UNIVERSITY OF CALIFORNIA
SANTA CRUZ

BREATHING RATE PREDICTION USING FINGER-TIP SENSOR

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Holakou Rahmanian

June 2015

The Dissertation of Holakou Rahmanian
is approved:

Professor Manfred K. Warmuth, Chair

Professor Matthew R. Guthaus

Professor David P. Helmbold

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

List of Tables

Abstract

Breathing Rate Prediction Using Finger-Tip Sensor

by

Holakou Rahmanian

Personalized health-care is trending and individuals tend to wear sensors in order to record their own health data. As a part of this trend, any redundancy in the data captured by wearable sensors must be exploited to reduce the number of devices one may wear. In this thesis, we work with a device which senses breathing and pulse through pressure tube and pulse oximetry, respectively. Extracting the dependency between these two measurements, we approximately predict the breathing rate by first reconstructing the breathing signal using the data coming from the finger-tip sensor, and then detecting the peaks in the reconstructed signal. For breathing signal reconstruction, two different techniques are used: (1) applying low- and high-pass filters on the pulse signal (2) training a neural network on a prepared dataset. Our experiments show that neural networks have a better performance comparing to filters in reconstructing the breathing signal, and consequently, predicting the breathing rate.

To my parents

Acknowledgments

I would like to thank all the reading committee members for their helpful comments and feedback. I would like to thank my advisor Manfred Warmuth who not only introduced me to the field of health informatics and the idea of personalized health-care, but also supported me and gave me guidance during the work on this thesis. I would also like to thank Matthew Guthaus for providing his sensor device, and Hunter Nichols for assisting me to obtain measurement data. Moreover, many thanks to volunteers who participated in my research to produce sensor data. And thanks to my friends and fellow graduate students in CS, CE, and EE departments here at UCSC for their helpful comments about my thesis.

Chapter 1

Introduction

1.1 Wearable Sensors and Personalized Healthcare

Regular exercising has become a part of almost everyone's daily routine. People are now becoming increasingly more concerned about storing and analyzing their personal records regarding their exercises and activities. They wear different sensors which are mostly connected to their smart phones recording their health data. Tens of sensors in the forms of wrist-let, chest strap, and others, have been designed to capture those data in a non-invasive manner. Hundreds of smart phone applications have been developed to properly store and analyze data coming from various sensors. These data and measurements might be heart beats and pulse, heart rate variation, number of strides they took, distance they went, breathing rate, and etc.

In general, there is a huge trend of "personalized health-care." Using various

wearable sensors and smart phone applications, people want to record their health data, analyze them and perhaps share it with their physicians if needed.

1.2 Motivation: Exploiting Redundancy

For the purpose of personalized health-care, people are not willing to wear several sensors; it is not only uncomfortable, but also may affect the recorded data itself since it might be more invasive. Wearing some sensors can even interfere with activities one may do. For instance, wearing a mask on face to capture breathing data may cause a negative effect on performance of the respiratory system.

Therefore we are interested in reducing the number of wearable sensors to a minimum. In order to do this, any possible redundancy and patterns among various measurements of different sensors must be exploited. Once this is done, we are able to capture the same measurements with fewer sensors. Thus our goal is to extract measurements of a particular sensor using the data captured from other sensors.

1.3 Breathing Rate Prediction

Cardiac and respiratory data are the most essential real-time data for monitoring the health of an individual. Different wearable sensors can be used to monitor these data streams. Heart beats and pulse are mostly sensed on the finger-tip, wrist or chest area, while breathing is usually captured by accelerometer on the chest area, or

sensing pressure on the mouth and/or nose.

In this thesis, we work with a device [10] which senses the breathing signal through a pressure tube and captures the pulse through a finger-tip sensor. Our central goal is to predict the breathing rate using the pulse signal in order to eliminate redundancy. We call this *the breathing rate prediction problem*. Since the finger-tip sensor seems to capture respiratory-related data in addition to the pulse, we are able to extract the breathing rate — verified by the pressure sensor — out of the data coming from the finger-tip sensor.

To predict the breathing rate, we first reconstruct the breathing signal using the data coming from the finger-tip sensor, then we detect the peaks of the reconstructed signal each of which denotes an occurrence of a breath. In order to reconstruct the breathing signal, we take two different approaches. In the first approach, we treat this reconstruction task as a digital signal processing problem and apply low- and high-pass filters in order to zero out the unrelated frequencies. This lets us obtain the approximate breathing signal and consequently breathing rate. In the second approach we use machine learning techniques and consider the breathing signal and the data from the finger-tip sensor as the target value and feature vector, respectively. Applying Neural Networks, we approximately predict the breathing signal from which the breathing rate can be extracted. Our experiments show that neural networks have a better performance comparing to filters in reconstructing the breathing signal, and consequently,

predicting the breathing rate.

1.4 Related Work

Extracting the respiratory signal from the measurements captured from different non-invasive sensors has been an interesting problem to solve since a few decades ago. The measurements, however, are mostly coming from the fairly advanced sensors in hospitals and away from personalized health-care and simple wearable devices. Also it seems that all of the previous works only use signal processing techniques (mostly filtering) to obtain the breathing signal and/or rate. Despite the recent wide usage of machine learning in health-care informatics [8], apparently there is still no machine learning approach for the breathing rate prediction problem. Unlike simple signal processing approaches, machine learning models are able to adapt to the data.

Different approaches have been proposed to extract the respiratory signal out of ECG measurements using techniques in signal processing and in particular filtering [14, 15, 18]. There are works in which the breathing signal is reconstructed from EBI [16], and PPG and IP [13] measurements again using only signal processing algorithms and techniques. A more related work to this thesis can be found in [20], in which the breathing rate is approximated by counting the number of peaks in LED signals from the pulse oximetry.

The rest of this thesis is organized as follows. In Chapter 2, we introduce and define the breathing rate prediction problem. In Chapter 3, we discuss applying filtering as an approach to solve this problem and also present the real-time setting for ideal filters to perform. Chapter 4 explores a machine learning approach towards the respiratory signal prediction problem and uses neural networks trained with a suitable dataset which was extracted from the signals. Finally, in Chapter 5 we compare the performances of the two approaches, and we conclude in Chapter 6.

Chapter 2

The Breathing Rate Prediction Problem

In this chapter, we start by describing the sensor device and see what types of data it generates. Then we define the breathing rate prediction problem and decompose it into two components. First, predicting the breathing signal, and second, detecting peaks in the reconstructed signal in order to predict the breathing rate.

2.1 The Sensor Device

In this thesis, we work with a device [10] (see Figure 2.1) which consists of two components and provides two synchronized sources of data with a shared sampling frequency of 20 Hz:

1. **Fingertip Component:** This component, which is shown in the upper-left of Figure 2.1, goes on a finger-tip of the human subject. It consists of two LEDs (red and infra-red) and for each LED it indirectly measures the portion of the beam

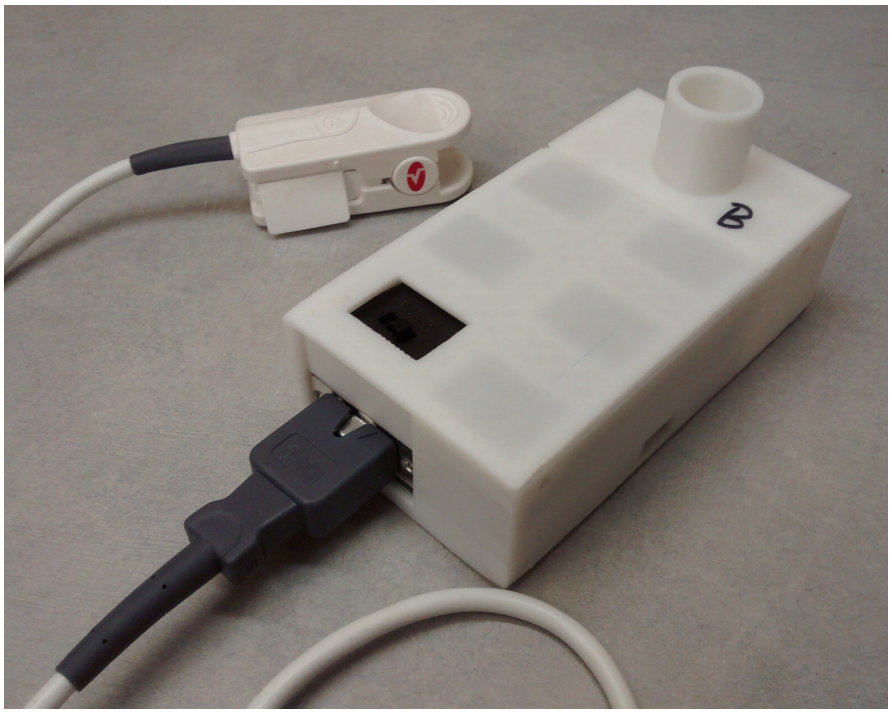

Figure 2.1: The sensor device with two components: (1) the finger-tip component on the upper-left and (2) the mouth component on the upper-right.

that goes through the finger in terms of a voltage output. This component mainly senses the pulse [11].

2. **Mouth Component:** This component, which is shown in the upper-right of Figure 2.1, is a tube that goes into the mouth of the human subject. It contains a pressure sensor that measures the pressure caused by breathing in mmHg. This component mainly senses the breathing signal.

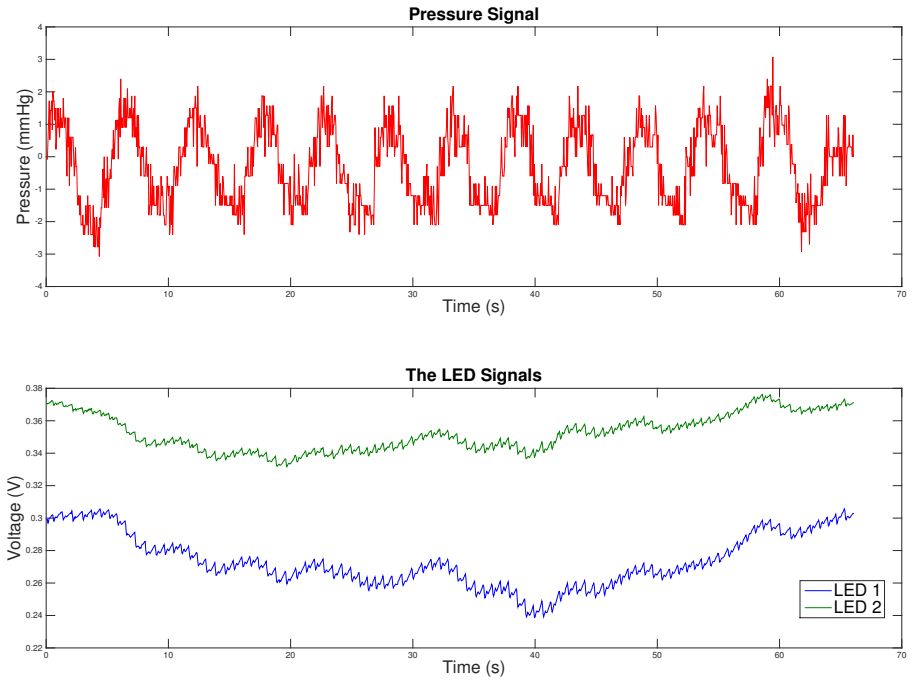

Figure 2.2: Sample cardiac and respiratory data from the device.

2.2 The Data

The data that we worked on consists of various sets of one-minute synchronized measurements of the two sensors on six different human subjects who remained seated and breathed normally entirely through their mouth. We performed 3 one-minute trials on each volunteer in order to obtain training and test set data. Figure 2.2 plots a sample measurement sensed by different components of the sensor device.

As one can observe, the pressure sensor measures the breathing signal as a

sinusoidal signal with some noise. The positive and negative values of pressure indicate exhalation and inhalation in the breathing, respectively.

