibman for their constructive comments and insightful suggestions regarding my research as well as my thesis. The completion of my dissertation would not have been possible without the contributions of Prof. Amy Reibman, Biao Ma, William Sanchez, Chengzhang Zhong, Tiberius Wehrly, and Emma Bonham, who designed and built the sensor-kit. Tiberius also acquired and labeled part of the data. I would also like to thank Prof. Alex Francis for his invaluable insight into biosignals. In addition, I want to thank Tarun Yellamraju for his help with the Deep Neural Network, as well as Marie-Claire Herbig and Sarah Mues who helped inspire some of the methods we used.

To my friends in West Lafayette, Muna, Rana, Mai, Eman, Ahmed, Diala, Rami, Aniesh, Amruthavarshini, Tarun, Manish, Julius, Rakesh, Camilo, Thilo, Pushyami, Danielle, Nikhita, and Victoria, it was a privilege to become friends with such outstanding individuals from all around the world. I think we can all agree that there aren't many exciting things to do in West Lafayette, but this gave us more time to really get to know each other (as well as stress over grad school and mull over our life choices and what brought us here and so on and so forth). I sincerely hope our paths cross again in the future.

To my friends back home, I am blessed to have been surrounded by friends who are not only fun to be with, but are also ambitious people who lift each other up.

Souad, I don't think it's possible for you to fully understand how much our friendship means to me, and how much I appreciate you. Thank you for always being there for me. Bdour, nothing beats a deep, soul-searching conversation with you over a cup of tea. We haven't been able to do that in person lately, but it doesn't make our conversation any less special. Thank you for sharing this journey with me in all its ups and downs. Sami, your entertaining phone calls and magnificent singing abilities always managed to cheer me up and give me a break from the stressful life of a graduate student. Thank you for providing a sense of home in a country 10,000 Km away from home. My high school best friends, Angy, Haya, and Leen, thank you for being constants in my life whom I can always count on.

This acknowledgement would not be complete without mentioning my aunts, uncles and cousins who I am truly blessed to have in my life. And most importantly, my beloved grandmother for ensuring that I am warm in the cold Indiana winters with her hand-made knitted scarfs.

Last but not least, my family. Laith, thank you for having so much faith in my abilities and believing in me. Lama, sometimes it feels like you understand me more than I understand myself. Thank you for always encouraging me to aim high. My baby sister Layan who is not a baby anymore, you are growing up into a smart, compassionate, and beautiful young lady and I cannot wait to see what great things you do in life. Mama and baba, nothing I say will ever be enough to describe the love and respect I have for you. Everything I am today is because of you and I couldn't have done any of this without your love and support.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

MSC, Purdue University, May 2019. Photoplythesmogram (PPG) Signal Reliability Analysis in a Wearable Sensor-Kit. Major Professor: Mireille Boutin.

In recent years, there has been an increase in the popularity of wearable sensors such as electroencephalography (EEG) sensors, electromyography (EMG) sensors, gyroscopes, accelerometers, and photoplethysmography (PPG) sensors. This work is focused on PPG sensors, which are used to measure heart rate in real time. They are currently used in many commercial products such as *Fitbit Watch* and *Muse Headband*. Due to their low cost and relative implementation simplicity, they are easy to add to custom-built wearable devices.

We built an Arduino-based wearable wrist sensor-kit that consists of a PPG sensor in addition to other low cost commercial biosensors to measure biosignals such as pulse rate, skin temperature, skin conductivity, and hand motion. The purpose of the sensor-kit is to analyze the effects of stress on students in a classroom based on changes in their biometric signals. We noticed some failures in the measured PPG signal, which could negatively affect the accuracy of our analysis. We conjectured that one of the causes of failure is movement. Therefore, in this thesis, we build automatic failure detection methods and use these methods to study the effect of movement on the signal.

Using the sensor-kit, PPG signals were collected in two settings. In the first setting, the participants were in a still sitting position. These measured signals were manually labeled and used in signal analysis and method development. In the second setting, the signals were acquired in three different scenarios with increasing levels of activity. These measured signals were used to investigate the effect of movement on the reliability of the PPG sensor.

Four types of failure detection methods were developed: Support Vector Machines (SVM), Deep Neural Networks (DNN), K-Nearest Neighbor (K-NN), and Decision Trees. The classification accuracy is evaluated by comparing the resulting Receiver Operating Characteristic (ROC) curves, Area Above the Curve (AAC), as well as the duration of failure and non-failure sequences. The DNN and Decision Tree results are found to be the most promising and seem to have the highest error detection accuracy.

The proposed classifiers are also used to assess the reliability of the PPG sensor in the three activity scenarios. Our findings indicate that there is a significant presence of failures in the measured PPG signals at rest, which increases with movement. They also show that it is hard to obtain long sequences of pulses without failure. These findings should be taken into account when designing wearable systems that use heart rate values as input.

1. INTRODUCTION

1.1 Motivation

There has been an increase in demand for wearable health monitoring devices such as smart watches and fitness trackers in recent years. In 2016, the global wearable sensors market size was valued at $149.3 million and it is expected to reach $2.86 billion by 2025 [1]. Initially, these wearable sensors were mostly used for fitness tracking, for example tracking the number of steps. Now they have advanced to include heart rate, glucose level, and blood pressure measurements [2]. Some devices are also being used to gather information on particular diseases to aid in the diagnosis and/or management of these diseases such as sleep apnea, diabetes, and cardiovascular diseases [3].

Heart rate (HR) is one of the most valuable measurements. Traditionally, ECG was used for the purpose of measuring HR, but it is not easily implemented in wearable, wireless devices. Therefore, PPG sensors have emerged as an alternative to monitor HR [4]. PPG sensors are non-invasive sensors that optically detect the change in the volume of blood flow by emitting infrared light and measuring the changes in the intensity of the light reflected from the tissues and blood vessels. They are mainly used to estimate the HR since there is a correlation between the peak locations in the PPG and a cardiac cycle [5]. The possibility of using PPG signals for purposes other than HR estimation has been studied and looks promising. It has been shown that the second derivative of the PPG signal can give important health information [6] that could assist in the evaluation and diagnosis of various cardiovascular diseases [2]. PPG signals have also been used in stress level detection applications [7, 8].

Consequently, when building a low cost device that estimates the stress level in students, using PPG sensors in that device was the obvious choice. For that

purpose, we built an Arduino-based wearable wrist sensor-kit that consists of multiple affordable commercial biosensors. One of which is a PPG sensor. Analyzing the output PPG signal revealed that it does not always follow the typical PPG waveform. There are also instances where the signal is very noisy or saturated. Unfortunately, this could result in inaccurate heart rate estimation and stress level detection.

To solve this issue, we need a system that can automatically detect signal failures in order to discard them so they do not negatively affect the subsequent inferences (activity level, stress,...). We also need to understand why these failures occur and what causes them. Therefore, in this thesis, we analyze the PPG signal and design several methods that can automatically detect failures in the signal. Furthermore, we investigate how activity affects the amount of failures in PPG signals to offer some insight into the reliability of the PPG sensors when movement is introduced.

In the remainder of this chapter, we will describe PPG signals and their properties. We will also provide a review of previous work that has been done to detect artifacts in PPG signals as well as the challenges that arise when dealing with these signals.

1.2 Photoplethysmography (PPG)

PPG sensors are non-invasive sensors that are typically worn on fingers, earlobe, forehead, and wrist. They emit infrared light and measure the intensity of the light reflected from the tissues and blood vessels. This corresponds to the changes in blood volume. PPG sensors are low-cost and simple to implement in wearable devices, which is why they have emerged as an alternative for Electrocardiogram (ECG) to measure HR.

The acquired PPG signals consist of a high-frequency AC component superimposed on a slow-varying DC component. The DC component is controlled by respiration, whereas the AC component changes with the blood volume variations with each heartbeat [4]. This can be seen in the typical waveform of a PPG pulse (Figure 1.1), which consists of a systolic phase and a diastolic phase. The distance between two

consecutive systolic peaks (known as the peak to peak interval) represents a complete cardiac cycle [5], which is why PPG is used to estimate the HR.

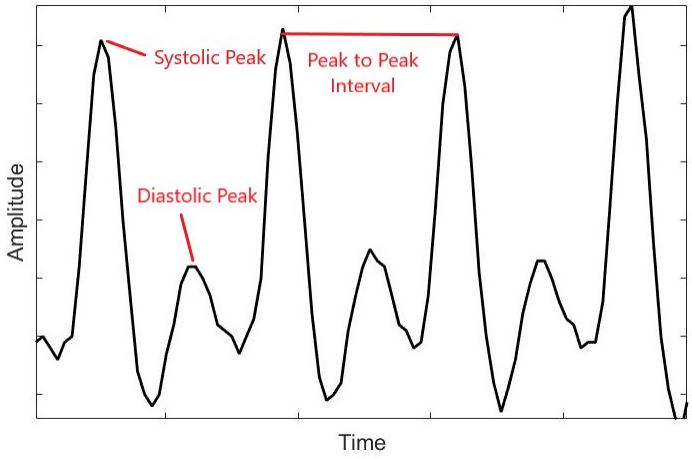


Fig. 1.1. Typical PPG waveform.

Besides estimating the HR, PPG can be used to estimate blood pressure (BP) [9,10]. In [11] the relative amplitude of the diastolic peak to the systolic peak is used to measure BP. In [9] three methods are implemented to detect the diastolic peak in the PPG waveform. The diastolic peak location is important to calculate the Pulse Transit Time (PTT) which is correlated with BP [12–14].

Since stress is highly associated with BP and HR, and both HR and BP can be estimated from several PPG waveform properties, PPG signals can be used to estimate stress levels [15–19].

1.3 Literature Survey

Although PPG sensors are low-cost and easy to implement, they are highly affected by motion and pressure disturbances [4]. There have been many attempts to reduce motion artifacts in corrupted PPG signals.

Multiple studies attempted to classify the signal by performing time-domain analysis on the pulse waveform morphology. A real-time segmentation and artifact detection based on the PPG waveform is proposed in [20]. The procedure is divided into six stages: first, clipped values from the raw signal are removed before performing any filtering. After that the signal is passed through a low-pass filter and a high-pass filter. In the fourth stage potential valleys and peaks are detected using adaptive thresholding, by applying a moving average filter with a span size of 75% of the last valid pulse waveform. For every window position, the maximum value is considered a potential peak and the minimum value is considered a potential valley. After that, diastolic peaks are discarded by checking if the vertical distance between the the peak and the valley are bigger than the previous valid pulse. When a complete pulse wave is detected several tests are performed such as rise time, systolic-to-diastolic duration ratio and number of diastolic peaks. Lastly, relative artifact detection is performed by comparing the rise time, duration and peak value of the current pulse with the previous pulse. This method resulted in a TPR of 99.6% and a FPR of 9.3%. It is worth noting that the number of disturbed pulses is estimated using the duration of undisturbed adjacent pulses. That is because the segments with failures might actually be composed of more than one heart pulse.

A similar algorithm is presented in [21] aiming to classify a pulse as "good" if a reliable heart rate can be obtained from it. The algorithm first detects peaks using a three-point peak detector. The second step is to calculate the HR from a 10 seconds window of the signal. The HR must be between 40 to 180 beats per minute. The algorithm then checks the maximum peak-to-peak duration is 3 seconds. If these checks are passed then template mapping is applied to classify each pulse. Template matching measures the correlation between pulses and decides the label based on how similar the pulses are. The resulting TPR is 91% with a FPR of 5%.

Another method is proposed in [22], where a minimum filter is used to find the lower envelope of the signal. This envelope is used to detect the troughs which helps in detecting diastolic peaks, since they are the maximum values between consecutive

troughs. After that, thresholding is applied on the pulse amplitude, trough depth difference and pulse width to find bad pulses. The algorithm performed poorly when the transition from good to bad in the PPG signal is gradual. The resulting mean TPR is 89% and mean FPR is 23%.

All of these studies are concerned with using the PPG signal to derive a reliable HR, and therefore they do not consider the presence of the diastolic peak. This is reflected in their labeling procedure. In addition, their data was collected in controlled lab environments using advanced sensors, whereas our data is collected using low-cost commercial sensors in a wearable device. There are other studies that collect data using a wearable device [5, 23] but their aim is to also enhance HR monitoring and not the reliability of the PPG signal itself.

2. WEARABLE SENSOR-KIT

Our sensor-kit, shown in Figure 2.1, is an Arduino-based device that consists of several biosensors attached to a wrist brace. It was designed and built for the purpose of measuring students' biometrics to estimate their stress levels. In order to do that, we needed a device that is small, light, low-cost, and reliable. A manual describing how to build the sensor-kit is available on the Purdue University Research Repository [24]. We summarize the instructions below.

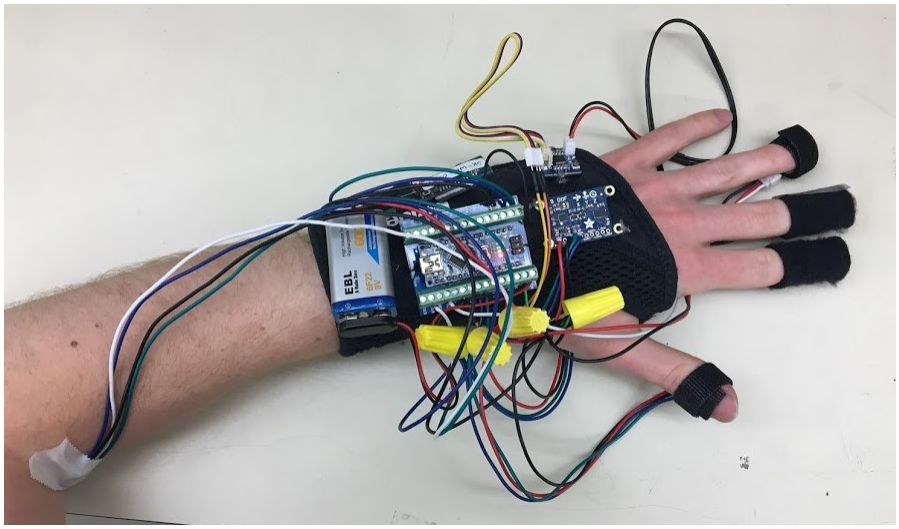


Fig. 2.1. Wearable wrist sensor-kit.

The bio-signals measured using the sensor-kit include skin temperature, skin conductivity, arm/hand motion, pulse rate and muscle intention. They are all measured using low-cost commercial sensors. The total cost of the sensor-kit is about $150. To build the sensor-kit, we utilized five types of sensors and connected them to a microcontroller. These components are all sewed on a wrist brace and need a 9V Lithium-ion battery to function. The components used are the following:

1. **Microcontroller:** Longruner Mini Nano Microcontroller Board Module for Arduino,

2. **Bluetooth Module:** DSD Tech Blurtooyj Serial Pass-through Module HC-05,

3. **Gyroscopic and Temperature Breakout Board:** Adafruit 9-DOF Accelerometer/ Magnetometer/ Gyroscope and Temperature Breakout Board, this is referred to as GYRO Sensor in the schematic diagram in Figure 2.3,

4. **Terminal Adapter Shield Expansion Board:** Aideepen 5pcs Nano Screw Terminal Adapter Shield Expansion Board Nano V3.0 AVR ATMEGA328P-AU Module for Arduino,

5. **Temperature Sensor:** Adafruit MCP9808 High Accuracy 12C Temp Sensor Breakout Board. Two such sensors are used. One is placed on the thumb (Thumb Temp), and one on the palm close to the base of the thumb (Palm Temp),

6. **Pulse Sensor:** Asiawill Pulse Heart Rate Sensor Module for Arduino, which is placed on the ring finger,

7. **Galvanic Skin Response (GSR) Sensor:** Seeed Grove GSR Sensor, which is placed on the index and middle fingers,

8. **Gyrocsopic Sensor:** Adafruit LSM9DSO FLORA 9-DOF Accelerometer/ Gyroscope/ Magnetometer, which is placed on the forearm and referred to as FLORA in the schematic diagram in Figure 2.3.

2.1 Building the Sensor-kit

To build the sensor-kit, the following items are needed in addition to the sensors mentioned before:

1. Battery Connector,

2. Hook-Up Wire Spool Set,
3. Wrist Brace,
4. Velcro Strap,
5. Adhesive Velcro Patches,
6. Soldering Iron and Solder,
7. Needles and Threads,
8. Scissors and Wire Cutting/Crimping/Stripping Tools.

The Arduino is first connected to the expansion board by soldering and then the expansion board is sewed onto the wrist brace. The gyroscopic and temperature breakout board, the GSR sensor, and the Bluetooth module are also sewed onto the wrist brace as in Figure 2.2.

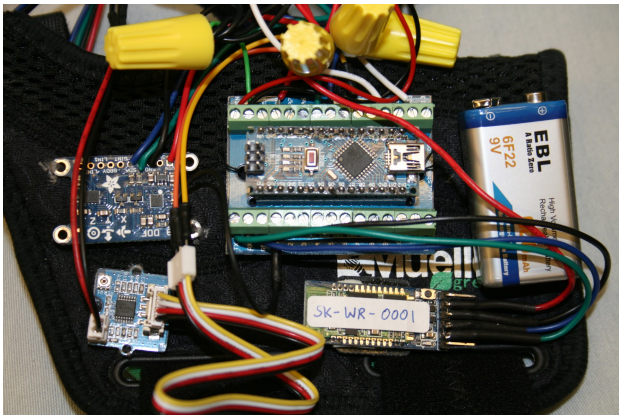


Fig. 2.2. A closer look at the component placement on the wrist brace.

The pulse sensor is sewed onto a Velcro strap to allow for adjustable pressure on the finger. The thumb temperature sensor is also attached to a Velcro strap. The palm temperature sensor as well as the FLORA gyroscope sensor are taped on the skin of the person wearing the sensor-kit.

The connections between the Arduino and the sensors can be seen in Figure 2.3. The details on how to set up the Arduino-Bluetooth-PC connections can be found on the project's website [25], as well as the construction manual [24].

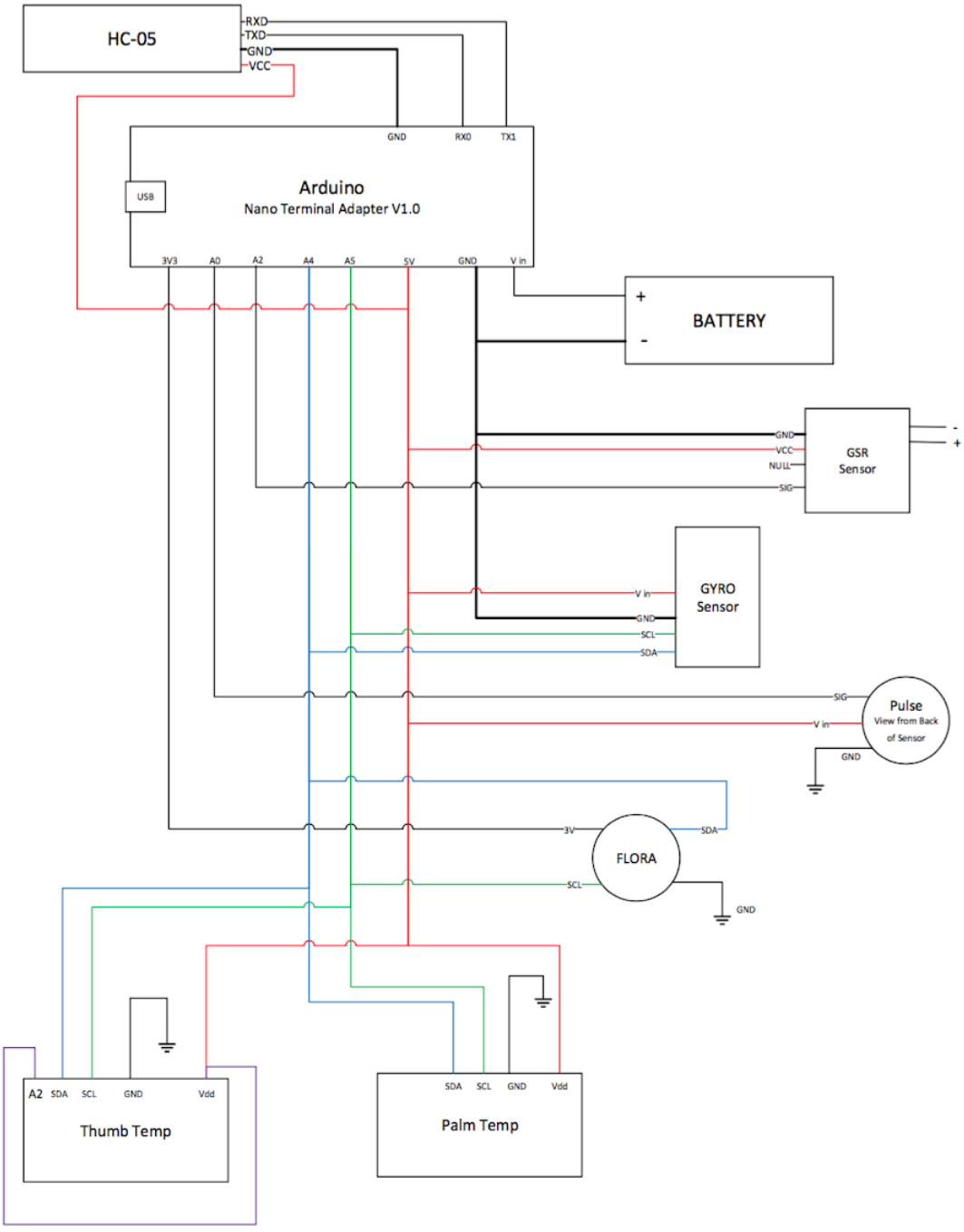


Fig. 2.3. Sensor-kit schematic diagram.

The code that initializes the data acquisition process and reads the output of the sensors and sends it to the laptop via Bluetooth was written in Python. It is available, along with the Arduino code, at [24].

2.2 Extracting Raw PPG Signal

In this study we are interested in the PPG signal. We therefore wrote a code that can extract the raw PPG signal from the output signal of the pulse sensor.

Some examples of the PPG signals that were collected using the sensor-kit can be seen in Figures 2.4 and 2.5. We can see that there are areas where the signal follows the waveform of a typical PPG signal. However, some areas are very noisy as in Figure 2.4. And others are saturated as in Figure 2.5. Some of the possible causes for this are pressure and motion as PPG signals are highly affected by motion and pressure disturbances [4].

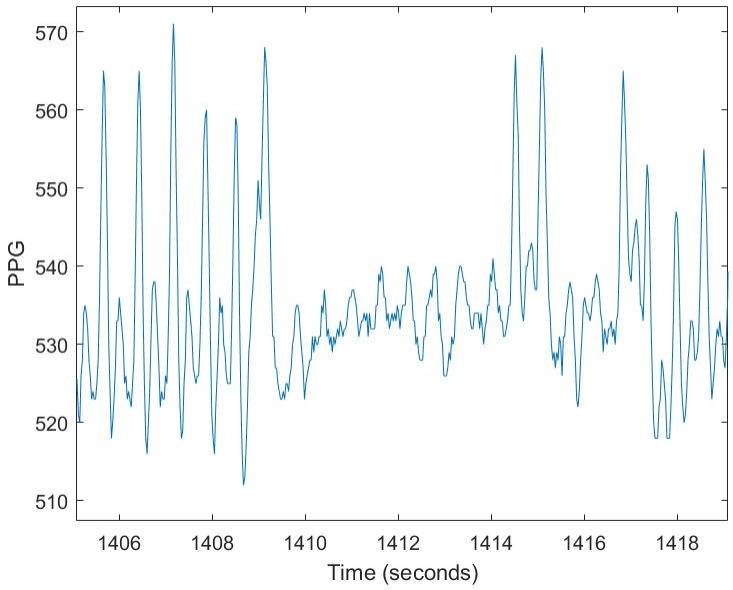


Fig. 2.4. Example of a noisy PPG signal measured with the sensor-kit.

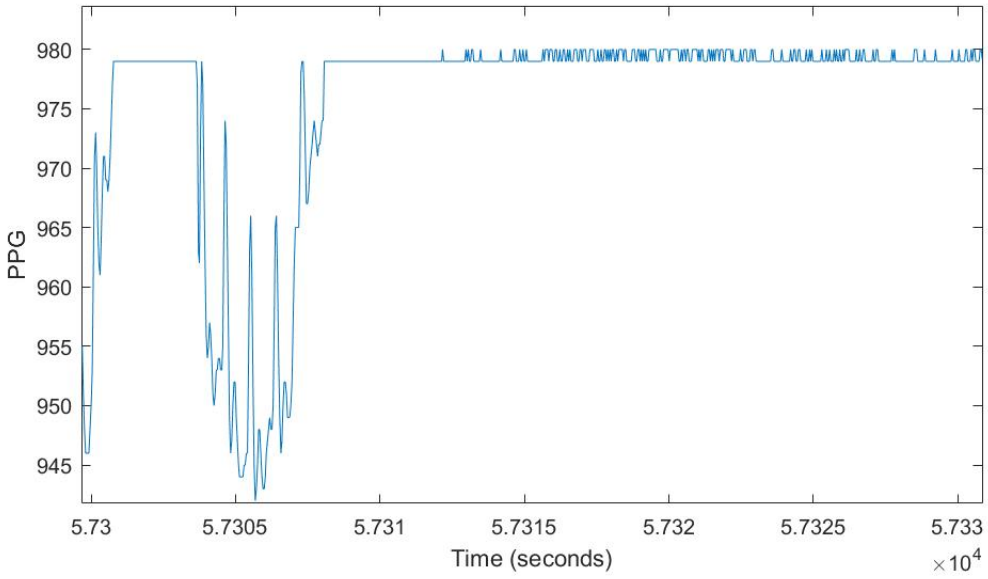


Fig. 2.5. Example of a saturated PPG signal measured with the sensor-kit.

These failures in the signal make it hard to detect the systolic and diastolic peaks, which negatively effects the accuracy of the stress level detection. Therefore, we need to design a failure detection method that can automatically find the failures in the PPG so they can be removed and would not effect the stress estimation.

3. METHODOLOGY

In order to design a method that can detect failures in PPG signals, we first need to analyze the signals and identify the characteristics that make failures distinguishable. We do this by first collecting data and manually labeling it to build a ground truth and gain an insight on the signal properties. Then we implement a method to segment the signal into windows that correspond to individual heart pulses. After that, we calculate some features that describe the geometry of each window. Lastly, we build four methods that take either the features or the raw PPG signal as input, and classify the pulses as either "Good" or "Bad".

Our second goal is to better understand the reason failures occur, specifically, the effects of movement on the signal. We therefore collect data that differs in the level of activity and use the previously proposed methods to track the performance of the PPG sensor in different activities.

This process is explained in details in the following sections.

3.1 Data Acquisition

This section gives a detailed description of the experimental setup and procedures used in data acquisition. In order to develop and test our failure detection methods, the wrist sensor-kit (Chapter 2) was used to collect biosignals. Recall that the Pulse Sensor was placed on the ring finger of the left hand. Care was taken to make sure it is secure but not too tight as to not ruin the signal.

The signals were collected in two settings. In the first experiment we had nine healthy participants. They were asked to sit still while wearing the sensor-kit on their left hand for about 30 minutes. The data from this experiment is manually labeled

and used to train and test the error detection classifiers. The collected signals as well as the labels are available at [26].

In the second experiment the data was collected from five participants in three scenarios that vary in the level of activity. In the first scenario, the participants were asked to sit as still as possible while wearing the sensor-kit on their left hand and resting it on a table. In the second phase they were asked to talk about a topic that they already know and are passionate about. The purpose is to introduce motion artifacts from communicative gestures. Lastly, the participants were asked to walk to introduce motion artifacts from movement. Each phase was about 10 minutes long. The data collected from this experiment is used in evaluating the reliability of the sensor-kit and is available at [27]. Note that we did not use this dataset in any way during the development of our failure detection methods so as to not affect its independence.