Interestingly enough, the finger-tip component senses a mixture of cardiac and respiratory data. Concretely, it not only measures the pulse (the high frequency part), but also the measurements are affected by a lower frequency signal which seems to be related to breathing.

2.3 The Breathing Rate Prediction Problem

Our goal is to approximately predict the the breathing rate in real-time [1] by using the data from the finger-tip component (i.e., the LED signals). Note that breathing rate is defined as the number of breaths (exhalation or inhalation) in one minute. Thus in order to predict the rate at any time t, we need to detect all the breaths in a time interval ending at t. In this thesis, we consider occurrence of an exhalation as one breath.

We predict the breathing rate in two steps. First, we reconstruct the breathing signal (signal reconstruction) using the LED signals. Then, having reconstructed the breathing signal, we detect the occurrences of exhalations throughout the signal (peak detection). Notice that we want to do this online; meaning that at each time, as we receive data from the finger-tip sensor, we predict the value of the breathing signal in that time and decide whether an exhalation has occurred. Each of these steps are

[1] To do this task in online manner, we may take some time first for calibration or learning phase.

discussed separately in the next two subsections.

2.3.1 Signal Reconstruction

Given the data from the fingertip sensor, we want to reconstruct the breathing signal. Note that since our ultimate goal is to find the breathing rate, we just need to reconstruct the signal sufficiently well so that there is a synchronized one-to-one correspondence between the peaks of the original and reconstructed signal. In other words, it does not matter to us if the reconstructed signal is shifted and/or scaled in amplitude comparing to the original signal.

In this thesis, we apply different techniques and algorithms to reconstruct the signal. In order to evaluate the performances of different approaches and algorithms, we need to use a loss function that can reflect our goal in reconstruction. As mentioned earlier, we are not concerned with the amplitude and baseline of the reconstructed signal. Thus we use the *signal error function* (SEF) which is defined as below:

Definition 1. *Let f and g be two signal functions defined over $X = \{x_1, \ldots, x_n\}$. Also let $\bar{f}$ and $\bar{g}$ be the mean value of f and g over X, respectively. Concretely:*

$$\bar{f} = \mathbb{E}_{x \in X}[f(x)], \quad \bar{g} = \mathbb{E}_{x \in X}[g(x)]$$

Given the mean value of the signals, let $\tilde{f}$ and $\tilde{g}$ be the zero mean signals obtained from centering f and g, respectively. That is:

$$\forall\, x \in X \quad \tilde{f}(x) = f(x) - \bar{f}, \qquad \tilde{g}(x) = g(x) - \bar{g}$$

Then the Signal Error Function (SEF) of the two signals is defined as the discrepancy between $\tilde{f}$ and the best matching $\tilde{g}$ upto amplitude scaling, which is:

$$SEF_L(f,g) = \min_{a\in\mathbb{R}^+} \sum_{x\in X} L(\tilde{f}(x), a\cdot\tilde{g}(x))$$

where $L : \mathbb{R}^2 \to \mathbb{R}^+$.

In Definition 1, we are mainly interested in two specific types of L: (1) absolute loss (i.e., $L(x,y) = |x-y|$) and (2) square loss (i.e., $L(x,y) = (x-y)^2$). Using square loss has this advantage that the minimizing a in Definition 1 has an analytic solution.

Lemma 1. *If the square loss is used in Definition 1, then the "a" which minimizes the discrepancy between $\tilde{f}$ and the best matching $\tilde{g}$ upto amplitude scaling in Definition 1 can be found as below:*

$$a^* = \frac{\sum_{x\in X} \tilde{f}(x)\cdot\tilde{g}(x)}{\sum_{x\in X}(\tilde{g}(x))^2}$$

Proof.

$$\begin{aligned} J(a) &= \sum_{x\in X} (\tilde{f}(x) - a\cdot\tilde{g}(x))^2 \\ &= \sum_{x\in X} (\tilde{f}(x))^2 + a^2\cdot(\tilde{g}(x))^2 - 2a\cdot\tilde{f}(x)\cdot\tilde{g}(x) \\ &= \left(\sum_{x\in X}(\tilde{g}(x))^2\right)\cdot a^2 - 2\left(\sum_{x\in X}\tilde{f}(x)\cdot\tilde{g}(x)\right)\cdot a + \sum_{x\in X}(\tilde{f}(x))^2) \end{aligned}$$

In order to find the minimizing a, we set $\frac{d\,J}{d\,a} = 0$. Thus:

$$\begin{aligned} &\frac{d\,J}{d\,a} = 2\left(\sum_{x\in X}(\tilde{g}(x))^2\right)\cdot a - 2\left(\sum_{x\in X}\tilde{f}(x)\cdot\tilde{g}(x)\right) = 0 \\ &\longrightarrow a^* = \frac{\sum_{x\in X}\tilde{f}(x)\cdot\tilde{g}(x)}{\sum_{x\in X}(\tilde{g}(x))^2} \end{aligned}$$

□

However, the square loss has one major flaw which is over-punishing outliers in the data by squaring their associated error. On the other hand, absolute loss is less sensitive to outliers, but the minimizing a in Definition 1 cannot be found in closed form analytically. Nevertheless, the function $\sum_{x \in X} L(\tilde{f}(x), a \cdot \tilde{g}(x))$ will be a one-dimensional convex piece-wise linear function and therefore a linear program which can be minimized through simplex-based methods.

In this thesis, we take two different approaches for signal reconstruction. In the first approach, we use filters to throw away unrelated frequencies so that we can merely have the respiratory-related frequencies in our signal (Chapter 2). In the second approach, we see the problem as a regression problem in machine learning. We prepare datasets (feature vectors with target values) and feed them to neural network (Chapter 3).

2.3.2 Peak Detection

Once the signal is reconstructed, we want to find the peaks (i.e., the occurrences of exhalation) in an online fashion. Peak detection may not be straightforward due to the fact that the reconstructed signal is not usually "nice." See Figure 2.3. Observe that the reconstructed signal may also contain a significant amount of noise. In addition, the reconstructed signal may not look like a wave with a fixed amplitude.

We use Algorithm 1 to detect all peaks. This algorithm belongs to the family

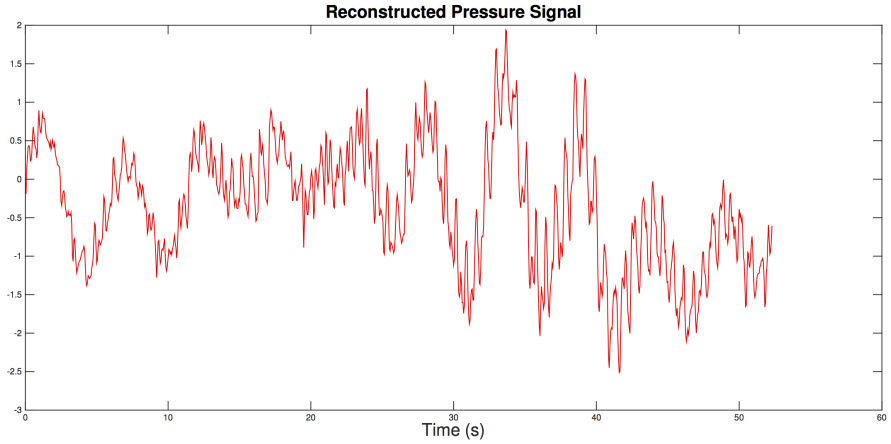

Figure 2.3: Sample reconstructed breathing signal which has ten peaks.

of moving average based filtering techniques which are common in detecting simple patterns in the signal [6]. Basically, at each point we find out whether the signal is going "up" or "down." Once we realize that the signal is going "up", the moment that the signal changes to go "down", we declare a peak [2].

The algorithm works with two window parameters (w_1 and w_2) and a threshold parameter (C). First, for each point, it checks whether the value of the point is less or greater than the average value of a window of w_1 next values. This indicates a *weak* "up" or "down." Note that averaging out the values of the signal has the same behavior as applying a low-pass filter [17]. This will help us getting rid of noise which are mostly higher frequency waves within the signal.

[2]Essentially, "up" and "down" indicate positive and negative derivative of the smoothed signal, respectively, and a"peak" is where the slope of the smoothed signal is equal to zero.

Algorithm 1 Linear Algorithm for Finding All Peaks in a Signal

procedure FINDPEAKS($f[1 \ldots n], w_1, w_2, C$)

Status $\leftarrow$ "None"

Peaks $\leftarrow \emptyset$

for $i \in \{1, 2, \ldots, n\}$ **do**

$I(i) \leftarrow$ Sgn(Average($f[i+1, i+2, \ldots, i+w_1]$)$-f[i]$)

if $\sum_{j=i-w_2}^{i} I(j) > C$ **then**

Status $\leftarrow$ "Up"

end if

if $\sum_{j=i-w_2}^{i} I(j) < -C$ **then**

if Status = "Up" **then**

Peaks $\leftarrow$ Peaks $\cup\{i - C\}$

end if

Status $\leftarrow$ "Down"

end if

end for

return Peaks

end procedure

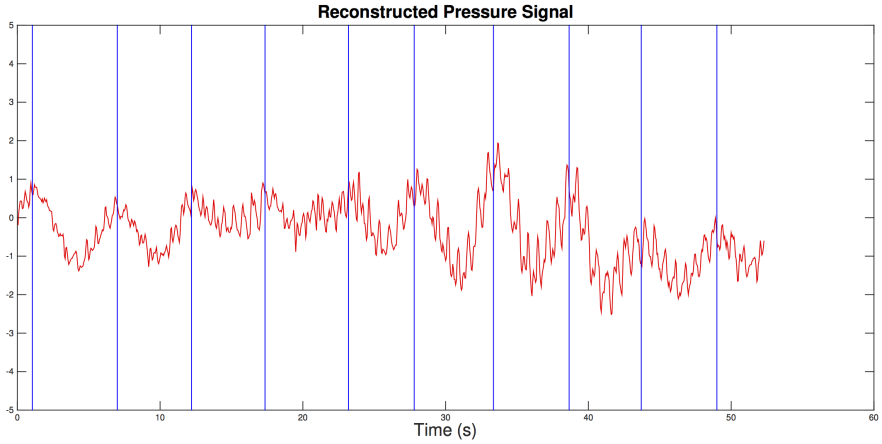

Figure 2.4: The reconstructed signal from Figure 2.3 with detected peaks using Algorithm 1.

Now a *strong* "up" or "down" is identified by a C-majority of a window of w_2 previous values. Concretely, if the number of weak "up"s are C more than the number of weak "down"s in the previous window, then we call it a strong "up". Strong "down"s are similarly identified.

Notice that Algorithm 1 can work in real-time. In other words, in order to apply the algorithm, you do not need to pass the entire signal at once; it can receive the values of the signal point by point as an input. Figure 2.4 shows the performance of Algorithm 1 on the signal in Figure 2.3 with parameters $w_1 = 30, w_2 = 20, C = 10$. Note that even though, the signal is considerably noisy, Algorithm 1 is still able to detect the approximate locations of the peaks throughout the signal.