3.2 Signal Preprocessing and Labeling

As explained in Section 2.2, we extracted the raw PPG signals from the measurements of the Pulse Sensor. A window of the measured signal is shown in Figure 3.1. In order to create a ground truth and understand the properties of the signal, we had to assign labels to the acquired PPG pulses. We only labeled the signals that were acquired in the first experiment, in which nine subjects were sitting still.

Recall from Section 1.2 that a typical PPG pulse waveform consists of two peaks: systolic peak, which has a higher amplitude, and a diastolic peak (Figure 1.1). The interval between two systolic peaks represents one heart pulse.

Different labeling techniques were applied in previous work. Some studies classified the pulses into three categories. In [28], the signals were categorized into “Excellent” when the systolic and diastolic peaks are both distinct, “Acceptable” when the systolic and diastolic waves are not very clear but the heart rate can be determined, and “Unfit” otherwise. On the other hand, in [22], the pulse is “good” when it has the

regular waveform and a similar amplitude, width and morphology to adjacent pulses, "Poor" when a pulse has a width of a "good" pulse and corresponds to a peak in the ECG signal but the amplitude and/or morphology are slightly different, and "bad" otherwise.

In this study, we used a simpler labeling approach and classified the signal as "good" or "bad" based on its shape. A "good" pulse is when both systolic and diastolic peaks are clear as in Figure 1.1. In that case we give the pulse the label "1". Otherwise, if either of the peaks is missing or if the signal is very noisy, then the pulse is given the label "0".

In order to label the signal, we had to manually study the raw PPG signal and assign the appropriate label. Deciding whether a pulse is "good" or "bad" was not always straightforward as the signal label was ambiguous in certain cases. An example of the measured raw PPG signal can be seen in Figure 3.1. The signal here is mostly "good". The signal in box (a) was labeled as "bad" since it is missing the diastolic peak. On the other hand, the pulse in box (b) has two small peaks after the systolic peak and therefore, it could be argued that it is "good"; since it does have the two peaks we are looking for. But it could also be considered "bad" since it has an extra peak. We decided to assign the label "good" to this type of waveform.

3.3 Signal Segmentation

Our main goal is to detect failures in the PPG signal. In order to do that, we need to divide the raw PPG signal into individual heart pulses. In this section, we will explain the signal segmentation technique that we developed and implemented.

A complete cardiac cycle is closely correlated with the distance between two consecutive systolic peaks in a PPG signal [5]. Therefore, to segment the signal into individual pulses we need to find the systolic peaks location. To find the locations of systolic peaks, the PPG signal is first filtered using a Binomial Low-pass filter

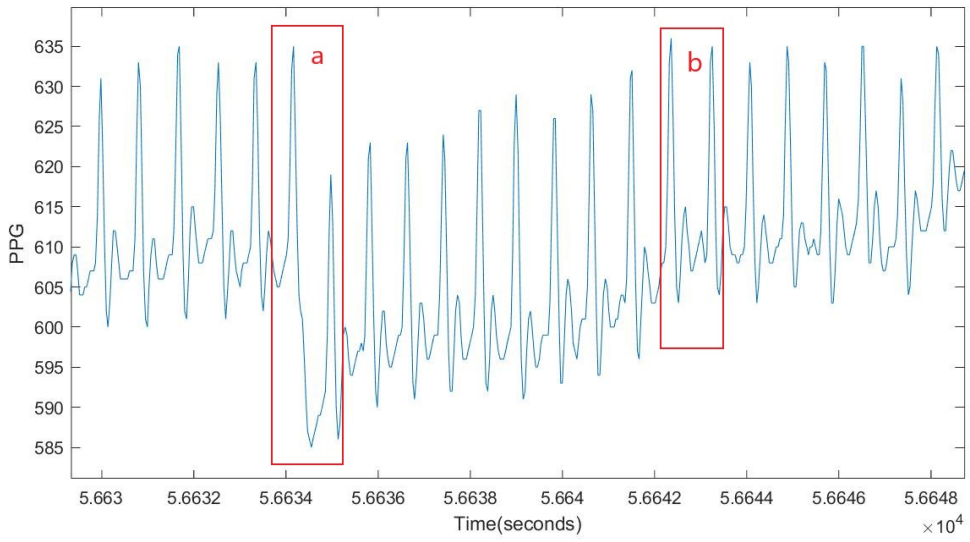


Fig. 3.1. A section of measured PPG signal.

to remove as much of the noise as possible. Then, using Matlab's built in function (FindPeaks), we find all the peaks in the signal.

The peaks found in the previous step include the systolic peaks, diastolic peaks, as well as any peaks caused by noise. Since we are only interested in finding the systolic peaks, we need to find a way to differentiate them from any other peaks.

To do that, we take a window of length w of the set of peaks previously found. We then model their amplitude distribution as a mixture of two Gaussian distributions $p(x)$ where x is the amplitude of the peak, as follows:

$$\begin{aligned} p(x) &= G_1(x) + G_2(x); \\ G_1(x) = \pi_1 e^{-(\frac{x-\mu_1}{\sigma_1})^2}, &\quad G_2(x) = \pi_2 e^{-(\frac{x-\mu_2}{\sigma_2})^2}. \end{aligned} \tag{3.1}$$

The first Gaussian $G_1(x)$ represents the diastolic peaks (lower amplitude), while the second Gaussian $G_2(x)$ represents the systolic peaks. Our goal now is to find the decision boundary that leads to the minimum Bayes error [29].

This is done by finding the optimal amplitude value that separates the two distributions. This value is the point where the two distributions intersect. To find this threshold T, we simply equalize the two distributions and solve the resulting quadratic equation (Equation 3.2) to find the point of intersection that lies between the mean values of the distributions μ_1 and μ_2:

$$(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_2^2})x^2 + 2(\frac{\mu_2}{\sigma_2^2} - \frac{\mu_1}{\sigma_1^2})x + (\frac{\mu_1}{\sigma_1^2} - \frac{\mu_2}{\sigma_2^2} + ln\frac{\pi_1}{\pi_2}) = 0. \tag{3.2}$$

After a threshold is found such that $\mu_1 < T < \mu_2$, the peak values within that group are compared to T. The peak is considered a systolic peak if it is larger than the threshold.

In order to pick a window length w, we tried multiple values and studied the resulting pulses. We found that $w = 6$ resulted in the best segmentation. So this is the value we used to separate the pulses. This segmentation procedure was implemented using Matlab and the code is available at [30].

3.4 Feature Extraction

For each PPG pulse vector $S \in \mathbb{R}^N$, twelve time-domain features were calculated:

1. Pulse Average Amplitude = $\mu = \frac{1}{N}\sum_{i=1}^{N} S_i$;
2. Norm Entropy = $\sum_{i=1}^{N} ||S_i||^2$;
3. Peak-to-Peak Interval = Length of Segment = N;
4. Systolic Peak Amplitude = $P_1 = S(1)$;
5. Diastolic Peak Amplitude = P_2;
6. Difference in Peaks Amplitude = $P_1 - P_2$;
7. Empirical Variance = $v = \sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N} |S_i - \mu|^2$;

8. Empirical Skewness $= s = \sum_{i=1}^{N} \frac{(S_i - \mu)^3}{\sigma^3}$;

9. Empirical Kurtosis $= k = \sum_{i=1}^{N} \frac{(S_i - \mu)^4}{\sigma^4}$;

10. Median $= \begin{cases} S(\frac{N+1}{2}), & \text{N is odd,} \\ \frac{S(\frac{N}{2}) + S(\frac{N}{2}+1)}{2}, & \text{N is even;} \end{cases}$

11. Minimum Amplitude Value within a Pulse $= \min_{i=1,\ldots,N} S(i)$;

12. Number of Peaks within a Pulse.

3.5 Training & Testing Datasets

For the purposes of developing and testing the failure detection methods, we only used the data from the first experimental setting (Section 3.1). The PPG signal from the nine participants was extracted and labeled. Then we used the segmentation technique from Section 3.3 to find separate pulses and extract 12 features from each pulse.

The data from five participants was merged and used for training the classifiers. To test the performance of the classifiers, we need to have a scenario similar to what we would have in a real-life application. In other words, we need to test on individual participants since PPG signals are subject dependant. Therefore, in the testing phase, we tested on a dataset that has the features of the four remaining participants as well as on each of the participants separately. The duration of the train and test datasets as well as the number of detected pulses and percentage of failure are shown in Table 3.1.

From the table, we can see that the percentage of failures varies from one subject to the other. Subjects 3 and 4 for example barley have any failures, whereas subject 1 has 13% and the training set has 9%. We can also see that the failure rate is much below 50%. This means that we have an issue of class imbalance.

Table 3.1.
Training and testing datasets

	Duration (minutes)	Number of Pulses	Percentage of Failures
Train Set	163	10678	9.2%
Test Set	106	6798	6.5%
Test Subject 1	37	2680	13.3%
Test Subject 2	23	1470	5.2%
Test Subject 3	23	1171	0.3%
Test Subject 4	22	6798	0.3%

3.5.1 Dealing with Class Imbalance

When dealing with class imbalance, the rules that describe the minority class are often fewer than those describing the majority class since the minority class is underrepresented [31]. Several techniques have been proposed in previous studies to deal with class imbalance in learning problems.

One of the most popular methods to minimize the effect of imbalanced data sets is sampling. Sampling consists of modifying the dataset to produce a balanced distribution. Even though there has been some conflicting opinions on the extent of performance improvement that a sampled balanced dataset provides compared to an imbalance dataset, for most applications sampling techniques do improve the classification accuracy [31].

Variations of sampling were applied in many studies. Although in many cases they improved the accuracy, they still suffered from some drawbacks. Undersampling instances of the majority class can leave out important instances that provide important differences between the two classes. On the other hand, oversampling instances of the minority class can lead to model overfitting since it produces duplicates [32].

To overcome overfitting, an oversampling technique called SMOTE (Synthetic Minority Oversampling TEchnique) was developed [33] to generate synthetic examples in the feature space instead of the data space. The technique finds a synthetic example along the line joining an instance in the minority class and all/some of its K-nearest neighbors. The number of neighbors depends on the desired class balance. This is

repeated for all the instances in the minority class. This process is demonstrated in the flowchart in Figure 3.2.

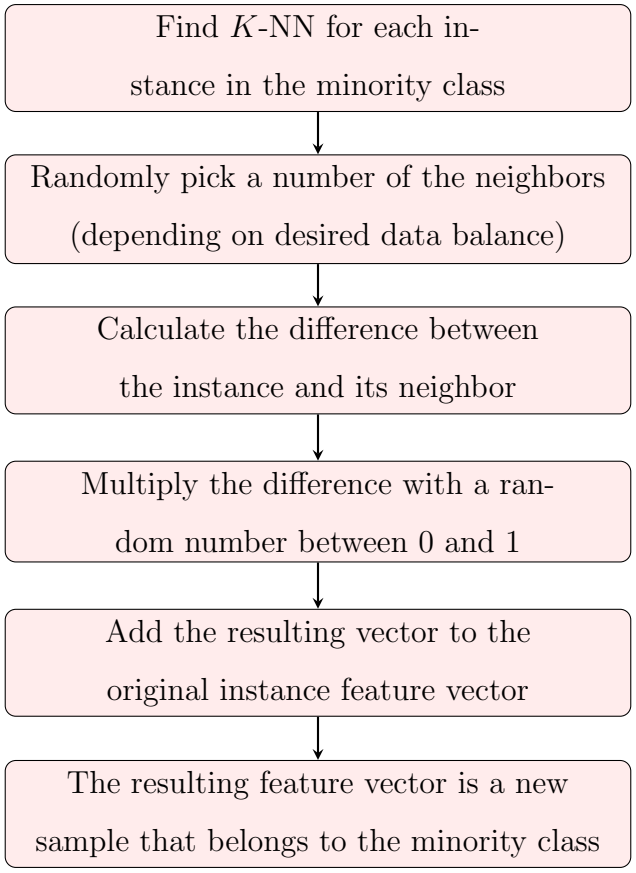


Fig. 3.2. SMOTE technique to deal with class imbalance.

We applied SMOTE technique with $K = 10$ to create a balanced training feature space (The modified data is available at [30]). The percentage of error in the training feature set, which was initially 7%, increased to 47.6%. The proposed classification methods were trained once using the original feature set, and once using the over-sampled feature set to examine the difference in performance.

3.6 Failure Detection Methods

We applied four machine learning methods to build classifiers that detect PPG signal failures. In this section, the methods and our implementation procedure are explained.

3.6.1 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning technique used for classification and regression. An SVM classifier tries to separate data into two categories divided by a clear gap that is as wide as possible.

Given training data (x_1, y_1),...,(x_n, y_n), where $x_i \in \mathcal{R}^p$ is a p-dimensional feature vector, and $y_i \in \mathcal{R}_1$ is the target output, we want to find the optimal hyper-surface with the largest margin that separates the instances for which $y_i = 0$ from the instances where $y_i = 1$.

For linearly separable data, as in Figure 3.3, the hyper-surface is described by a line equation:

$$\mathbf{w}^T\mathbf{x} + b = 0.$$

The instances above the boundary have label "1" and the ones below it will be labeled with a "-1".

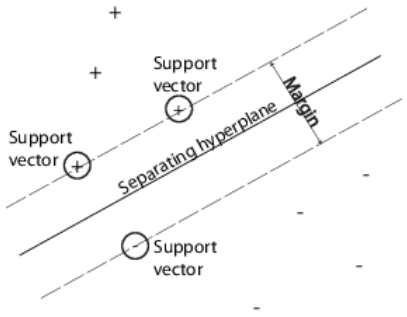


Fig. 3.3. SVM classifier [34] defined by a hyper-plane.

For data that is not linearly separable, a kernel is used to transform it into a higher dimensional space and then try to find the optimal linear boundary. This approach is referred to as "kernel trick" [35]. Some of the typically used kernels are: Polynomial, Sigmoid, and Gaussian (Radial Basis Function). Lastly, when the data is not separable (in both the original and the higher dimensional space), slack variables are introduced to allow some misclassification to occur.