Chapter 3

Signal Reconstruction by Filtering

Every signal consists of waves with various frequencies and amplitudes. By using Fourier series, any periodic function can be decomposed into the sum of finite or infinite set of simple sine waves. In particular, given a discrete signal in time domain, we can find the components of the signal in the frequency domain which is basically the spectrum of frequencies and their amplitudes that are constructing the signal. Using discrete Fourier transform (DFT) and its inverse, we can go from time domain to frequency domain and vice versa, respectively [17] (See Figure 3.1). Concretely, if x is a discrete signal of length N and X is its associated frequency spectrum (which also has the length N), then:

$$\text{DFT:} \qquad \forall\, k \in \{0, \ldots, N-1\} \quad X_k = \sum_{n=0}^{N-1} \omega^{kn} x_n$$

$$\text{Inverse-DFT:} \qquad \forall\, n \in \{0, \ldots, N-1\} \quad x_n = \frac{1}{N} \sum_{k=0}^{N-1} \omega^{-kn} X_k$$

in which $\omega = e^{-\frac{2\pi i}{N}}$. Assuming that x and X are represented by $N \times 1$ column vectors, DFT and its inverse can be also shown in matrix-vector multiplication form:

DFT:
$$\begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{N-1} \\ & & \vdots & \\ 1 & \omega^{j} & \cdots & \omega^{j(N-1)} \\ & & \vdots & \\ 1 & \omega^{N-1} & \cdots & \omega^{(N-1)^2} \end{bmatrix}}_{N \times N} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

Inverse-DFT:
$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} = \frac{1}{N} \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \cdots & \omega^{-(N-1)} \\ & & \vdots & \\ 1 & \omega^{-j} & \cdots & \omega^{-j(N-1)} \\ & & \vdots & \\ 1 & \omega^{-(N-1)} & \cdots & \omega^{-(N-1)^2} \end{bmatrix}}_{N \times N} \begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{bmatrix}$$

DFT and its inverse are done in $O(N^2)$ if we use naive matrix-vector multiplication procedure which is inefficient. However, by using Fast Fourier Transform [17], they can be done in $O(N \log N)$.

In this chapter, we start by introducing filters in signal processing. Then we discuss how applying filters can help us reconstructing the breathing signal, and consequently predicting the breathing rate. Since we are interested in applying filters in

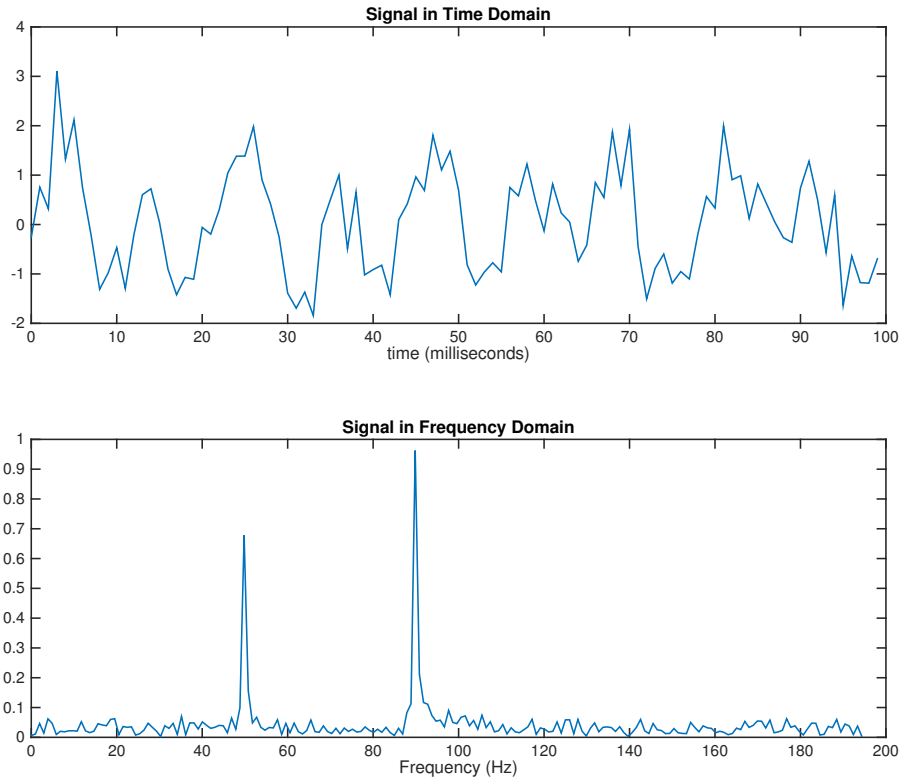

Figure 3.1: Comparing time and frequency domain of a discrete signal. The figure on the top shows a discrete signal in time domain which is a mixture of two sine waves with frequencies 50 Hz and 90 Hz with amplitudes 0.7 and 1, respectively, and some Gaussian noise. The figure on the bottom shows the same signal in frequency domain.

real-time, we also explore the setting for online filtering and online DFT. In particular, we propose a novel and efficient algorithm for the latter called Fast Sliding Window DFT.

3.1 Filters

In digital signal processing, filters [17] are used to remove a specific range of frequencies out of a given signal. In particular, given a signal, low- and high-pass filters allow frequencies below and beyond a given threshold to pass, respectively. In other words, in filters, we zero out the unrelated frequencies in the frequency domain and find the corresponding filtered signal in the time domain.

Concretely, the filters work as if a specific function is multiplied to the frequency spectrum in the frequency domain so that certain frequencies are passed and the rest of attenuated. For instance, the low-pass filter is basically multiplying the function depicted in Figure 3.2 (which is known as the *frequency response*) with the frequencies of the signal in the frequency domain.

As one can observe, this filter zeros out the frequencies beyond a certain threshold and passes the frequencies below that. In many applications, the filter in Figure 3.2 (which is also known as the *ideal filter*) is hard to implement. Therefore, there are different alternatives as approximations to ideal frequency response that can be used

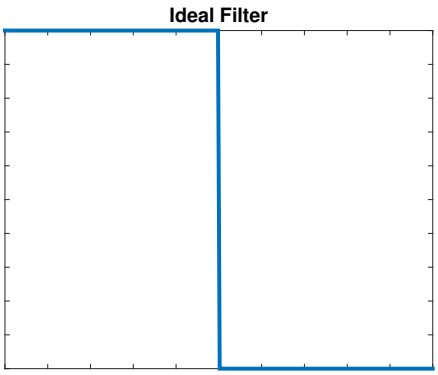

Figure 3.2: Ideal low-pass filter frequency response

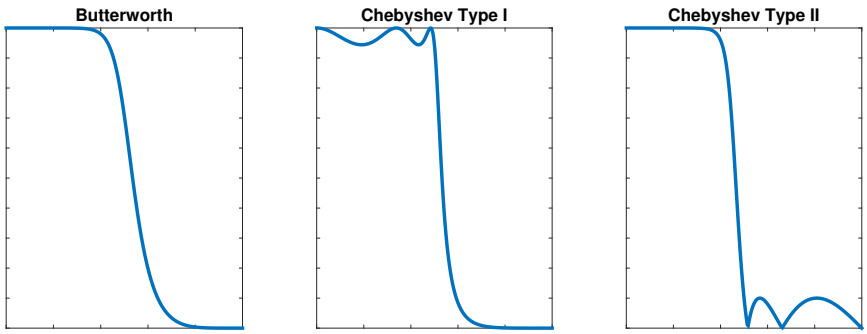

Figure 3.3: Frequency responses associated to some other common low-pass filters.

(See Figure 3.3).

3.2 Reconstructing the Breathing Signal

As discussed before, the finger-tip component of the device senses a mixture of cardiac and respiratory signals. Since we want to extract and reconstruct the breathing signal, we need to filter out unrelated frequencies including the pulse signal. Therefore

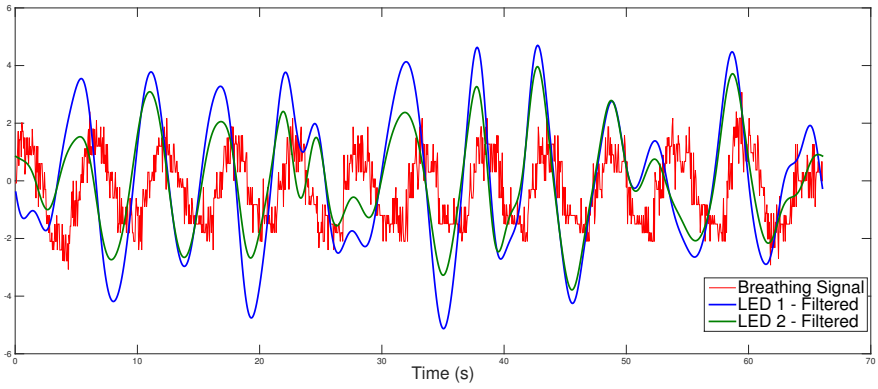

Figure 3.4: Filtered and amplitude-scaled LED signals compared to the breathing signal.

given a range of respiratory frequency spectrum, we are able to extract the breathing signal.

Depending on the population, we can use different range of respiratory frequency spectrum. In our experiment, we assume that the breathing frequency falls into 0.1 to 0.5 Hz. This is 6 to 30 breathes per minute which is a very wide range and in particular includes the typical respiratory rate for a healthy adult at rest [2].

Thus, assuming that the breathing signal is directly mixed within the data measured by the finger-tip sensor, we can obtain the closest function to the respiratory signal by filtering out the frequencies below 0.1 Hz and beyond 0.5 Hz from LED signals. Note that this can be done by applying a low-pass filter with threshold 0.5 Hz and a high-pass filter with threshold 0.1 Hz. See Figure 3.4.

As mentioned in the earlier chapters, there are two LEDs (red and IR) in the

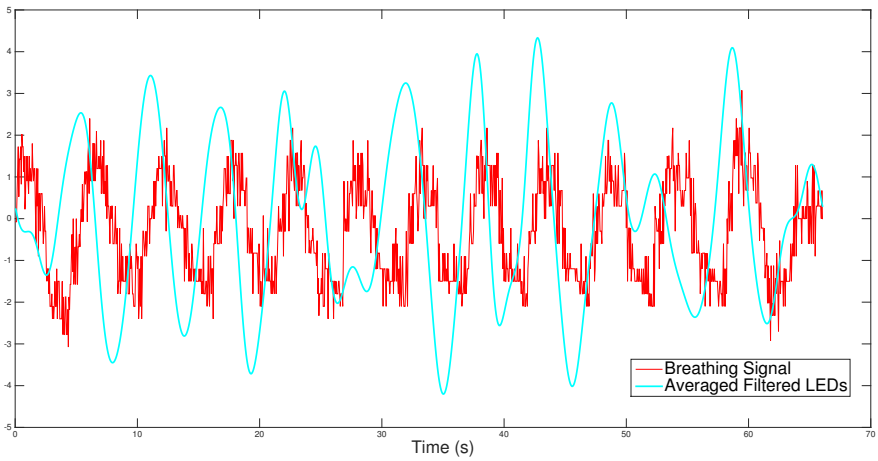

Figure 3.5: Average of filtered and amplitude-scaled LED signals compared to the breathing signal.

finger-tip component of the sensor. Both of these LEDs captures some information about the respiratory signal. Thus we apply our filters to both LEDs and the final breathing signal reconstruction will be obtained from averaging the two filtered LED signals. See Figure 3.5.

In this thesis, we work with ideal filters for breathing signal reconstruction. It turns out that using other filters like Butterworth does not give us the desired results due to their approximate nature. Also notice that since our objective is to reconstruct the breathing signal in real-time, we must be able to do this filtering in online fashion despite the fact that DFT and most of filtering-related algorithms need the entire signal offline as one batch of data. In the next section, we explore techniques that can make the filtering algorithms online.

3.3 Online Window-Based Algorithms

The filtering algorithms and in particular the ideal filter only work offline which means the entire signal must be given. By using a *sliding window*, these algorithms can be made online. Concretely, we work with a sufficiently large window of N sampled datapoints from the signal ending at the current time. In each iteration, the filtering algorithm is applied to the current window. In other words, we approximate the output of the algorithm given the entire signal by only focusing on the subset of N points of the entire signal from the current window. Then, we shift the window forward and reiterate. Observe that since the sampling frequency might be very high, in order to achieve an efficient online algorithm, we may need to shift our sliding window by more than one point. Throughout this section, we assume that we want to shift our window by Δ points.