Implementation

In our implementation of SVM, we use a non-linear Gaussian kernel. Therefore, the decision boundary is non-linear and can take any shape. This can cause a risk of overfitting the training data. Overfitting means that the obtained model fits the training data too well, which negatively impacts the models ability to generalize on testing data. This issue is more prominent in high-dimensions.

Therefore, we did not use all the twelve calculated features (Section 3.4) but rather, used three combinations of them. These combinations are the following:

1. SVM 1: Peak-to-Peak Interval, Difference in Peaks Amplitude, Systolic Peak Amplitude

2. SVM 2: Peak-to-Peak Interval, Difference in Peaks Amplitude, Number of Peaks within a Pulse

3. SVM 3: Peak-to-Peak Interval, Pulse Average Amplitude, Variance, Number of Peaks within a Pulse

As mentioned in Section 3.5, each of the SVM models was trained twice for comparison; once on the original feature set and once using the balanced feature set. And so we end up with 6 SVM classifiers.

We applied 10-fold Cross Validation (CV) in the training process. The observations were split into 10 partitions and the model was trained on 9 partitions and tested

on the 10^{th}. The error (misclassification rate) is then calculated. This is repeated for all 10 partitions and the resulting error is averaged.

To find the best SVM hyper-parameters that minimize the CV error, Bayesian Optimization was implemented. We then used the obtained parameters to build the SVM model.

In the testing phase, we apply the obtained SVM model to the testing data. The output is a $n \times 2$ matrix, where the first column is the probability that instance i belongs to class "0", and the second column is the probability that it belongs to class "1". Using these probabilities, we calculate the score (Equation 3.3) and use it to decide the predicted class.

$$\text{score}_i = \frac{P(\hat{y}_i = 0)}{P(\hat{y}_i = 1)} \tag{3.3}$$

Training and testing SVM models was done using Matlab. The code is available at [30].

3.6.2 K-Nearest Neighbor

K-Nearest Neighbor (K-NN) is a supervised machine leaning method that is used in classification problems. It classifies an object based on the class labels of its neighbors.

To classify an instance in the test dataset, the algorithm calculates the distance between the instance and each training sample. It then examines the labels of the K samples with the smallest distance (neighbors) and assigns the instance to the class that is most common among these neighbors. Any measure of distance can be used for this purpose, but the Euclidean distance is the most common.

Implementation

In our implementation of K-NN, we calculate the Euclidean Distance between each instance in the testing feature set and each instance in the training feature set.

Let $\vec{p}$ be a 12×1 feature vector in the test dataset such that $\vec{p} = \begin{bmatrix} p_1 & p_2 & ... & p_{12} \end{bmatrix}^T$ and $\vec{q}$ be a 12×1 feature vector in the train dataset such that $\vec{q} = \begin{bmatrix} q_1 & q_2 & ... & q_{12} \end{bmatrix}^T$, then the Euclidean Distance D is given by:

$$D(\vec{p}, \vec{q}) = D(\vec{q}, \vec{p}) = \sqrt{\sum_{i=1}^{12} (p_i - q_i)^2}$$

We set two values for K: 5 and 10. As mentioned in Section 3.5, each of the K-NN models was trained twice; once on the original feature set and once using the balanced feature set. In total, we end up with 4 K-NN classifiers.

After calculating the distance and obtaining the labels of the K nearest neighbors, we find the number of neighbors in class "0" and divide it by the number of neighbors in class "1". This gives us a measure that is similar to the score in the SVM (Equation 3.3). We can then use this score to determine the predicted class.

The code was implemented in Matlab and is available at [30].

3.6.3 Deep Neural Network

A Neural Network is a an interconnected group of nodes (neurons), loosely inspired by the human nervous system, that learns to perform a certain machine learning task by analyzing labeled training samples. Neural Networks are divided into layers, each layer contains a number of interconnected neurons as in Figure 3.4. The first layer is an "input layer", followed by a "hidden layer", and finally an "output layer". When there are multiple hidden layers, the network is called a Deep Neural Network (DNN).

Besides the neurons, a DNN consists of weighted connections between the neurons. Data travels between neurons via the connections based on the assigned weight. A weight of zero indicates no connection [37]. In a feedforward network, random weights are assigned to the connections between the neurons. Each neuron has an activation function which determines whether the neuron should be activated or not. This is done by calculating a weighted sum of the inputs and comparing it to a threshold.

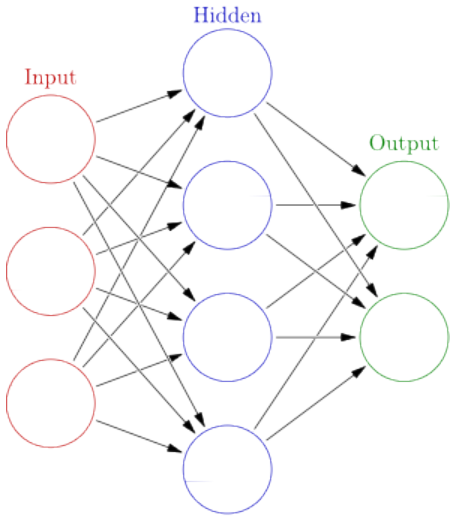


Fig. 3.4. A typical Neural Network structure [36].

Examples of activation functions are the hyperbolic tangent, sigmoid function, and rectified linear unit.

The output of the neuron is a number between 0 and 1. The output of the last layer represents the probability of the instance belonging to a class.

Implementation

In our implementation of DNN, we have 8 fully connected layers with a rectified linear unit (ReLU) activation function.

As mentioned in Section 3.5, each of the DNN models was trained twice; once on the original feature set and once using the balanced feature set. In the testing phase, the obtained DNNs are applied on the testing datasets. The output is the score (Equation 3.3) which is compared to a threshold to determine the class of each instance.

The code was implemented in Python using Keras and is available at [30].

3.6.4 Decision Tree

The Decision Tree method was hand-crafted after studying the shape of the PPG signals that we have. It was created to mimic the process that we were using to decide if a pulse is "good" or "bad". We tried to pinpoint the properties of the signal that led to us labeling it as "bad" and then find a way to quantify these properties.

The Tree consists of multiple stages. Each pulse goes through certain check points and either passes to the next stage or is classified as "bad". Essentially, we try to remove the pulses that we are certain are "bad". We check the duration of the pulse, the number of diastolic peaks, and the amplitude of the diastolic peak compared to the systolic peaks.

The input to the tree is the raw PPG signal and not the features set as in the other classification methods. The process first begins by applying the clustering function that we previously implemented when segmenting the signal (Section 3.3). This function is used to find the systolic peaks in the signal. While using it to segment the signal, we noticed that in the sections where the signal is very noisy and has a lot of bumps, different window sizes lead to a detection of different peaks. Whereas when the signal is "good", it does not matter what the window size is because clustering always detects the same peaks.

Therefore, in the Decision Tree, we use the clustering function to find the peaks using two window sizes and then we compare the resulting peaks. The window sizes are set to six and ten.

Figure 3.5 describes the process at a high-level. A more detailed explanation of the algorithm is provided in the steps below as well as the flowchart in Figure 3.6.

Step 1: Find systolic peaks using Clustering function (Section 3.3) with window sizes 6 (resulting in an ordered set of peaks $P6$) and 10 (resulting in an ordered set of peaks $P10$).

Step 2: Going through each pulse, if either of the following criterion are met, then the predicted label is "0" (bad):

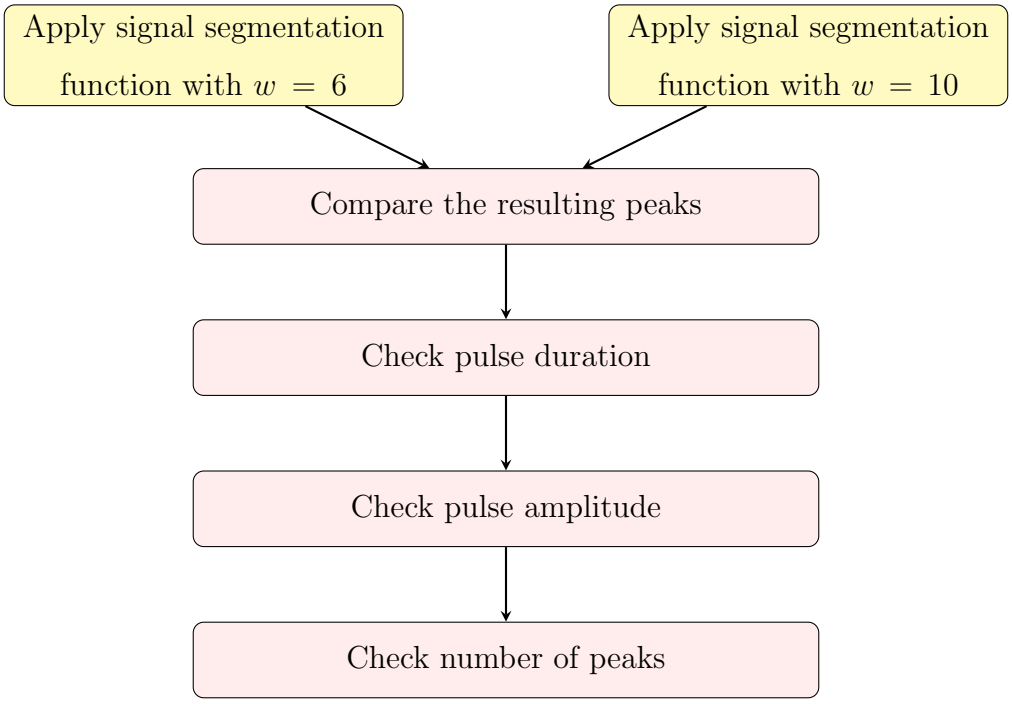


Fig. 3.5. High-level description of our Decision Tree classification steps.

(a) if the two systolic peaks in $P6$, that correspond to the current pulse, are not also found in $P10$. $P6(i)$ in the flowchart denotes an element (a peak) in the list $P6$.

(b) if pulse duration (D) is less than 0.5 seconds or larger than 1.5 seconds.

Step 3: If the pulse passes the checks above, then we find the number of peaks within the pulse (this function is represented in the diagram with "PK(x, y)", where x and y are the two systolic peaks that form a pulse). If only one peak (p) is found and its amplitude is not larger than either of the systolic peaks, then the predicted label is "1" (good). If, however, its amplitude is larger than the systolic peaks, then it is labeled "0".

Step 4: If there is more than one peak in the pulse, then it might be the case that when clustering with window size 6 a peak was missed. So we check if there are systolic peaks found in $P10$ that are between the two current peaks. In the flowchart, "Pulse(x, y)" denotes the pulse that starts with systolic peak x and ends at y. $P10(j)$ denotes an element (a peak) in the list $P10$. If this is indeed the case, then we actually have two pulses and we need to again check the number of peaks within the pulses. And accordingly decide if its "good" or "bad". If that is not the case, then the pulse is "bad".

The Decision Tree was implemented in Matlab. The code is available at [30].

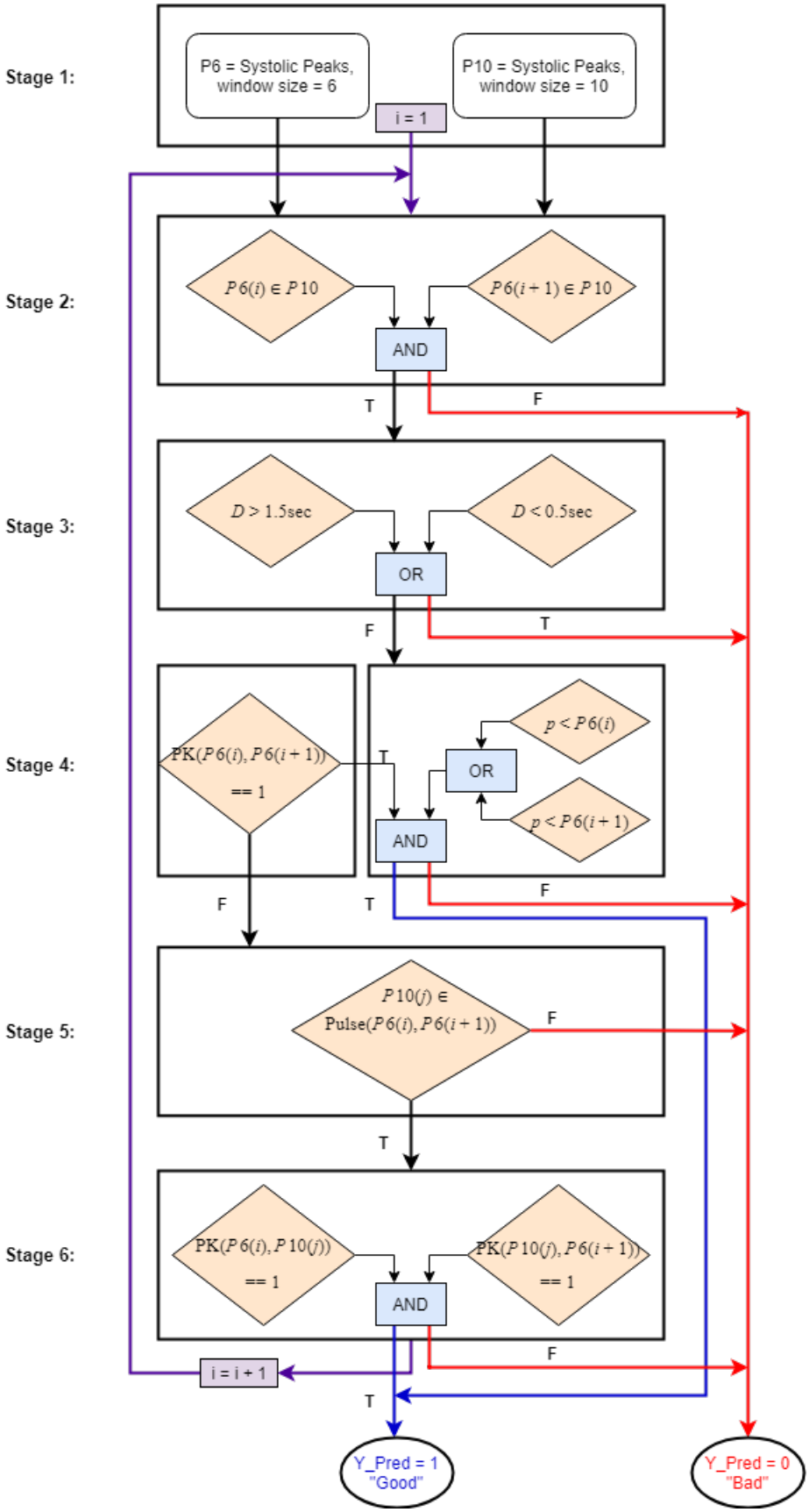


Fig. 3.6. Decision Tree classifier flowchart.