In the remaining of this section, two online algorithms are represented that can be used towards filtering out unrelated frequencies in this work. In the first algorithm, we propose a $O(N + N \log \Delta)$ procedure to find the DFT of the current window given the DFT of previous window. As we will discuss in 3.3.2 in more details, finding the DFT of the window is a sub-module of performing the ideal filtering process, as it must followed by finding the inverse DFT. In other words, this algorithm does not perform

the entire ideal filtering process and gives us the frequency spectrum given the signal in the time domain. Nevertheless, applying an efficient sliding window DFT algorithm can expedite the ideal filtering process. The best algorithm in the literature [12] can do this task in $O(N\Delta)$ time complexity by successively shifting the window by one point and finding its associated DFT each time (we will discuss it in 3.3.1 by addressing the problem for $\Delta = 1$). In the second algorithm, we propose a setting in which the entire filtering process can be done online in $O(N\Delta)$ at each iteration given the filtered signal from previous window. This means as long as $\Delta = O(\log N)$, we can use this technique instead of doing FFT and Inverse-FFT from scratch for the window in each iteration.

3.3.1 Fast Sliding Window DFT

FFT is an algorithm to find Discrete Fourier Transform: given N equally spaced samples from a signal, it outputs a spectrum of N complex sinusoids ordered by their frequencies whose combination results in the same sample values. As we discussed earlier in this chapter, Discrete Fourier Transform is essentially a multiplication of a $N \times N$ matrix and a N-dimensional vector:

$$\begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{N-1} \\ & & \vdots & \\ 1 & \omega^j & \cdots & \omega^{j(N-1)} \\ & & \vdots & \\ 1 & \omega^{N-1} & \cdots & \omega^{(N-1)^2} \end{bmatrix}}_{N \times N} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

in which $\omega = e^{-\frac{2\pi i}{N}}$ and $x = [x_0, x_1, \ldots, x_{N-1}]^T$. FFT does this multiplication in $O(N \log N)$ using divide and conquer strategy [7].

Given the DFT of a window of N samples, we are interested in finding the DFT of the shifted window by Δ samples efficiently. This problem is called *sliding window DFT*. To this purpose, we propose *Fast Sliding Window DFT algorithm* which is a divide-and-conquer algorithm inspired by FFT itself and solves the problem in $O(N + N \log \Delta)$ time complexity.

First we focus on $\Delta = 1$ and prove that sliding window DFT can be done in $O(N)$ time [12]. The problem is to find:

$$\begin{bmatrix} X_0^{(1)} \\ X_1^{(1)} \\ \vdots \\ X_{N-1}^{(1)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{N-1} \\ & & \vdots & \\ 1 & \omega^j & \cdots & \omega^{j(N-1)} \\ & & \vdots & \\ 1 & \omega^{N-1} & \cdots & \omega^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_N \end{bmatrix}$$

given:

$$\begin{bmatrix} X_0^{(0)} \\ X_1^{(0)} \\ \vdots \\ X_{N-1}^{(0)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{N-1} \\ & & \vdots & \\ 1 & \omega^j & \cdots & \omega^{j(N-1)} \\ & & \vdots & \\ 1 & \omega^{N-1} & \cdots & \omega^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_{N-1} \end{bmatrix}$$

Observe that for all $k \in \{0, 1, \ldots, N-1\}$:

$$\begin{aligned} X_k^{(1)} &= \sum_{n=0}^{N-1} \omega^{kn} x_{n+1} \\ &= \omega^{-k} \sum_{n=0}^{N-1} \omega^{k(n+1)} x_{n+1} \\ &= \omega^{-k} \sum_{n=1}^{N} \omega^{kn} x_n \\ &= \omega^{-k} \left(X_k^{(0)} - x_0 + \omega^{kN} x_N \right) \\ &= \omega^{-k} \left(X_k^{(0)} - x_0 + x_N \right) \end{aligned}$$

$$\longrightarrow X_k^{(1)} = \omega^{-k}\left(X_k^{(0)} - x_0 + x_N\right) \tag{3.1}$$

where the last equation follows from the fact that $\omega^N = 1$. If we do this update for all N indices of X, then the sliding window DFT can be done in $O(N)$ time.

Now we assume that $1 < \Delta < N$ and both Δ and N are powers of two. We show that sliding window DFT can be done in $O(N \log \Delta)$ time. The task is to find:

$$\begin{bmatrix} X_0^{(\Delta)} \\ X_1^{(\Delta)} \\ \vdots \\ X_{N-1}^{(\Delta)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{N-1} \\ & & \vdots & \\ 1 & \omega^j & \cdots & \omega^{j(N-1)} \\ & & \vdots & \\ 1 & \omega^{N-1} & \cdots & \omega^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x_\Delta \\ x_{\Delta+1} \\ \cdots \\ x_{N+\Delta-1} \end{bmatrix}$$

given:

$$\begin{bmatrix} X_0^{(0)} \\ X_1^{(0)} \\ \vdots \\ X_{N-1}^{(0)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{N-1} \\ & & \vdots & \\ 1 & \omega^j & \cdots & \omega^{j(N-1)} \\ & & \vdots & \\ 1 & \omega^{N-1} & \cdots & \omega^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_{N-1} \end{bmatrix}$$

Observe that for each $k \in \{0, 1, \ldots, N-1\}$, using Equation 3.1 repeatedly, we can

obtain:

$$
\begin{aligned}
X_k^{(\Delta)} &= \omega^{-k}\left(X_k^{(\Delta-1)} - x_{\Delta-1} + x_{N+\Delta-1}\right) \\
&= \omega^{-k}\left(\left(\omega^{-k}\left(X_k^{(\Delta-2)} - x_{\Delta-2} + x_{N+\Delta-2}\right)\right) - x_{\Delta-1} + x_{N+\Delta-1}\right) \\
&= \omega^{-2k} X_k^{(\Delta-2)} + \omega^{-2k}(x_{N+\Delta-2} - x_{\Delta-2}) + \omega^{-k}(x_{N+\Delta-1} - x_{\Delta-1}) \\
&\cdots \\
&= \omega^{-\Delta k} X_k^{(0)} + \sum_{n=0}^{\Delta-1} \omega^{-(\Delta-n)k}(x_{N+n} - x_n) \\
&= \omega^{-\Delta k}\left(X_k^{(0)} + \sum_{n=0}^{\Delta-1} \omega^{nk} x'_n\right)
\end{aligned}
$$

where in the last line $x'_n = x_{N+n} - x_n$ for all $n \in \{0, \ldots, \Delta - 1\}$. Note that the above expression can also be written as below:

$$
X_k^{(\Delta)} = \omega^{-\Delta k}\left(X_k^{(0)} + \begin{bmatrix} 1 & \omega^k & \cdots & \omega^{k(\Delta-1)} \end{bmatrix} \begin{bmatrix} x'_0 \\ x'_1 \\ \vdots \\ x'_{\Delta-1} \end{bmatrix}\right)
$$

Therefore:

$$\underbrace{\begin{bmatrix} X_0^{(\Delta)} \\ X_1^{(\Delta)} \\ \vdots \\ X_{N-1}^{(\Delta)} \end{bmatrix}}_{N\times 1} = \underbrace{\begin{bmatrix} 1 \\ \omega^{-\Delta} \\ \omega^{-2\Delta} \\ \cdots \\ \omega^{-(N-1)\Delta} \end{bmatrix}}_{N\times 1} \otimes \underbrace{\left(\underbrace{\begin{bmatrix} X_0^{(0)} \\ X_1^{(0)} \\ \vdots \\ X_{N-1}^{(0)} \end{bmatrix}}_{N\times 1} + \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{\Delta-1} \\ & & \vdots & \\ 1 & \omega^{j} & \cdots & \omega^{j(\Delta-1)} \\ & & \vdots & \\ 1 & \omega^{N-1} & \cdots & \omega^{(N-1)(\Delta-1)} \end{bmatrix}}_{N\times\Delta} \underbrace{\begin{bmatrix} x'_0 \\ x'_1 \\ \vdots \\ x'_{\Delta-1} \end{bmatrix}}_{\Delta\times 1} \right)}_{N\times 1}$$

In which $\otimes$ is element-wise vector multiplication. Observe that once we find the matrix-vector multiplication (the $N \times \Delta$ matrix and the Δ-dimensional vector) in the expression above, we can find the RHS in $O(N)$ time since it only consists of element-wise vector multiplication and addition. From now on, we focus on finding the matrix-vector multiplication in an efficient way, i.e., $O(N \log \Delta)$. Thus our task is to compute the following:

$$\overbrace{\underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{\Delta-1} \\ & & \vdots & \\ 1 & \omega^{j} & \cdots & \omega^{j(\Delta-1)} \\ & & \vdots & \\ 1 & \omega^{N-1} & \cdots & \omega^{(N-1)(\Delta-1)} \end{bmatrix}}_{N\times\Delta}}^{M(N,\Delta)} \overbrace{\underbrace{\begin{bmatrix} x'_0 \\ x'_1 \\ \vdots \\ x'_{\Delta-1} \end{bmatrix}}_{\Delta\times 1}}^{\mathbf{x}'}$$

As one can notice, this problem is similar to the original DFT problem except the fact that we are dealing with a rectangular matrix (of size $N \times \Delta$) instead of a square matrix (of size $N \times N$). We apply the similar divide and conquer technique that is used in FFT. In fact, the potential for using divide and conquer strategy in this matrix-vector multiplication becomes apparent when $M(N, \Delta)$'s columns are segregated into evens and odds (See Figure 3.6).

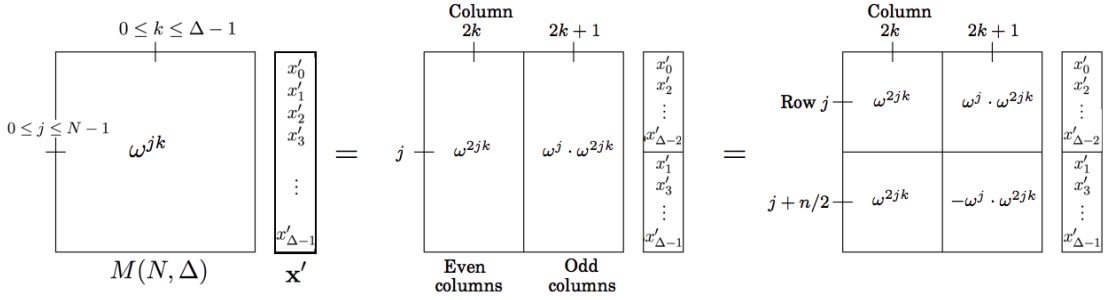

Figure 3.6: Segregating columns into evens and odds (Figure inspired from [7]).

Note that the entries in the bottom half of the matrix are simplified using $\omega^{N/2} = -1$ and $\omega^N = 1$. Notice that the top left $N/2 \times N/2$ submatrix is $M(\frac{N}{2}, \frac{\Delta}{2})$, as is the one on the bottom left. And the top and bottom right submatrices are almost the same as $M(\frac{N}{2}, \frac{\Delta}{2})$, but with their jth rows multiplied through by ω^j and $-\omega^j$, respectively (See Figure 3.7).

Thus the product of $M(N, \Delta)$ with vector $(x'_0, x'_1, \ldots, x'_{\Delta-1})^T$ can be expressed in terms of two products: the product of $M(N/2, \Delta/2)$ with $(x'_0, x'_2, \ldots, x'_{\Delta-2})^T$ and with $(x'_1, x'_3, \ldots, x'_{\Delta-1})^T$. Therefore, assuming that $T(N, \Delta)$ is the time complexity of

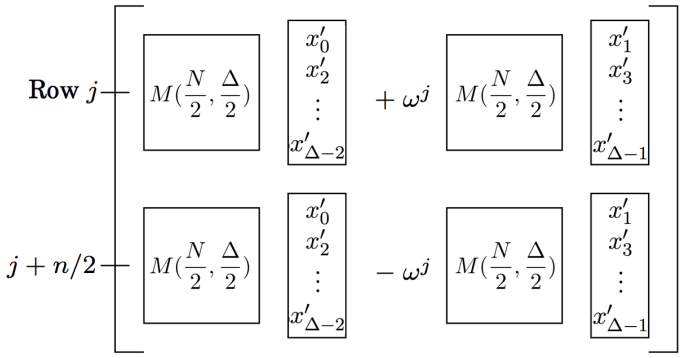

Figure 3.7: Divide-and-conquer (Figure inspired from [7]).

the algorithm with window of length N and shifting parameter Δ, we have:

$$T(N, \Delta) = 2\,T(N/2, \Delta/2) + O(N)$$

Now in order to find the time complexity, we obtain:

$$\begin{aligned} T(N, \Delta) &= 2\,T(N/2, \Delta/2) + cN = 2\,(2\,T(N/4, \Delta/4) + cN/2) + cN \\ &= 2^2\,T(N/2^2, \Delta/2^2) + 2cN \\ &= \cdots \\ &= 2^k\,T(N/2^k, \Delta/2^k) + kcN \end{aligned}$$

Replacing $k = \log \Delta$, we have:

$$T(N, \Delta) = \Delta \times T(N/\Delta, 1) + c\,N \log \Delta$$

Note that for all A, $T(A, 1) = O(A)$, since the matrix product will be just a

scalar multiplication. Thus:

$$\begin{aligned} T(N, \Delta) &= \Delta \times T(N/\Delta, 1) + c\, N \log \Delta \\ &= \Delta \times c'\, (N/\Delta) + c\, N \log \Delta \\ &= c'\, N + c\, N \log \Delta \\ &= O(N \log \Delta) \end{aligned}$$

3.3.2 Online Sliding Window Ideal Filtering

In this thesis, we are interested in specific type of ideal filtering. Given a signal, suppose we want to apply ideal low- and high-pass filters in order to maintain only the frequencies in an specific interval $[f_{min}, f_{max}]$ and zero out the rest of frequencies. Let x and y be the original and filtered signal, respectively, both of size N. Also denote their associated frequency spectrum by X and Y, respectively. Observe that, if we want to do this task offline, then this can be done in $O(N \log N)$ time:

1. Using FFT, go from x (time domain) to X (frequency domain): $O(N \log N)$ time
2. Zero out the unrelated frequencies from X to obtain Y: $O(N)$ time
3. Using Inverse-FFT, go from Y (frequency domain) to y (time domain): $O(N \log N)$ time

Now consider the ideal filtering problem in online fashion meaning that the equally spaced samples are being received in real-time. In order to ideally filter the signal online, one can use a sufficiently large window of samples – say N – and as this

window is shifted forward, the ideally filtered signal of the given window can be found in each iteration. Using the three steps explained above for each iteration, we can find the filtered signal in each window in $O(N \log N)$ once the window is shifted.

Before going to sliding window ideal filtering, first we collapse the three steps mentioned earlier into one. Although, the collapsed form is much less efficient for batch version ($O(N^2)$ instead of $O(N \log N)$), it will introduce a way to do the task in online fashion through a sliding window. Assuming $w = e^{-\frac{2\pi i}{N}}$, we have:

$$\begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{N-1} \\ & & \vdots & \\ 1 & \omega^j & \cdots & \omega^{j(N-1)} \\ & & \vdots & \\ 1 & \omega^{N-1} & \cdots & \omega^{(N-1)^2} \end{bmatrix}}_{N \times N} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

Assume that our desired frequency interval $[f_{min}, f_{max}]$ corresponds to the indices a to b in X. Therefore:

$$\underbrace{Y}_{N \times 1} = \underbrace{\mathbf{1}_{[a,b]}}_{N \times 1} \otimes \underbrace{X}_{N \times 1}$$

in which $\otimes$ is element-wise vector multiplication and $\mathbf{1}_{[a,b]}$ is a column vector of size N whose ith element is one if $a \leq i \leq b$ and zero otherwise. Finally:

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \frac{1}{N} \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \cdots & \omega^{-(N-1)} \\ & & \vdots & \\ 1 & \omega^{-j} & \cdots & \omega^{-j(N-1)} \\ & & \vdots & \\ 1 & \omega^{-(N-1)} & \cdots & \omega^{-(N-1)^2} \end{bmatrix}}_{N \times N} \begin{bmatrix} Y_0 \\ Y_1 \\ \vdots \\ Y_{N-1} \end{bmatrix}$$

Now putting everything together we can write y in terms of x:

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \frac{1}{N} \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \cdots & \omega^{-(N-1)} \\ & & \vdots & \\ 1 & \omega^{-j} & \cdots & \omega^{-j(N-1)} \\ & & \vdots & \\ 1 & \omega^{-(N-1)} & \cdots & \omega^{-(N-1)^2} \end{bmatrix}}_{N \times N}$$

$$\times \left(\mathbf{1}_{[a,b]} \otimes \left(\underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega & \cdots & \omega^{N-1} \\ & & \vdots & \\ 1 & \omega^{j} & \cdots & \omega^{j(N-1)} \\ & & \vdots & \\ 1 & \omega^{N-1} & \cdots & \omega^{(N-1)^2} \end{bmatrix}}_{N \times N} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix} \right) \right)$$

$$= \frac{1}{N} \underbrace{\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \cdots & \omega^{-(N-1)} \\ & & \vdots & \\ 1 & \omega^{-j} & \cdots & \omega^{-j(N-1)} \\ & & \vdots & \\ 1 & \omega^{-(N-1)} & \cdots & \omega^{-(N-1)^2} \end{bmatrix}}_{N \times N} \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ & & \vdots & \\ 0 & 0 & \cdots & 0 \\ 1 & \omega^{a} & \cdots & \omega^{a(N-1)} \\ & & \vdots & \\ 1 & \omega^{j} & \cdots & \omega^{j(N-1)} \\ & & \vdots & \\ 1 & \omega^{b} & \cdots & \omega^{b(N-1)} \\ 0 & 0 & \cdots & 0 \\ & & \vdots & \\ 0 & 0 & \cdots & 0 \end{bmatrix}}_{N \times N} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

$$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{N \times N}$$

$$\longrightarrow \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix} = \frac{1}{N} \underbrace{\underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ \omega^{-a} & \cdots & \omega^{-b} \\ & \vdots & \\ \omega^{-aj} & \cdots & \omega^{-bj} \\ & \vdots & \\ \omega^{-a(N-1)} & \cdots & \omega^{-b(N-1)} \end{bmatrix}}_{N \times (b-a+1)} \underbrace{\begin{bmatrix} 1 & \omega^{a} & \cdots & \omega^{a(N-1)} \\ & & \vdots & \\ 1 & \omega^{j} & \cdots & \omega^{j(N-1)} \\ & & \vdots & \\ 1 & \omega^{b} & \cdots & \omega^{b(N-1)} \end{bmatrix}}_{(b-a+1) \times N} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}}_{N \times N}$$

Define P as the matrix multiplication below:

$$P = \underbrace{\underbrace{\begin{bmatrix} 1 & \cdots & 1 \\ \omega^{-a} & \cdots & \omega^{-b} \\ & \vdots & \\ \omega^{-aj} & \cdots & \omega^{-bj} \\ & \vdots & \\ \omega^{-a(N-1)} & \cdots & \omega^{-b(N-1)} \end{bmatrix}}_{N\times(b-a+1)} \underbrace{\begin{bmatrix} 1 & \omega^{a} & \cdots & \omega^{a(N-1)} \\ & & \vdots & \\ 1 & \omega^{j} & \cdots & \omega^{j(N-1)} \\ & & \vdots & \\ 1 & \omega^{b} & \cdots & \omega^{b(N-1)} \end{bmatrix}}_{(b-a+1)\times N}}_{N\times N}$$

We can find the value of an arbitrary element in P which is a $N \times N$ matrix. Let P_{jk} be the element of matrix P located at row j and column k for arbitrary $0 \leq k, j \leq N-1$ [1]. Hence, we can obtain:

$$P_{jk} = \begin{bmatrix} \omega^{-aj} & \cdots & \omega^{-bj} \end{bmatrix} \begin{bmatrix} \omega^{ak} \\ \vdots \\ \omega^{bk} \end{bmatrix} = \sum_{r=a}^{b} \omega^{(k-j)r}$$

$$\longrightarrow P_{jk} = \begin{cases} b-a+1 & j = k \\ \frac{\omega^{(k-j)(b+1)} - \omega^{(k-j)a}}{\omega^{k-j}-1} & k \neq j \end{cases}$$

Note that P is a structured matrix. Concretely:

$$P_{jk} = f_{j-k}, \quad \text{in which} \quad f_q = \begin{cases} b-a+1 & q = 0 \\ \frac{\omega^{q(b+1)} - \omega^{qa}}{\omega^{q}-1} & q \neq 0 \end{cases}$$

[1] Here for simplicity, we use $\{0, 1, \ldots, N-1\}$ for indexing the rows and columns of P

Observe that since $0 \leq k, j \leq N-1$, thus $-N+1 \leq q \leq N-1$. Also for $q \neq 0$, since ω is a Nth unit root, we have:

$$\begin{aligned} f_{N+q} &= \frac{\omega^{(N+q)(b+1)} - \omega^{(N+q)a}}{\omega^{(N+q)} - 1} \\ &= \frac{\omega^{q(b+1)}\omega^{N(b+1)} - \omega^{qa}\omega^{Na}}{\omega^{q}\omega^{N} - 1} \\ &= \frac{\omega^{q(b+1)} - \omega^{qa}}{\omega^{q} - 1} = f_q \end{aligned}$$

This means that the subscript of f is significant upto modulo N. Therefore:

$$P = \underbrace{\begin{bmatrix} f_0 & f_{N-1} & f_{N-2} & \cdots & f_2 & f_1 \\ f_1 & f_0 & f_{N-1} & \cdots & f_3 & f_2 \\ f_2 & f_1 & f_0 & \cdots & f_4 & f_3 \\ & \vdots & & \vdots & & \\ f_{N-2} & f_{N-3} & f_{N-4} & \cdots & f_0 & f_{N-1} \\ f_{N-1} & f_{N-2} & f_{N-3} & \cdots & f_1 & f_0 \end{bmatrix}}_{N \times N}$$

Therefore we can perform the ideal filtering in one collapsed $O(N^2)$-step of $y = \frac{1}{N} Px$ in which P is computed as above. Concretely:

$$\forall k \in \{0, 1, \ldots, N-1\} \qquad y_k = \frac{1}{N} \sum_{j=0}^{N-1} f_{k-j}\, x_j$$

Now that we have obtained the collapsed step for ideal filtering, let us focus on online sliding window ideal filtering. Given the ideally filtered signal of a window of N samples, we are interested in finding the ideally filtered signal of the shifted window by Δ samples efficiently. This problem is called *sliding window ideal filtering*. To this

purpose, we propose a method which uses the collapsed step for ideal filtering and is able to do it online in $O(N\Delta)$ time complexity.

First we focus on $\Delta = 1$ and prove that sliding window ideal filtering can be done in $O(N)$ time. The problem is to find:

$$\begin{bmatrix} y_0^{(1)} \\ y_1^{(1)} \\ \vdots \\ y_{N-1}^{(1)} \end{bmatrix} = \frac{1}{N} P \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}$$

Given:

$$\begin{bmatrix} y_0^{(0)} \\ y_1^{(0)} \\ \vdots \\ y_{N-1}^{(0)} \end{bmatrix} = \frac{1}{N} P \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

Observe that for all $k \in \{0, 1, \ldots, N-1\}$:

$$\begin{aligned} y_k^{(1)} &= \frac{1}{N} \sum_{j=0}^{N-1} f_{k-j} x_{j+1} = \frac{1}{N} \sum_{j=1}^{N} f_{k-j+1} x_j \\ &= \frac{1}{N} \left(\sum_{j=0}^{N-1} f_{k-j+1} x_j \right) - \frac{1}{N} f_{k+1} x_0 + \frac{1}{N} f_{k-N+1} x_N \\ &= \frac{1}{N} \left(\sum_{j=0}^{N-1} f_{(k+1)-j} x_j \right) + \frac{1}{N} f_{k+1} (x_N - x_0) \qquad (f_{k+1-N} = f_{k+1}) \\ &= y_{k^+}^{(0)} + \frac{1}{N} f_{k+1} (x_N - x_0) \end{aligned}$$

$$\longrightarrow y_k^{(1)} = y_{k^+}^{(0)} + \frac{1}{N} f_{k+1} (x_N - x_0) \tag{3.2}$$

In which k^+ is defined to be $k+1$ modulo N. If we do this update for all N indices, then the sliding window ideal filtering can be done in $O(N)$.