3.7 Evaluation Methods

3.7.1 Automated Failure Detection Method Evaluation

In order to evaluate the performance of the failure detection classifiers that we have implemented, we apply several evaluation techniques.

ROC curves and AAC

When classifying an instance in a 2-class classification problem, simply calculating the classification accuracy (the number of correct predictions divided by the total number of predictions) is not enough. There are four possible outcomes to the classification. If the instance is actually in class "1" , it could either be correctly classified and counted as a True Positive (TP), or be classified as a "0" and counted as a False Negative (FN). The same occurs when the instance is actually in class "0"; if it is correctly classified then it counts as True Negative (TN) and if it is classified as "1" then it is a False Positive (FP).

In our data, since we have a low percentage of failures (Section 3.5), classifying everything as a "1" would result in a high classification accuracy even though all the failures were not found. Therefore, only considering the classification accuracy does not provide the whole picture as if removes the distinction between the correctly classified "good" pulse and correctly classified "bad" pulses. To avoid this problem, we use the True Positive Rate (TPR) and False Positive Rate (FPR) (Equation 3.4) as a measure of classification performance:

$$\begin{aligned} TPR &= \frac{TP}{TP+FN}, \\ FPR &= \frac{FP}{FP+TN}. \end{aligned} \tag{3.4}$$

Recall that the output of the classifiers (except the tree) is a score (Equation 3.3). The predicted labels are determined by comparing the score to a threshold λ, such that:

$$\hat{y}_i = \begin{cases} 1, & \text{score}_i < \lambda, \\ 0, & \text{score}_i > \lambda. \end{cases} \tag{3.5}$$

The value of the threshold depends on the problem and the desired FPR and TPR. In this study, we want to be able to identify most of the failures in the signal, while missing few of the "good" pulses. This corresponds to having a low FPR and a high TPR.

And so we applied different thresholds and calculated the TPR and FPR that resulted from each threshold. We then had a (FPR,TPR) pair that belongs to each classifiers at each threshold value. Drawing these values on a FPR-TPR plane results in a curve known as a Receiver Operating Characteristic (ROC) curve for each classifier. These curves are widely used in evaluating and comparing the performance of classifiers [38].

In the ROC space, the best classifier is represented with a point in the upper left corner with a FPR of 0 and a TPR of 1. A classifier that randomly assigns labels results in a point along a diagonal line from the bottom left corner to the upper right corner. This line separates the ROC space; the points above it represent good classification results, whereas the points below it represent bad classification results. A good classifier will have an ROC curve that is always higher than the diagonal line.

To quantify the results from the ROC curves, we can use them to calculate the Area Above the Curve (AAC). The AAC value will always be between 0 and 1. The diagonal line results in an AAC of 0.5. And so the AAC should not be larger than 0.5. The smaller the AAC, the better the classification.

Pulse Sequence Duration

To further investigate the classification performance, we examine the duration of "good" and "bad" pulse sequences. That is the number of successive pulses that were classified as "good" and the number of successive pulses that were classified as "bad". We then compare the sequence duration in the train and test data to the sequence duration in the predicted labels.

3.7.2 Impact of Movement on PPG Sensor Reliability

In order to investigate the effect of movement on PPG signals, we first apply the segmentation technique (Section 3.3) and calculate the same twelve features (Section 3.4). Then we merge the data from the same activity together.

In order to use the classification methods to decide whether the given pulses are "good" or "bad", we need to pick a decision threshold and then compare the percentage of failures detected in the data in the three activities. We use the same thresholds we used in building the ROC curves for the test data (Section 3.7.1).

Percentage of Failures and AUC

To compare the different classifiers (that have different threshold vales), we look at the corresponding FPR for each threshold. We end up with a percentage of failures at each FPR for each classifier. This is represented with a curve for each classifier.

The Area Under the Curve (AUC) is calculated to get scalar representation of the amount of error in the signal. We compare the AUC values for each classifier when the participants are resting, talking, and walking.

Pulse Sequence Duration

Similar to our evaluation method for the classifiers, we examine the duration of the good and bad pulse sequences, and compare them to the ground truth from the training and testing datasets.

4. EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Automated Failure Detection Method Evaluation

In this section, we evaluate and compare the performance of the proposed failure detection methods using three evaluation techniques. The first is the ROC curve. To obtain the ROC curve, the decision threshold for each method, except the Decision Tree, is varied. The True Positive Rate (TPR) and False Positive Rate (FPR) values are calculated at each threshold. We want to be able to identify most (ideally all) of the failures in the signal, while missing few (ideally none) of the "good" pulses. This corresponds to having a low FPR and a high TPR.

Recall from Section 3.6 that the output of all the classification methods, except the Tree, is the score ($\text{score}_i = \frac{P(\hat{y}_i=0)}{P(\hat{y}_i=1)}$). To obtain the ROC curve, the score is compared to a varying threshold λ (Equation 3.5).

On the other hand, the output of the Decision Tree is either "0" or "1". And therefore, there is no threshold to vary. Which is why, on the FPR-TPR plane, the Tree is represented with one point.

As explained in Section 3.5, we have two training sets: the original imbalanced set and the up-sampled balanced set. Each set was separately used in training each model, thus resulting in two different classifiers for each method (except the Tree). As for the testing phase, we tested on a set that contains the features from the four test subjects merged together, as well as on each test subject separately. The reason it is important to test the models on each subject is because the PPG signal, like other biosignals, is subject dependant. In a real-life setting, the proposed classifiers would be used to identify failures in individual people. It is therefore useful to assess if any of our proposed methods perform equally well across subjects.

The results of testing on the dataset with features from all the test subjects is illustrated in Figure 4.1. The models trained using the up-sampled training feature set are drawn in dashed lines. As we can see, the three undermost curves are the three SVM models trained using the original dataset. When training using the up-sampled set, the curves go up. The DNN curves are the uppermost curves when the FPR is larger that about 10%. For values less than 10%, the K-NN has higher TPR values. The Decision Tree's TPR value corresponding to a FPR of about 30% is the highest.

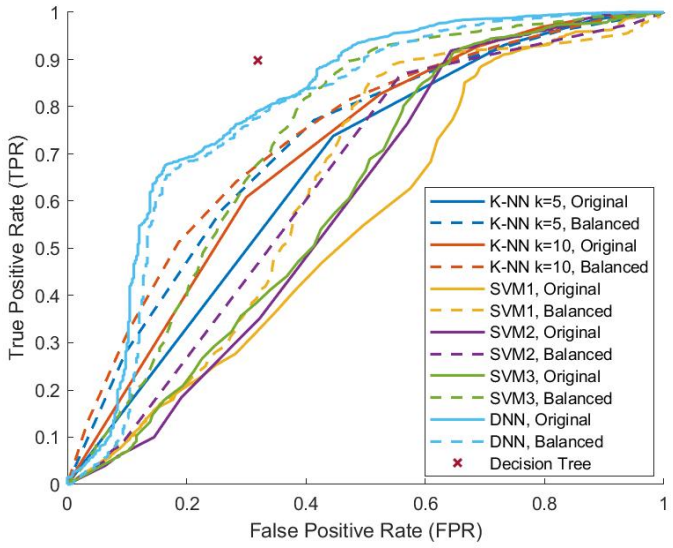


Fig. 4.1. ROC of testing results on all subjects combined.

The ROC curves resulting from testing separately on each subject are in Figure 4.2. The best and the worst methods differ for each subject. SVM1 curve is the uppermost in the first participant, however it is at the bottom in the second and third participants. The Decision Tree is the best failure detection method in the second participant since it has a FPR of 10% that corresponds to a TPR of 90%. Most of the methods performed poorly on the third subject. The best, however, is K-NN with k=5. This K-NN model does not perform well in the fourth participant. The best in this subject are SVM2 and SVM3.

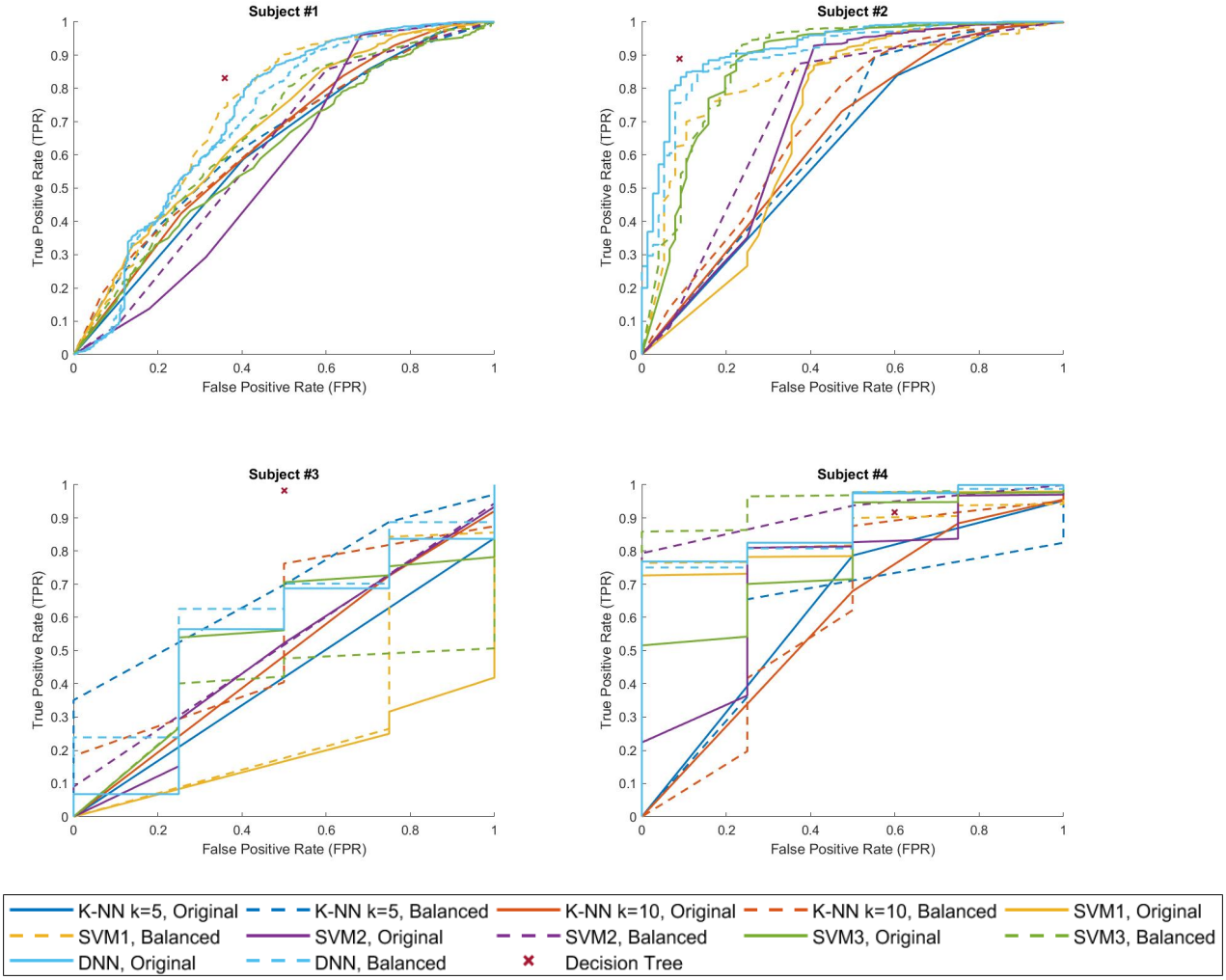


Fig. 4.2. ROC of testing results on individual subjects.

It is obviously hard to conclude which method has the best performance by studying the curves, therefore we calculate the Area Above the Curve (AAC). The smaller the AAC is, the closer the ROC curve is to the y-axis. This means it has a higher TPR for a low FPR. Accordingly, the methods with the lowest AAC have a better classification accuracy. The AAC values can be seen in Table 4.1. The smallest value for each row is highlighted. The table also includes the percentage of support vectors in each SVM model.

Table 4.1.

AAC of ROC curves

	K-NN				SVM						DNN	
	k = 5		k = 10		Feature Set 1		Feature Set 2		Feature Set 3			
	Original	Balanced	Original	Balanced	Original	Balanced	Original	Balanced	Original	Balanced	Original	Balanced
					15.80%	30.90%	14.80%	58.70%	15.60%	36.40%		
All Subjects	0.34	0.28	0.30	0.26	0.44	0.37	0.41	0.36	0.39	0.27	0.20	0.22
Sub. 1	0.38	0.36	0.36	0.36	0.32	0.28	0.43	0.39	0.39	0.33	0.28	0.29
Sub. 2	0.38	0.34	0.35	0.31	0.32	0.81	0.29	0.27	0.14	0.12	0.08	0.09
Sub. 3	0.58	0.31	0.52	0.44	0.81	0.69	0.52	0.48	0.46	0.62	0.46	0.39
Sub. 4	0.37	0.40	0.40	0.41	0.13	0.14	0.27	0.08	0.21	0.05	0.11	0.12

From the percentage of support vectors, we can see that SVM2 has a very high number of support vectors. This indicates that the model may have overfitted the data. Looking at the AAC values, we can see that, overall, DNN has the best failure detection accuracy. This is because it has small AAC values in all the testing cases.

Another finding from the table is that the AAC values for the models that were trained using the up-sampled feature set are smaller compared to the ones trained using the original feature set. This suggests that training with a balanced dataset leads to a more accurate error detection.