Now we assume that $1 < \Delta < N$. Then the task of sliding window ideal filtering is to find:

$$\begin{bmatrix} y_0^{(\Delta)} \\ y_1^{(\Delta)} \\ \vdots \\ y_{N-1}^{(\Delta)} \end{bmatrix} = \frac{1}{N} P \begin{bmatrix} x_\Delta \\ x_{\Delta+1} \\ \vdots \\ x_{N+\Delta-1} \end{bmatrix}$$

Given:

$$\begin{bmatrix} y_0^{(0)} \\ y_1^{(0)} \\ \vdots \\ y_{N-1}^{(0)} \end{bmatrix} = \frac{1}{N} P \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

Observe that for each k, using Equation 3.2 repeatedly, we can obtain:

$$\begin{aligned} y_k^{(\Delta)} &= y_{k^+}^{(\Delta-1)} + \frac{1}{N} f_{k+1}(x_{N+\Delta-1} - x_{\Delta-1}) \\ &= y_{k^{++}}^{(\Delta-2)} + \frac{1}{N} f_{k+2}(x_{N+\Delta-2} - x_{\Delta-2}) + \frac{1}{N} f_{k+1}(x_{N+\Delta-1} - x_{\Delta-1}) \\ &\vdots \\ &= y_{k^{\Delta+}}^{(0)} + \frac{1}{N} \sum_{j=1}^{\Delta} f_{k+j}(x_{N+\Delta-j} - x_{\Delta-j}) \end{aligned}$$

In which $k^{\Delta+}$ is defined to be $k+\Delta$ modulo N. Note that the equation above can be also be written as below:

$$y_k^{(\Delta)} = y_{k^{\Delta+}}^{(0)} + \frac{1}{N}\begin{bmatrix} f_{k+\Delta} & f_{k+\Delta-1} & \cdots & f_{k+1} \end{bmatrix}\begin{bmatrix} x_N - x_0 \\ x_{N+1} - x_1 \\ \vdots \\ x_{N+\Delta-1} - x_{\Delta-1} \end{bmatrix}$$

Therefore:

$$y^{(\Delta)} = y_{\Delta^+}^{(0)} + \frac{1}{N}\underbrace{\begin{bmatrix} f_\Delta & f_{\Delta-1} & f_{\Delta-2} & \cdots & f_2 & f_1 \\ f_{\Delta+1} & f_\Delta & f_{\Delta-1} & \cdots & f_3 & f_2 \\ f_{\Delta+2} & f_{\Delta+1} & f_\Delta & \cdots & f_4 & f_3 \\ & \vdots & & \vdots & & \\ f_{N+\Delta-2} & f_{N+\Delta-3} & f_{N+\Delta-4} & \cdots & f_0 & f_{N-1} \\ f_{N+\Delta-1} & f_{N+\Delta-2} & f_{N+\Delta-3} & \cdots & f_1 & f_0 \end{bmatrix}}_{N\times\Delta}\underbrace{\begin{bmatrix} x_N - x_0 \\ x_{N+1} - x_1 \\ \vdots \\ x_{N+\Delta-1} - x_{\Delta-1} \end{bmatrix}}_{\Delta\times 1}$$

In which $y_{\Delta^+}^{(0)}$ is $y^{(0)}$ which is circularly shifted Δ units upward. So in order to find $y^{(\Delta)}$, you need to do shifting on $y^{(0)}$ and do a multiplication between a $N\times\Delta$ matrix and a Δ dimensional vector which costs $O(N)$ and $O(N\Delta)$, respectively. Thus we can find the ideally filtered signal in the new window given the ideally filtered signal in the previous one in $O(N\Delta)$ time complexity.

Chapter 4

Signal Reconstruction by Neural Networks

As mentioned in earlier chapters, in order to predict the breathing rate, we first reconstruct the breathing signal from LED signals, then detect the peaks of the reconstructed breathing signal each of which denotes an occurrence of a breath. In previous chapter, we studied the breathing signal reconstruction using filtering. Another way of reconstructing the breathing signal using LED signals is to use machine learning algorithms, i.e., to see the problem as a regression problem. In this approach, we can look the LED signals as the feature vectors and the breathing signal as the target value.

In this chapter, we start with how to prepare proper dataset for this regression problem. Then we discuss how the breathing signal reconstruction can be done using neural networks. After that we discuss the regularization techniques being used in

our model in order to avoid over-fitting. Moreover, we also explore two additional techniques as a pre-processing step namely applying moving average filters and using denoising auto-encoders.

4.1 Preparing the Dataset

In order to have the settings of a regression problem, we need to prepare a proper dataset as a pre-processing step. Each point in the dataset consists of a feature vector extracted from the LED signals, and a target value which is basically the value of the breathing signal. Concretely, for the t-th sample value from the breathing signal (t is sufficiently large), we identify $2k$ features — k features from each LED. We consider the sample values of LEDs at times $\{t, t-l, t-2l, , t-(k-1)l\}$ as features (See Figure 4.1). In other words, given the point to predict in breathing signal, we first down-sample l points to one in both LED signals and then take the last k samples ending at current time from each LED signal as features.

First, notice that this dataset is online-friendly. Concretely, once the model is trained, given a real-time stream of LED signals, it is able to do prediction in online fashion after certain time which is $t \geq (k-1) \times l + 1$.

4.1.1 Choosing k and l

As mentioned before, l indicates the amount of down-sampling and k is the number of down-sampled data. Observe that l can be proportionate on sampling fre-

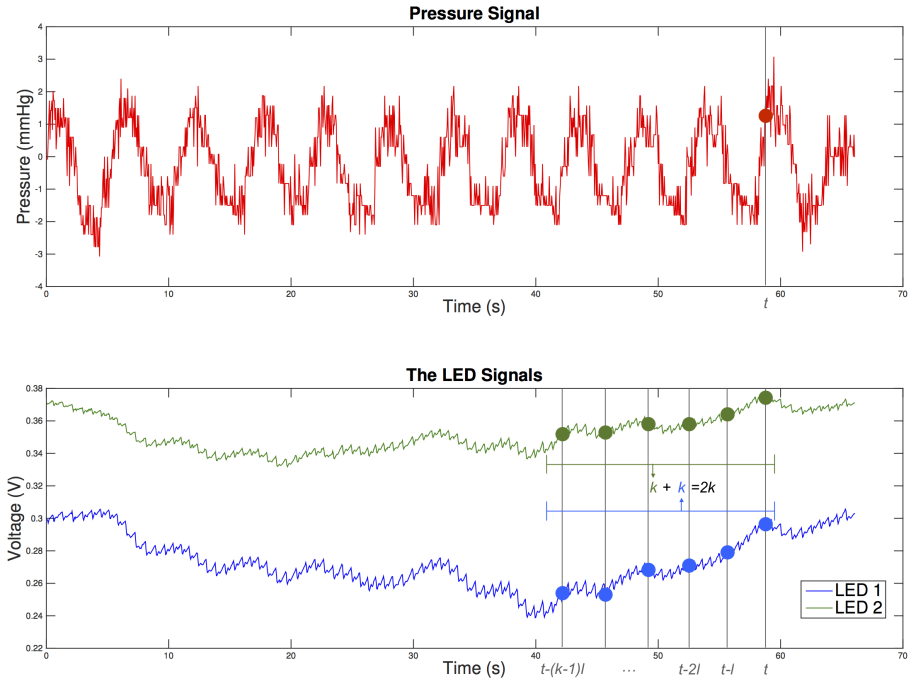

Figure 4.1: The feature vector corresponding to time t consists of down-sampled values of LED signals at times $\{t, t-l, t-2l, , t-(k-1)l\}$.

quency. Also notice that k and l must be tuned in a way that:

- $k \times l$ is a sufficiently large window of last sample values of LED signals.
- l is small enough so that the down-sampling still captures periodicity of both LED signals.
- k is small enough so that we avoid high dimensional space and "*curse of dimensionality*"

In this thesis, we used $k = 12$ and $l = 16$. Note that the base sampling frequency from the device is 20 Hz, and consequently, the time between two consecutive samples is 50 ms. This means that to extract the features for each datapoint, we pick a sample from both LED signals every $800\,ms$ [1] focusing on a window of 8.8 seconds [2] ending at current time.

4.2 Neural Network for Reconstruction

In order to solve this regression problem, we use neural networks [4]. The main reason that we use neural network is the fact that we need to extract non-linear high-level features from the raw data for which there is no straightforward approach to find. These high-level features can be different characteristics of the breathing signal at a given time like the amplitude and phase of the signal. The neurons in the neural network extract those features automatically through the learning phase.

[1] $800 = 16 \times 50$
[2] $8800 = (12 - 1) \times 16 \times 50$

In this thesis, we used shallow neural networks with one or two hidden layers with limited number of nodes. Due to limitation on the size of the data, any excess number of parameters and complexity of the model will lead to over-fitting. Also using too few neurons may under-fit and fail to capture the essence of the data. In particular we tried one hidden layer with 9 neurons, and two hidden layers with 6 and 3 neurons.

4.3 Regularization

Preliminary experiments showed that our model has low training error, but it has a relatively poor performance on the test set on most cases. It seems that since the training sets are not large, our model is prone to over-fitting. To avoid over-fitting, we used several regularization techniques[4]:

1. *Early Stopping:* This procedure is used to control the complexity of a neural network. In training the networks, optimization algorithms are used to decrease the error function through iterations. This error function is defined with respect to the training set which usually causes over-fitting. However, instead of training set, we use an independent set of data, called a *validation set*, based on which we keep track of the error function. This error function often decreases at first and then increases after a certain number of iterations indicating that the model starts to overfit. Thus, in order to have network with good generalization performance, the training can be stopped when the error is minimum with respect to the validation

set.

2. *Adding Regularizer to the Cost Function:* Another way to prevent the network from becoming too complex, is to add regularizer terms to the error function. In this case, we add the mean squared of the weights to the error function as a regularizer. With this regularizer the weights can not get very large and make the model complex, and consequently, minimizing the error function will keep the model simple. Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [1] is used to minimize the regularized error function. BFGS is a second-order method which approximates Newton's algorithm to solve unconstrained non-linear optimization problems.

3. *Training Multiple Times:* The error function in in the neural network has combinatorial number of local minima. Depending on which local minimum we have obtained, the performance of the network on the unseen data might be very different. Therefore training only one single network may have poor accuracy. Thus in our model, we train the neural network multiple times (10 times) — each time with different random initialization — and use the average of the trained network to predict on the test set.

4.4 Additional Techniques

In addition to the regularization techniques mentioned earlier, we also used some more procedures so that the model has a better generalization performance. Here

we introduce filtering and auto-encoders, both of which are pre-processing and pre-training steps.

4.4.1 Moving Average

As we have seen in previous chapters, each LED signal is a mixture of breathing and pulse signal. In this case, the pulse signal is the "noise" which has a higher frequency. As a pre-processing step, we can eliminate the pulse signal from the input LED signals before preparing the dataset (discussed in 4.1). By doing this step, the neural network will learn the regression problem using the smoothed LED signals which is the essential information about the respiratory signal from the finger-tip sensor.

In order to do this smoothing, we can apply a low-pass filter to zero out the higher frequencies related to the pulse signal. However, instead of applying this filter which will cost $O(N \log N)$, we can perform a *moving average* through a pass over the data in $O(N)$ time. Concretely, we shift a sliding window over the signal and compute the average each time. Mathematically, since moving average is a type of convolution, it is equivalent to a low-pass filter which is used in signal processing [17].

The resulting signal is a smoothed version of the original signal (See Figures 4.2 and 4.3). Applying the moving average filter, we can also remove the noise from the breathing signal as well. In that case, we can feed the signals with less noise to the neural network, so that it can predict better on noisy unseen data.

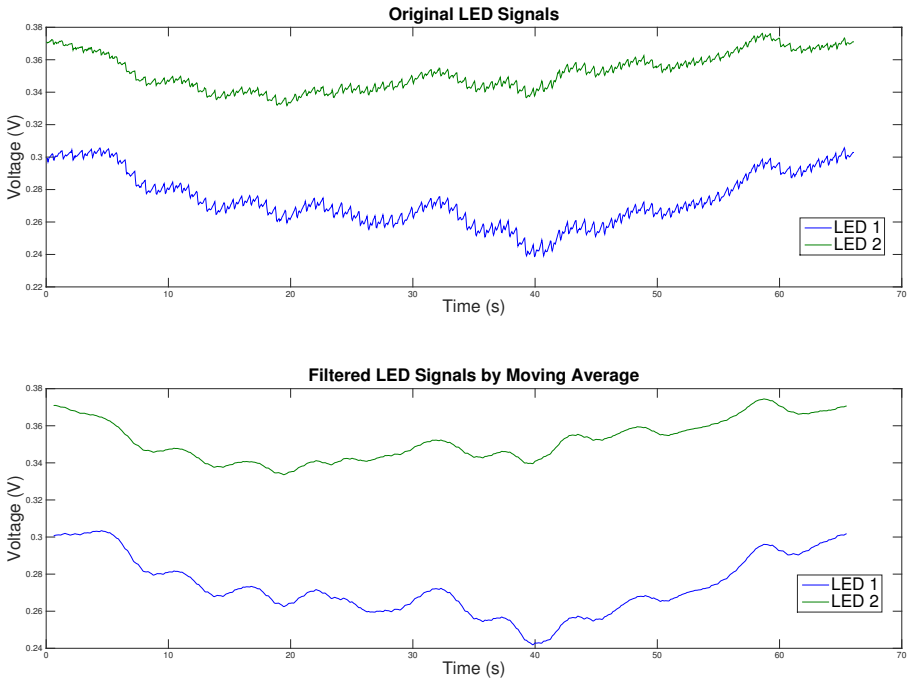

Figure 4.2: The original and filtered LED signals using the moving average with window of 24 samples.

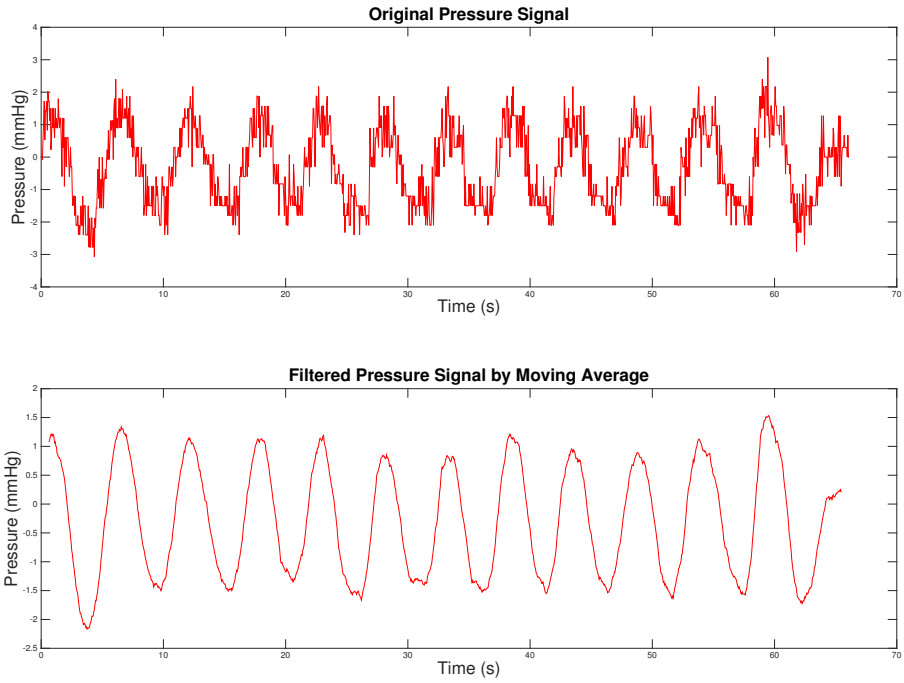

Figure 4.3: The original and filtered breathing signal using the moving average with window of 24 samples.

4.4.2 Auto-Encoders

Auto-encoders [3] were proposed to overcome the difficulties related to learning deep neural networks. The goal of auto-encoders is to map the inputs into high-level intermediate representations, so that it can be used as an initial step of unsupervised learning before training the model. This step is also known as *pre-training* [3].

In each iteration of this unsupervised learning procedure, we are focused on exactly one hidden layer of the network. Concretely, a new neural network with single hidden layer is trained given the same target values as inputs which are coming from the mapping of original inputs based on previous layers. Therefore, in this greedy layer-wise algorithm, we compute the set of weights for the neurons in each layer such that it can reproduce its input. Then, we use these weights as an initialization and proceed with the training phase of neural network as a regular supervised learning step (which is also known as *fine-tuning* [3]).

In this thesis, we perform pre-training to our network by treating the layers of the network as stacked auto-encoders.

4.4.2.1 Denoising Auto-Encoders

Denoising auto-encoders [19] are a variant of ordinary auto-encoders developed from the idea of making the intermediate presentation learned by auto-encoders robust to possible corruption in the input. Similar to ordinary auto-encoders, denois-

ing auto-encoders also work in greedy layer-wise manner; however, they are trained to reconstruct the original inputs given the perturbed input. Like ordinary auto-encoders, denoising auto-encoders can be stacked and pre-trained to initialize the weights of the neural networks.

In this thesis, we also try performing denoising auto-encoders in order to find a robust representation of the inputs. Note that having robust high-level features can help in this case since the inputs are very noisy due to the fact that they are low-frequency breathing signal polluted by high-frequency pulse signal.

Chapter 5

Experiments and Results

For our experiments, 6 volunteers from both genders and different races participated in this research. We performed 3 one-minute trials of measurements on each human subject, that is, we obtained three datasets associated with each person. In each trial, while they had the finger-tip component on their fingers, they breathed normally into the tube [1]. In order to have a machine learning setting, for each human subject, we used two datasets for training and one for testing.

We implemented the algorithms in MATLAB and applied both ideal filtering and neural networks techniques for the data corresponding to each volunteer. Due to several local minima for the cost function in neural networks, in each run of the algorithm, we observed that the reconstructed signals were not the same [2]; however, this

[1] Since breathing into the mouthpiece involved some efforts, we asked them to breath "slowly", so that their efforts would not cause heavy breathing.

[2] As discussed in 4.3, we train multiple neural networks (10 in this case) and use their average for prediction in order to mitigate this problem.

problem did not exist with using filters as a reconstruction approach.

For neural network, we tried 8 different configurations which were the combinations of the three options below:

1. Using one or two hidden layers. [3]

2. Using the Moving Average (MA) as a pre-processing step.

3. Using Denoising Auto-Encoders (AE) as a pre-processing step [4].

In all configurations, we feed the neural networks with raw breathing signal as target, except MA in which smoothed breathing signal is passed as target. Also in all configurations, we train the neural networks with the square error as the loss function.

For ideal filtering, we presume that performing it offline and in batch fashion has a better precision in comparison with doing it online with sliding window. Consequently, we used offline ideal filtering for the sake of these experiments. We observe that, even by doing so, using ideal filtering for signal reconstruction has a worse performance comparing to neural network in most cases.

To evaluate the performance of each signal reconstruction method, we use the *Signal Error Function* (SEF) defined in Definition 1 in Chapter 2 with one-dimensional

[3] We used 9 neurons for one hidden layers, and layers of 6 and 3 neurons for two hidden layers.
[4] This is done after applying the Moving Average step - if exists

absolute and square loss functions:

$$\text{Absolute Loss:} \qquad SEF_{\text{ABS}}(f,g) = \min_{a\in\mathbb{R}^+} \sum_{x\in X} \left|\tilde{f}(x) - a\cdot\tilde{g}(x)\right|$$

$$\text{Square Loss:} \qquad SEF_{\text{SQR}}(f,g) = \min_{a\in\mathbb{R}^+} \sum_{x\in X} \left(\tilde{f}(x) - a\cdot\tilde{g}(x)\right)^2$$

where $\tilde{f}$ and $\tilde{g}$ are the zero-mean signals obtained from centering f and g, respectively. It is worth noting that one may expect that the minimizing "a" in definition above be close to 1 for neural networks; however, in our experiments, it varies from 0.2 to 5.

Tables 5.1 and 5.2 show the performance of neural networks for each of the 8 settings and offline ideal filtering for each human subject in terms of SEF with square and absolute losses. See Appendix for visualized performances of applying both filtering and neural networks to each human subject. Table 5.3 summarizes the performance of each algorithm and setting in terms of square and absolute losses.

As one can observe, working with two hidden layers has a better performance with respect to single hidden layer in neural networks. Also, it seems that using the techniques of the moving average and denoising auto-encoders as pre-processing steps does not help improve the performance in neural network. Based on our experiments, reconstruction using filters is not as competitive in terms of SEF with absolute and square loss. This approach is also not as efficient. In fact, as one can see the visualized performances (see the Appendix) they may over-count the number of inhalations. Nevertheless, ideal filters seem to have a reasonable performance and can serve as a good

Subject	Single Hidden Layer				Two Hidden Layers				Filter
	w/ MA		w/o MA		w/ MA		w/o MA		
	w/ AE	w/o AE	w/ AE	w/o AE	w/ AE	w/o AE	w/ AE	w/o AE	
1	3.844	3.514	3.795	3.712	3.277	**3.078**	3.839	3.434	4.424
2	4.566	5.919	4.254	4.661	4.078	**3.915**	4.033	4.102	4.917
3	0.699	0.671	0.579	0.665	**0.234**	0.345	0.282	0.416	1.795
4	1.222	1.820	0.739	1.206	0.359	0.484	0.224	**0.152**	2.870
5	1.088	1.095	1.273	1.160	0.998	**0.950**	1.166	1.026	1.216
6	2.197	2.222	1.514	1.497	1.516	1.806	**1.341**	1.531	1.953

Table 5.1: Absolute loss of different algorithms and setting for all human subjects. Note that filtering has the worst performance in all but one case.

Subject	Single Hidden Layer				Two Hidden Layers				Filter
	w/ MA		w/o MA		w/ MA		w/o MA		
	w/ AE	w/o AE	w/ AE	w/o AE	w/ AE	w/o AE	w/ AE	w/o AE	
1	21.949	20.811	20.928	20.939	21.444	**20.102**	21.517	20.666	23.367
2	33.847	34.322	31.636	32.417	33.601	33.277	**31.446**	31.559	32.740
3	2.816	**2.132**	2.397	2.373	2.989	2.411	2.269	2.487	4.304
4	8.013	7.074	7.477	7.119	7.322	7.843	**6.581**	7.237	9.304
5	3.398	3.394	3.388	3.396	3.370	3.352	3.391	3.385	**2.381**
6	11.866	11.587	12.121	12.020	11.021	11.226	11.326	11.509	**6.028**

Table 5.2: Square loss of different algorithms and setting for all human subjects.

Loss	Single Hidden Layer				Two Hidden Layers				Filter
	w/ MA		w/o MA		w/ MA		w/o MA		
	w/ AE	w/o AE	w/ AE	w/o AE	w/ AE	w/o AE	w/ AE	w/o AE	
Square	13.648	13.220	12.991	13.044	13.291	13.035	**12.755**	12.807	13.021
Absolute	2.269	2.540	2.026	2.150	**1.744**	1.763	1.814	1.777	2.862

Table 5.3: Average performances of different algorithms and setting.

baseline algorithm to compare against.