Since the Decision Tree is defined for only one FPR value, its AAC cannot be computed. Therefore, to compare its performance with the other methods, we examine the TPR values obtained from all the methods at the FPR value of the Decision Tree. A high TPR means that a large number of "good" signals were accurately classified. The TPR values are displayed in Table 4.2 with the highest TPR in each row highlighted.

From the table, we can see that the highest TPR values are obtained when using the Decision Tree. This means that for that specific FPR (32%), the Tree's performance surpasses the other classification methods.

To further investigate the classification performance, we examine the duration of "good" and "bad" pulse sequences. That is the number of successive pulses that were classified as "good" and the number of successive pulses that were classified as "bad". Figure 4.3 shows the histogram of sequence lengths in the training and testing

Table 4.2.
TPR of Decision Tree's FPR

	Decision Tree FPR	K-NN				SVM						DNN		Decision Tree
		k = 5		k = 10		Feature Set 1		Feature Set 2		Feature Set 3				
		Original	Balanced	Original	Balanced	Original	Balanced	Original	Balanced	Original	Balanced	Original	Balanced	
All Subjects	0.32	0.53	0.66	0.63	0.68	0.33	0.41	0.35	0.47	0.39	0.68	0.79	0.78	0.90
Sub. 1	0.36	0.52	0.58	0.54	0.55	0.60	0.75	0.36	0.48	0.50	0.59	0.66	0.64	0.84
Sub. 2	0.10	0.14	0.16	0.16	0.20	0.13	0.69	0.15	0.18	0.56	0.59	0.83	0.76	0.89
Sub. 3	0.50	0.42	0.70	0.48	0.76	0.21	0.18	0.52	0.54	0.56	0.42	0.56	0.63	0.98
Sub. 4	0.60	0.83	0.82	0.76	0.89	0.97	0.91	0.82	0.95	0.96	0.99	0.88	0.88	0.92

datasets. We can see that the two distributions are very similar. They both have a large number of long "good" sequences that are interrupted by short failures.

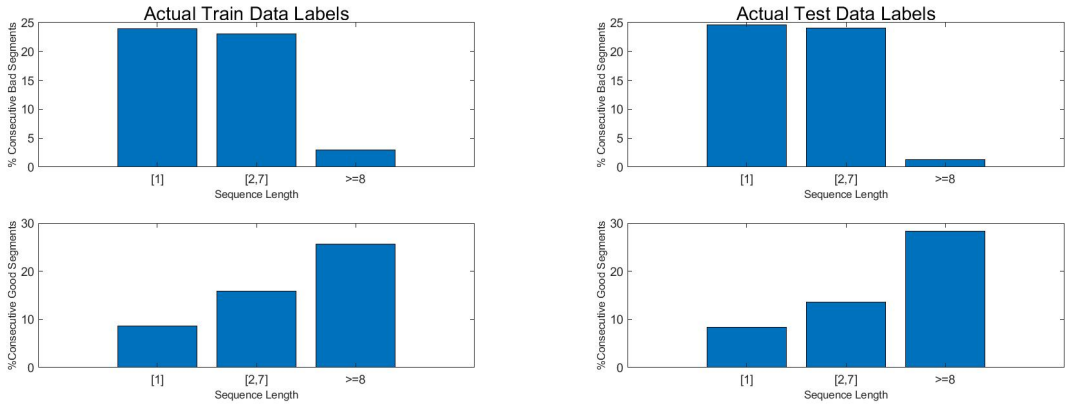


Fig. 4.3. Sequence duration histogram for training and testing datasets (Ground Truth).

To compare the predicted labels to the actual labels, we need to pick a decision threshold. We decide to use the threshold that resulted in a FPR equal to the FPR of the Tree. This allows us to compare the results of all the methods. The FPR of the Tree is 30%. The resulting histograms can be seen in Figure 4.4.

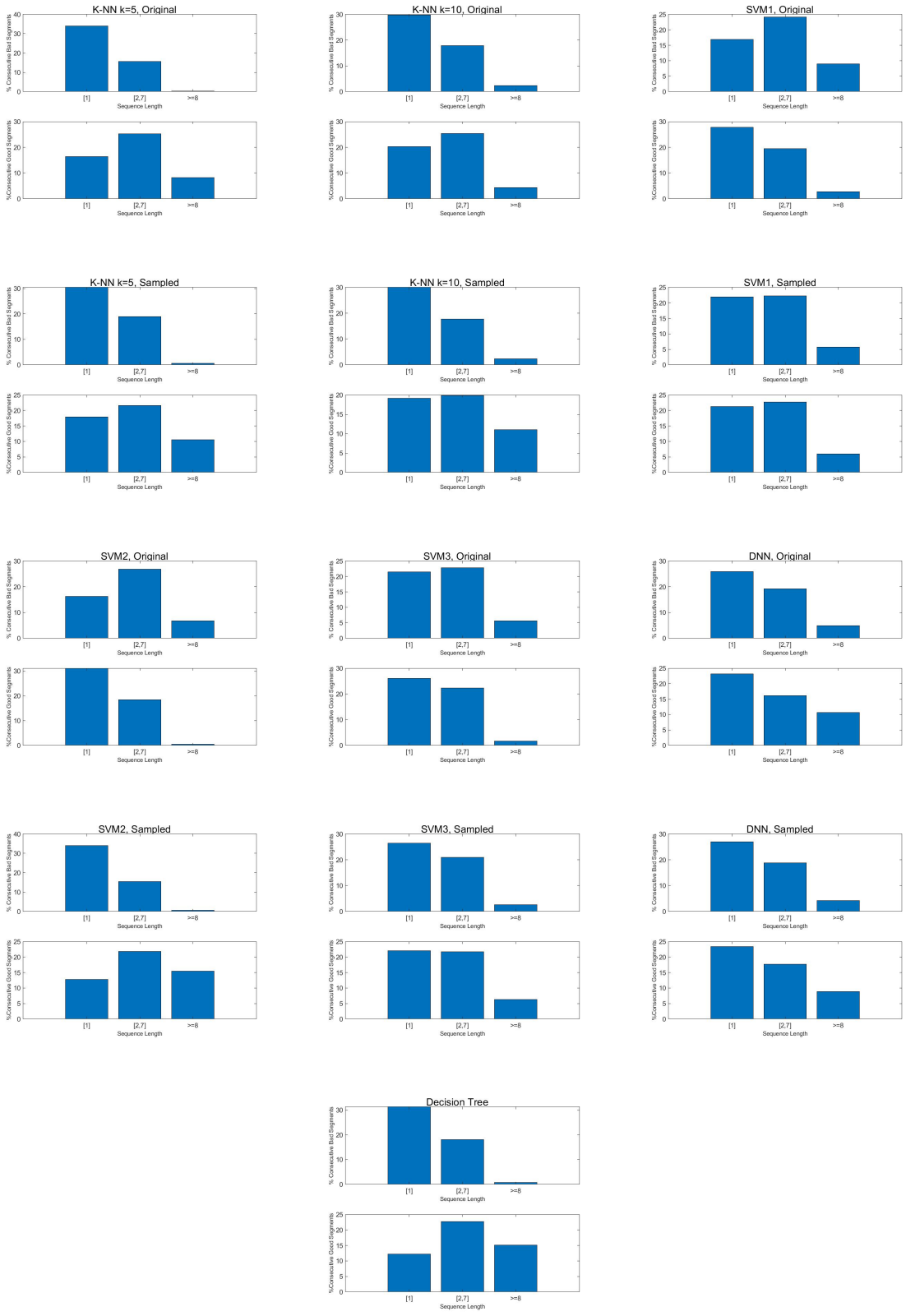


Fig. 4.4. Sequence duration histogram for all the proposed methods.

We can see that the histograms of "bad" sequences have the same overall appearance as in the testing and training histograms. This indicates that all the methods were able to detect "bad" sequences in similar proportions. On the other hand, if we study the distribution of "good" sequences, we can see that there is a smaller number of long "good" sequences than in the ground truth. This implies that the methods tend to break up long sequences of "good" pulses. If we compare the methods' performance, the Decision Tree seems to have the largest number of long "good" sequences, whereas, the SVM models were not able to retain long "good" PPG pulse sequences.

4.2 Impact of Movement on PPG Sensor Reliability

In this section, we will use the previously proposed methods to assess the reliability of the sensor-kit when acquiring data in different scenarios. To do that, we apply the proposed classification models to the unlabeled dataset that varies in the activity level. Recall that the dataset is divided into three activities; resting, talking and gesturing, and walking. After applying the failure detection methods, we compute the percentage of errors detected in each scenario.

In order to use the classification methods to label the data, we need to pick a decision threshold. We choose to pick multiple thresholds by varying the FPR from 0% to 100% and taking the corresponding thresholds that were obtained in the testing phase. The percentage of failures detected at each threshold is recorded and plotted as can be seen in Figure 4.5. Note that since there is no decision threshold in the Tree method, it is represented with a point in the plot of every method for comparison purposes.

Studying the curves, we notice that the blue line (which denotes participants resting) is the bottom line in all the methods for most of the threshold values. It is followed by the red line (which denotes participants talking) and finally the yellow line (which denotes participants walking). This indicates that the percentage of failure in the PPG signal increases with the increase of activity level.

This is consistent with the Area Under the Curve (AUC) values that we calculate for each curve in Table 4.3. The larger the AUC value is, the more failures are detected. We can see that the AUC values increase with movement in all the methods.

Table 4.3.
AUC of *FPR - Ratio of Error Detected* curve

Method/Scenario			**At rest**	**Talking & Gesturing**	**Walking**
K-NN	k = 5	Original	0.20	0.21	0.25
		Balanced	0.26	0.30	0.33
	k = 10	Original	0.23	0.25	0.29
		Balanced	0.25	0.28	0.31
SVM	Feature Set 1	Original	0.66	0.70	0.71
		Balanced	0.53	0.61	0.61
	Feature Set 2	Original	0.61	0.65	0.63
		Balanced	0.39	0.46	0.47
	Feature Set 3	Original	0.62	0.74	0.75
		Balanced	0.58	0.75	0.77
DNN		Original	0.51	0.63	0.73
		Balanced	0.27	0.44	0.56

Another interesting observation is the large number of detected failures in the signal. This is unexpected since in the training and testing datasets we had a very low error percentage. K-NN models seem to be detecting the smallest number of failures compared to the other methods.

As we did when evaluating the failure detection methods, we examine the duration of "good" and "bad" pulse sequences in the three activity scenarios. The resulting histograms are in Figure 4.6 (A numerical summary is presented in Tables 4.4 and 4.5). It is clear that it is hard to find a long "good" sequence regardless of the method used. We can also see from the histograms that there is a decrease in the number of long "good" sequences with the increase in activity level.

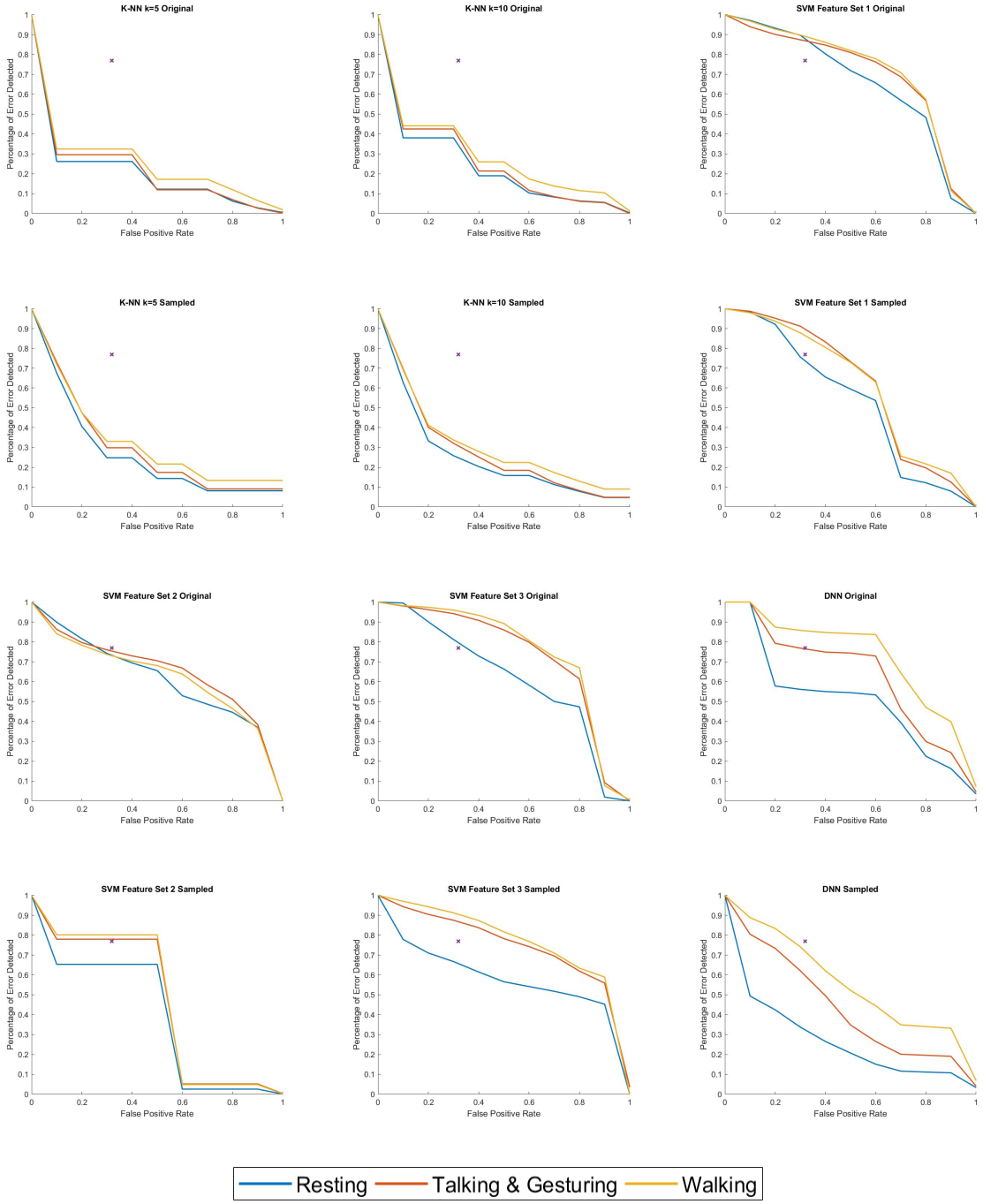
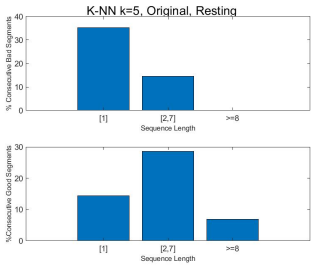
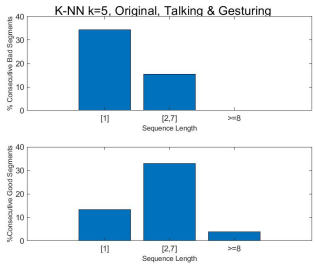
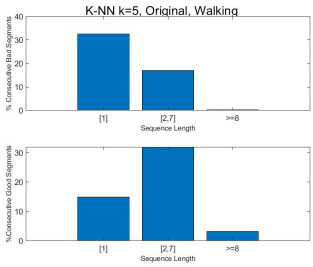
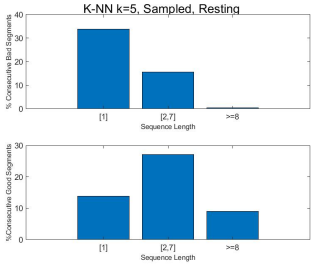


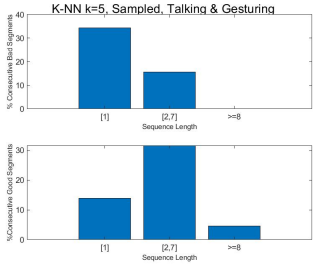
Fig. 4.5. Ratio of error detected at different FPR for three activity levels.

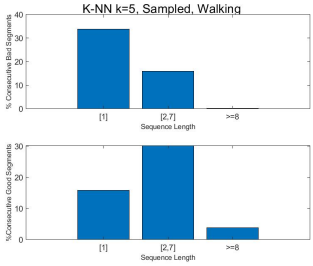

K-NN k=5, Original, Resting
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

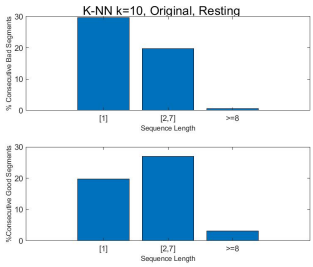

K-NN k=5, Original, Talking & Gesturing
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

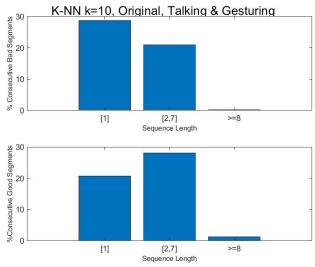

K-NN k=5, Original, Walking
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

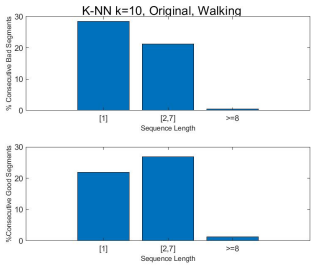

K-NN k=5, Sampled, Resting
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

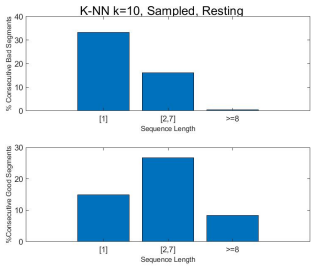

K-NN k=5, Sampled, Talking & Gesturing
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

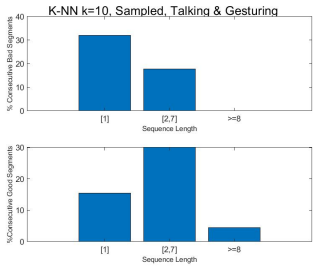

K-NN k=5, Sampled, Walking
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

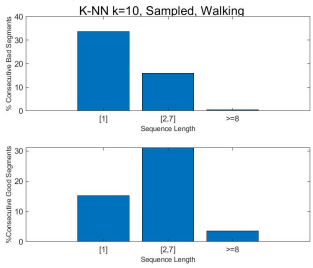

K-NN k=10, Original, Resting
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

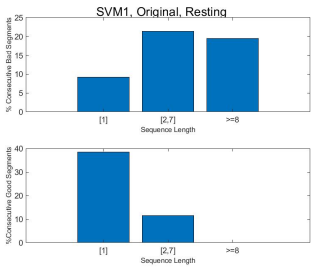

K-NN k=10, Original, Talking & Gesturing
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

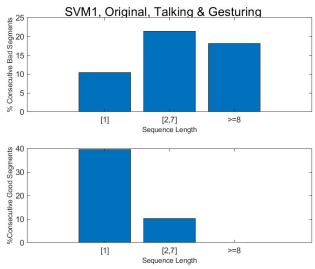

K-NN k=10, Original, Walking
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length


K-NN k=10, Sampled, Resting
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

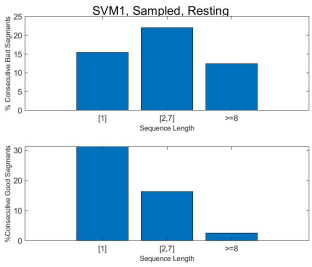

K-NN k=10, Sampled, Talking & Gesturing
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

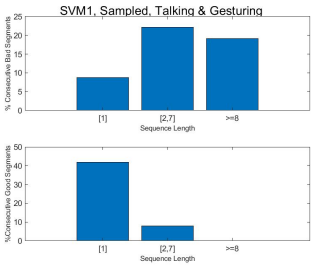

K-NN k=10, Sampled, Walking
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

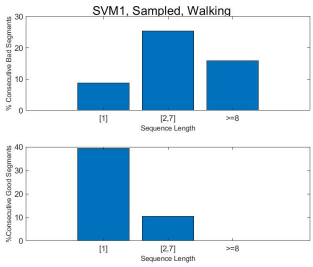

SVM1, Original, Resting
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

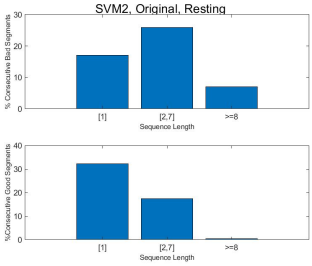

SVM1, Original, Talking & Gesturing
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

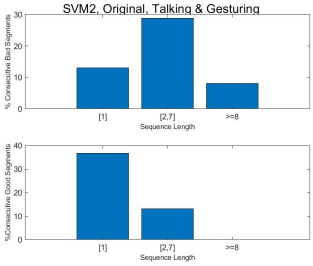

SVM1, Original, Walking
% Consecutive Bad Segments
[1]
[2,7]
>=8
Sequence Length
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length


SVM1, Sampled, Resting
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

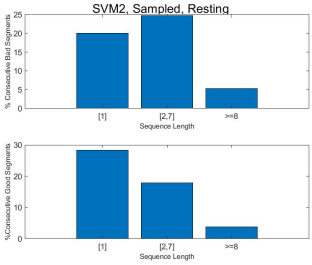

SVM1, Sampled, Talking & Gesturing
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

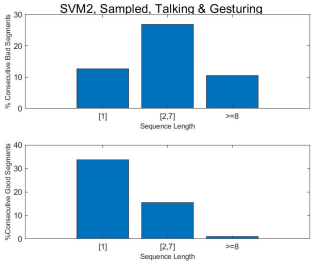

SVM1, Sampled, Walking
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

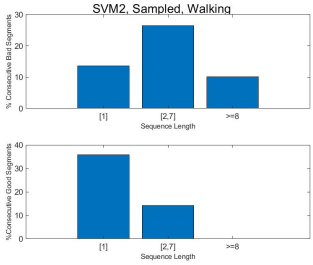

SVM2, Original, Resting
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

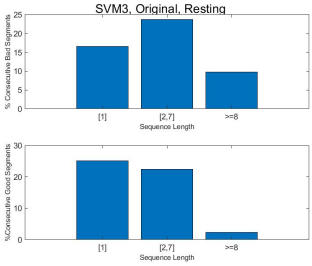

SVM2, Original, Talking & Gesturing
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

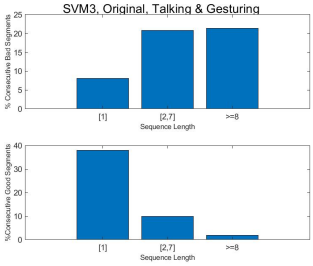

SVM2, Original, Walking
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

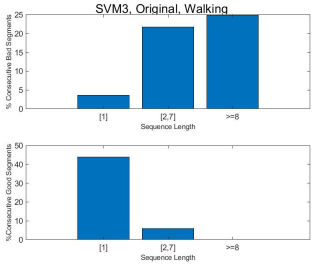

SVM2, Sampled, Resting
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

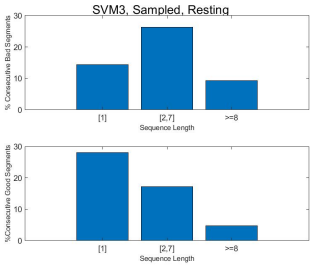

SVM2, Sampled, Talking & Gesturing
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

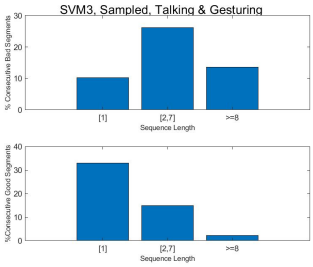

SVM2, Sampled, Walking
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

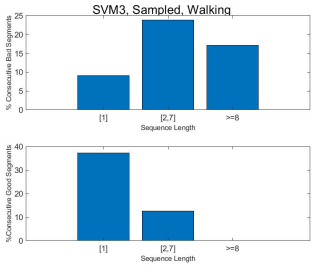

SVM3, Original, Resting
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length


SVM3, Original, Talking & Gesturing
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length


SVM3, Original, Walking
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length


SVM3, Sampled, Resting
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length


SVM3, Sampled, Talking & Gesturing
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length


SVM3, Sampled, Walking
% Consecutive Bad Segments
%Consecutive Good Segments
[1]
[2,7]
>=8
Sequence Length

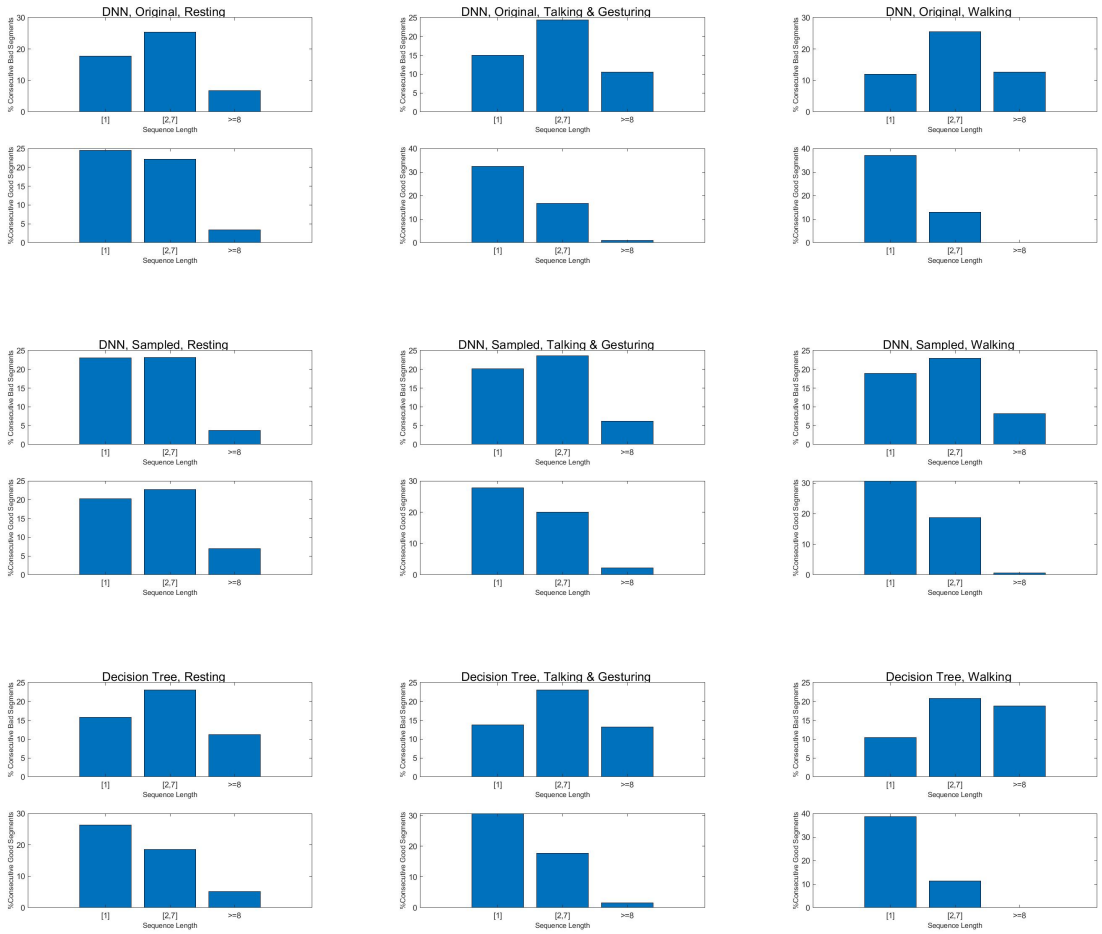


Fig. 4.6. Sequence duration histogram for all the proposed methods for three activity levels.

Table 4.4.
Histogram of "bad" pulse sequence duration for all the proposed methods for three activity levels

	[1]			[2,7]			>=8		
Method	**Resting**	**Talking**	**Walking**	**Resting**	**Talking**	**Walking**	**Resting**	**Talking**	**Walking**
SVM1, Original	9.2	10.5	6.0	21.4	21.4	22.5	19.5	18.1	21.6
SVM1, Balanced	15.5	8.7	8.8	22.1	22.1	25.4	12.4	19.1	15.9
SVM2, Original	17.1	13.1	14.6	25.9	28.8	28.3	7.0	8.1	7.1
SVM2, Balanced	20.0	12.7	13.6	24.7	26.8	26.4	5.3	10.5	10.1
SVM3, Original	16.6	8.1	3.6	23.7	20.9	21.7	9.8	21.3	24.9
SVM3, Balanced	14.4	10.3	9.1	26.3	26.1	23.9	9.3	13.6	17.2
K-NN k=5, Original	35.2	34.4	32.6	14.5	15.5	17.1	0.2	0.1	0.3
K-NN k=5, Balanced	33.9	34.3	33.8	15.7	15.6	16.0	0.4	0.0	0.2
K-NN k=10, Original	29.7	28.8	28.4	19.7	21.0	21.2	0.6	0.3	0.4
K-NN k=10, Balanced	33.3	32.0	33.7	16.2	17.9	15.9	0.4	0.1	0.4
DNN, Original	17.8	15.0	12.0	25.4	24.4	25.5	6.8	10.6	12.6
DNN, Balanced	23.1	20.2	18.9	23.2	23.6	22.9	3.8	6.1	8.2
Decision Tree	15.8	13.8	10.4	23.0	23.0	20.8	11.2	13.3	18.8

Table 4.5.
Histogram of "good" pulse sequence duration for all the proposed methods for three activity levels

	[1]			[2,7]			>=8		
Method	**Resting**	**Talking**	**Walking**	**Resting**	**Talking**	**Walking**	**Resting**	**Talking**	**Walking**
SVM1, Original	38.5	39.5	42.6	11.5	10.3	7.3	0.0	0.2	0.0
SVM1, Balanced	31.1	41.9	39.4	16.3	7.9	10.5	2.6	0.2	0.0
SVM2, Original	32.2	36.8	35.4	17.3	13.2	14.5	0.4	0.0	0.0
SVM2, Balanced	28.3	33.6	35.8	17.8	15.4	14.1	3.9	0.9	0.0
SVM3, Original	25.1	37.9	43.9	22.4	10.0	5.9	2.4	1.9	0.0
SVM3, Balanced	28.0	32.9	37.3	17.2	14.8	12.6	4.8	2.3	0.0
K-NN k=5, Original	14.5	13.3	14.9	28.7	32.9	32.0	6.9	3.8	3.1
K-NN k=5, Balanced	13.9	13.9	15.8	27.2	31.6	30.3	9.0	4.6	3.9
K-NN k=10, Original	19.8	20.6	21.9	27.0	28.1	26.9	3.2	1.3	1.2
K-NN k=10, Balanced	15.0	15.4	15.2	26.7	30.1	31.2	8.3	4.6	3.6
DNN, Original	24.5	32.4	37.0	22.1	16.8	12.9	3.4	0.9	0.0
DNN, Balanced	20.3	27.8	30.8	22.7	20.0	18.7	7.0	2.2	0.6
Decision Tree	26.4	30.6	38.7	18.5	17.8	11.3	5.1	1.5	0.0

5. CONCLUSIONS

In summary, we analyzed PPG signals and developed multiple methods to detect failures in the signal. We proposed a segmentation method that clusters the peaks into two Gaussian distributions to find the systolic peaks and used them to divide the signal into pulses. The classifiers we built are: SVM, K-NN, DNN, and Decision Tree. We compared their ROC curves and the AAC. We also examined the duration of good and bad signal sequences.

In our experiments, none of the proposed methods performed consistently well. This indicates that we need more training data, as five training subjects were not sufficient to capture all the possible variations in the PPG signal. Overall, our Decision Tree was found to perform best, albeit with only one specific FPR value (Table 4.2). When looking at different FPR values, the DNN performance appears to be somewhat better then the others (Table 4.1).

We studied the distribution of the duration of good and bad pulse sequences in the training and testing data (ground truth) and compared it to that in the classification results. The histograms indicated that all the methods were able to detect bad sequences in a similar proportion to the ground truth. On the other hand, the methods resulted in more short good sequences and less long good sequences compared to the ground truth (Figures 4.3 and 4.4). This revealed that the methods tend to destroy long sequences of good pulses.

Furthermore, we assessed the reliability of the PPG sensors when movement is introduced in the acquisition procedure. Our findings showed that there is a large number of failures in the PPG signal that increased with movement (Table 4.3). This trend was consistently observed with all proposed classification methods. We also studied the distribution of the duration of good and bad pulse sequences and noticed

that there is a decrease in the length of good sequences as we increase the activity level. This trend was also observed with all classification methods.

In conclusion, we have demonstrated that while there is no clear winner when it comes to using our proposed methods in detecting failures in PPG signals, the methods can still be used to monitor the relative performance of the sensors. The idea of observing the sensors' performance in different activities can be extended to monitoring their accuracy overtime or in different humidity levels, for example.

REFERENCES

REFERENCES

[1] "Global Wearable Sensors Market Size, Industry Report, 2018-2025," Grand View Research, Tech. Rep., 2018. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/global-wearable-sensor-market/methodology

[2] D. Castaneda, A. Esparza, M. Ghamari, C. Soltanpur, and H. Nazeran, "A review on wearable photoplethysmography sensors and their potential future applications in health care," *International journal of biosensors & bioelectronics*, vol. 4, no. 4, pp. 195–202, 2018.

[3] H. C. Koydemir and A. Ozcan, "Wearable and Implantable Sensors for Biomedical Applications," *Annual Review of Analytical Chemistry*, vol. 11, no. 1, pp. 127–146, 2018. [Online]. Available: https://doi.org/10.1146/annurev-anchem-061417-125956

[4] T. Tamura, Y. Maeda, M. Sekine, and M. Yoshida, "Wearable Photoplethysmographic Sensors: Past and Present," *Electronics*, vol. 3, no. 2, pp. 282–302, 2014. [Online]. Available: http://www.mdpi.com/2079-9292/3/2/282

[5] S. Das, S. Pal, and M. Mitra, "Real time heart rate detection from PPG signal in noisy environment," in *2016 International Conference on Intelligent Control Power and Instrumentation (ICICPI)*, 2016, pp. 70–73.

[6] M. Elgendi, "On the analysis of fingertip photoplethysmogram signals," *Current cardiology reviews*, vol. 8, no. 1, pp. 14–25, 2 2012. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/22845812 https://www.ncbi.nlm.nih.gov/pmc/PMC3394104/

[7] P. M. Mohan, V. Nagarajan, and S. R. Das, "Stress measurement from wearable photoplethysmographic sensor using heart rate variability data," in *2016 International Conference on Communication and Signal Processing (ICCSP)*, 2016, pp. 1141–1144.

[8] P. H. Charlton, P. Celka, B. Farukh, P. Chowienczyk, and J. Alastruey, "Assessing mental stress from the photoplethysmogram: a numerical study," *Physiological Measurement*, vol. 39, no. 5, 5 2018.

[9] X. He, R. A. Goubran, and X. P. Liu, "Secondary Peak Detection of PPG Signal for Continuous Cuffless Arterial Blood Pressure Measurement," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 6, pp. 1431–1439, 2014.

[10] S. Mottaghi, M. H. Moradi, and L. Roohisefat, "Cuffless Blood Pressure Estimation during Exercise Stress Test," *International Journal of Bioscience, Biochemistry and Bioinformatics*, vol. 2, no. 6, 2012.

[11] W. B. Gu, C. C. Y. Poon, and Y. T. Zhang, "A novel parameter from PPG dicrotic notch for estimation of systolic blood pressure using pulse transit time," in *2008 5th International Summer School and Symposium on Medical Devices and Biosensors*, 2008, pp. 86–88.

[12] R. Shriram, A. Wakankar, N. Daimiwal, and D. Ramdasi, "Continuous cuffless blood pressure monitoring based on PTT," in *2010 International Conference on Bioinformatics and Biomedical Technology*, 2010, pp. 51–55.

[13] M. Yee-Man Wong, C. Poon, and Y.-T. Zhang, "An Evaluation of the Cuffless Blood Pressure Estimation Based on Pulse Transit Time Technique: a Half Year Study on Normotensive Subjects," *Cardiovascular engineering (Dordrecht, Netherlands)*, vol. 9, pp. 32–38, 4 2009.

[14] G. Fortino and V. Giampà, "PPG-based methods for non invasive and continuous blood pressure measurement: An overview and development issues in body sensor networks," in *2010 IEEE International Workshop on Medical Measurements and Applications, MeMeA 2010 - Proceedings*, 2010, pp. 10–13.

[15] A. Fernandes, R. Helawar, R. Lokesh, T. Tari, and A. V. Shahapurkar, "Determination of stress using Blood Pressure and Galvanic Skin Response," in *2014 International Conference on Communication and Network Technologies*, 2014, pp. 165–168.

[16] R. Rakshit, V. R. Reddy, and P. Deshpande, "Emotion Detection and Recognition Using HRV Features Derived from Photoplethysmogram Signals," in *Proceedings of the 2Nd Workshop on Emotion Representations and Modelling for Companion Systems*, ser. ERM4CT '16. New York, NY, USA: ACM, 2016, p. 2:12:6. [Online]. Available: http://doi.acm.org/10.1145/3009960.3009962

[17] D. Zheng, X. Zhang, H. Liu, K. Budidha, and P. A. Kyriacou, "Article 1863 Citation: Budidha K and Kyriacou PA (2019) Photoplethysmography for Quantitative Assessment of Sympathetic Nerve Activity (SNA) During Cold Stress," *Frontiers in Physiology — www.frontiersin.org*, vol. 1, p. 1863, 2019. [Online]. Available: www.frontiersin.org

[18] G. Udovičić, J. erek, M. Russo, and M. Sikora, "Wearable Emotion Recognition System based on GSR and PPG Signals," in *Proceedings of the 2nd International Workshop on Multimedia for Personal Health and Health Care*. ACM, 2017, pp. 53–59. [Online]. Available: http://doi.acm.org/10.1145/3132635.3132641

[19] R. R. Singh, S. Conjeti, and R. Banerjee, "A comparative evaluation of neural network classifiers for stress level analysis of automotive drivers using physiological signals," *Biomedical Signal Processing and Control*, vol. 8, no. 6, pp. 740–754, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1746809413000992

[20] C. Fischer, B. Dömer, T. Wibmer, and T. Penzel, "An Algorithm for Real-Time Pulse Waveform Segmentation and Artifact Detection in Photoplethysmograms," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 2, pp. 372–381, 2017.

[21] C. Orphanidou, T. Bonnici, P. Charlton, D. Clifton, D. Vallance, and L. Tarassenko, "Signal Quality Indices for the Electrocardiogram and Photoplethysmogram: Derivation and Applications to Wireless Monitoring," *IEEE Journal of Biomedical and Health Informatics*, vol. 19, no. 3, pp. 832–838, 2015.

[22] J. A. Sukor, S. Redmond, and N. Lovell, "Signal quality measures for pulse oximetry through waveform morphology analysis," *Physiological measurement*, vol. 32, pp. 369–384, 4 2011.

[23] D. Yang, Y. Cheng, J. Zhu, D. Xue, G. Abt, H. Ye, and Y. Peng, "A Novel Adaptive Spectrum Noise Cancellation Approach for Enhancing Heartbeat Rate Monitoring in a Wearable Device," *IEEE Access*, vol. 6, pp. 8364–8375, 2018.

[24] E. Bonham, T. Wehrly, D. Alabed, B. Ma, W. Sanchez, A. Reibman, and M. Boutin, "Sensor-kit Assembly Manual," 2019. [Online]. Available: https://purr.purdue.edu/publications/3181/1

[25] "Wearable Sensor Kit to Characterize Affect Project." [Online]. Available: https://engineering.purdue.edu/brl/SKT/SensorKit.html

[26] T. Wehrly, D. Alabed, and M. Boutin, "Labeled Raw PPG Signals Measured Using Wearable Sensor-kit," 2019. [Online]. Available: https://purr.purdue.edu/publications/3179/1

[27] ——, "Raw PPG Signal Measured Using Wearable Sensor-kit in Varying Levels of Activity," 2019. [Online]. Available: https://purr.purdue.edu/publications/3180/1

[28] M. Elgendi, "Optimal Signal Quality Index for Photoplethysmogram Signals," *Bioengineering (Basel, Switzerland)*, vol. 3, no. 4, p. 21, 9 2016. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/28952584 https://www.ncbi.nlm.nih.gov/pmc/PMC5597264/

[29] K. Fukunaga, "Chapter 3 - HYPOTHESIS TESTING," in *Introduction to Statistical Pattern Recognition (Second Edition)*, 2nd ed., K. Fukunaga, Ed. Boston: Academic Press, 1990, pp. 51 – 123. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780080478654500090

[30] D. Alabed and M. Boutin, "Code and Datasets for Photoplythesmogram (PPG) Signal Reliability Analysis in a Wearable Sensor-Kit," 2019. [Online]. Available: https://purr.purdue.edu/publications/3183/1

[31] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Trans. on Knowl. and Data Eng.*, vol. 21, no. 9, pp. 1263–1284, 9 2009. [Online]. Available: https://doi.org/10.1109/TKDE.2008.239

[32] N. V. Chawla, "Data Mining for Imbalanced Datasets: An Overview," in *Data Mining and Knowledge Discovery Handbook*, Maimon Oded, , and L. Rokach, Eds. Boston, MA: Springer US, 2010, pp. 875–886. [Online]. Available: https://link.springer.com/book/10.1007/978-0-387-09823-4

[33] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *J. Artif. Int. Res.*, vol. 16, no. 1, pp. 321–357, 6 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=1622407.1622416

[34] "Support Vector Machines for Binary Classification." [Online]. Available: https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html

[35] C. Campbell and Y. Ying, "Learning with Support Vector Machines," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 5, no. 1, pp. 1–95, 2 2011.

[36] "Neural Network Definition — DeepAI." [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/neural-network

[37] D. Kriesel, *A Brief Introduction to Neural Networks*, 2007. [Online]. Available: available at http://www.dkriesel.com

[38] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016786550500303X