Chapter 6

Conclusions

6.1 Overview and Conclusions

In this thesis, inspired by the trend of personalized health-care and wearable sensors, we explore the problem of predicting breathing rate given the data coming from a finger-tip sensor. To solve this problem, we attempted two approaches to reconstruct the respiratory signal.

First, using fixed high and low frequency thresholds, we applied ideal low- and high-pass filters to the LED signals from finger-tip sensor. Despite the reasonable performance, it is not straightforward to make this approach online. In our experience, the resulting online filtering algorithm is either slow or less accurate. Inspired by the problem, we developed the setting for sliding window DFT and ideal filtering. Moreover, we proposed an efficient algorithm for doing sliding window DFT.

In the second approach, we see the problem as a regression problem in machine learning. Using neural networks, once it is trained on the human subject, we were able to approximately predict the breathing rate in real-time. Two hidden layers seemed to be more accurate; however, using moving average filtering and/or denoising auto-encoders does not look to improve the performance significantly.

6.2 Future Work

There are potentially several ways to continue this work in order to improve the results and performance. These possible future works are discussed as follows:

1. *Online Filtering and Machine Learning Techniques:* In this thesis, we attempted to solve the prediction problem by either filtering LED signals or train neural networks, but we did not do a mixture of both. The reason is simply the fact that applying ideal filter to the data in real-time is not known to be efficient. However, perhaps using simpler filters (e.g. moving average) can make it possible to apply filters in online fashion efficiently. In that case, one can also use neural networks or some other model on top of filtering to learn the breathing signal.

2. *Visualizing the Neurons:* Each neuron in neural network represents a non-linear feature in the high dimensional data. Those neurons that appear in layers which are closer to the output represent a higher level feature. By finding the input which maximizes the output of a neuron and somehow visualizing that, one can

see which feature a neuron is representing.

3. *Finding a Better Loss Function for Signals:* The loss functions that we defined in this thesis, even though they are not directly involved in the learning model, have some flaws. For instance, consider a signal like $\hat{y} = x + \sin x$ as a reconstruction of the signal $y = \sin x$. Observe that the peaks of y and $\hat{y}$ are synchronized. Also note that $\hat{y}$ has distinguishable peaks which can be detected by our peak detection algorithm. Nevertheless, it will incur a high loss using either of our loss functions.

4. *Using Data-Dependent Filters:* If we use regular low- and high-pass filters, the filters will remain fixed and will not be able to adapt to the data. Using a filter that can "learn" from data can be very helpful in this context. These filters are known as data-dependent filters which seem to be practical in different applications [9] [5].

Bibliography

[1] M. Avriel. *Nonlinear programming: analysis and methods*. Courier Corporation, 2003.

[2] K.E. Barrett et al. *Ganong's review of medical physiology*. McGraw-Hill Medical Asia, 2010.

[3] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[4] C.M. Bishop et al. *Pattern recognition and machine learning*, volume 4. Springer New York, 2006.

[5] E.A. Byrne and S.W. Porges. Data-dependent filter characteristics of peak-valley respiratory sinus arrhythmia estimation: A cautionary note. *Psychophysiology*, 30(4):397–404, 1993.

[6] H.C. Chen and S.W. Chen. A moving average based filtering system with its application to real-time QRS detection. In *Computers in Cardiology, 2003*, pages 585–588. IEEE, 2003.

[7] S. Dasgupta, C.H. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw-Hill, Inc., 2006.

[8] S. Dua, U.R. Acharya, and P. Dua. *Machine Learning in Healthcare Informatics*. Springer, 2014.

[9] S.R. Fanello, C. Keskin, P. Kohli, S. Izadi, J. Shotton, A. Criminisi, U. Pattacini, and T. Paek. Filter forests for learning data-dependent convolutional kernels. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1709–1716. IEEE, 2014.

[10] M. Guthaus and L. Klein. unpublished data. 2015.

[11] K.A. Hausman and E.B. Merrick. Pulse oximeter, November 28 1989. US Patent 4,883,353.

[12] E. Jacobsen and R. Lyons. The Sliding DFT. *Signal Processing Magazine, IEEE*, 20(2):74–80, 2003.

[13] L. Mason. *Signal processing methods for non-invasive respiration monitoring*. PhD thesis, Department of Engineering Science, University of Oxford, 2002.

[14] G.B. Moody, R.G Mark, M.A. Bump, J.S. Weinstein, A.D Berman, J.E. Mietus, and A.L. Goldberger. Clinical validation of the ECG-derived respiration (EDR) technique. *Group*, 1(3), 1986.

[15] G.B Moody, R.G Mark, A. Zoccola, and S. Mantero. Derivation of respiratory signals from multi-lead ECGs. *Computers in cardiology*, 12(1985):113–116, 1985.

[16] Y.M. Mughal. Decomposing of cardiac and respiratory signals from electrical bio-impedance data using filtering method. In *The International Conference on Health Informatics*, pages 252–255. Springer, 2014.

[17] A.V. Oppenheim, R.W. Schafer, J.R. Buck, et al. *Discrete-time signal processing*, volume 2. Prentice-hall Englewood Cliffs, 1989.

[18] A. Travaglini, C. Lamberti, J. DeBie, and M. Ferri. Respiratory signal derived from eight-lead ECG. In *Computers in Cardiology 1998*, pages 65–68. IEEE, 1998.

[19] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[20] J.E. Whitney II and L. Solomon. Respiration rate signal extraction from heart rate. In *Aerospace/Defense Sensing, Simulation, and Controls*, pages 104–112. International Society for Optics and Photonics, 2001.

Appendix A

Visualized Performance over Human Subjects

In all cases, we used two hidden layers in neural network due its better performance. Since using moving average filter and denoising auto-encoders does not improve the performance significantly, they are not used in the visualized performances below.

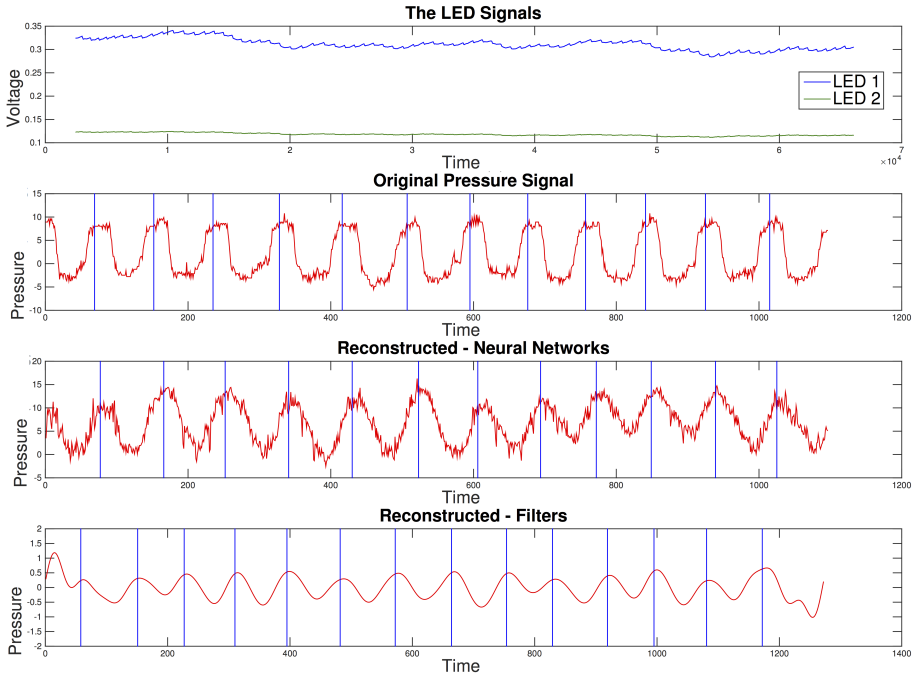

Figure A.1: Comparing the reconstructed breathing signals on human subject 1.

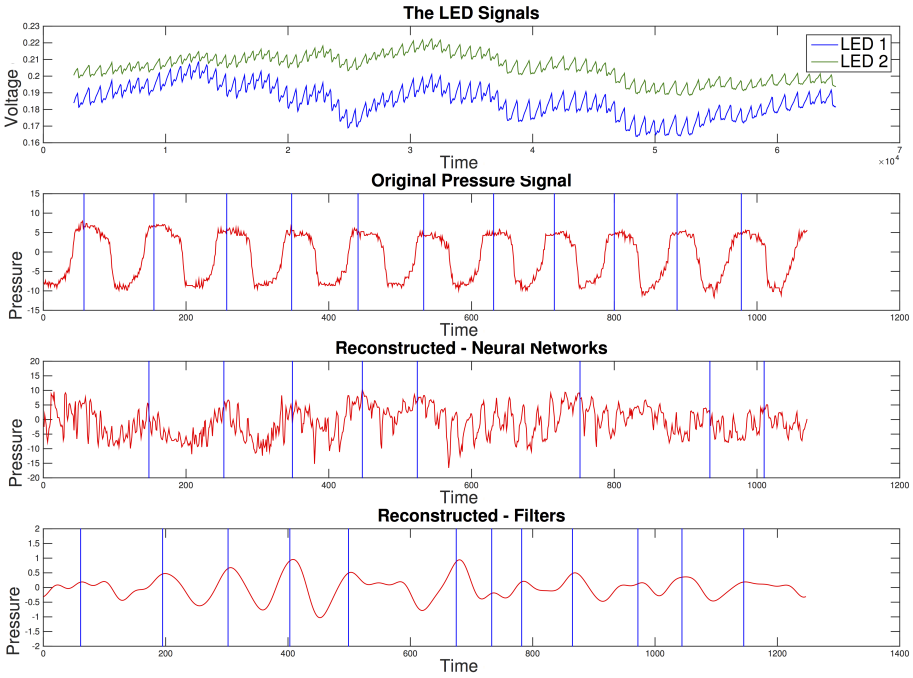

Figure A.2: Comparing the reconstructed breathing signals on human subject 2.

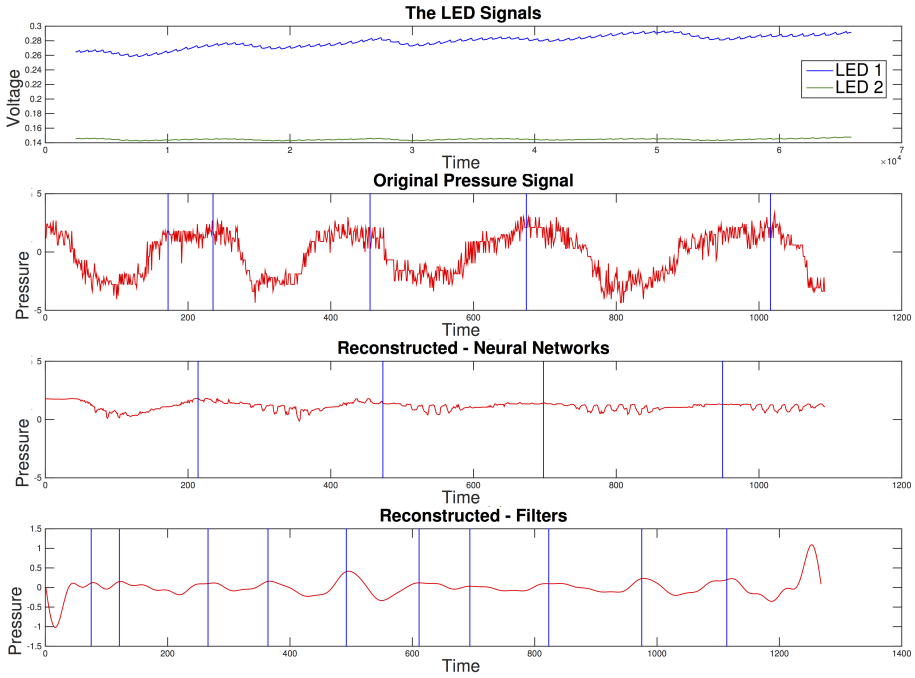

Figure A.3: Comparing the reconstructed breathing signals on human subject 3.

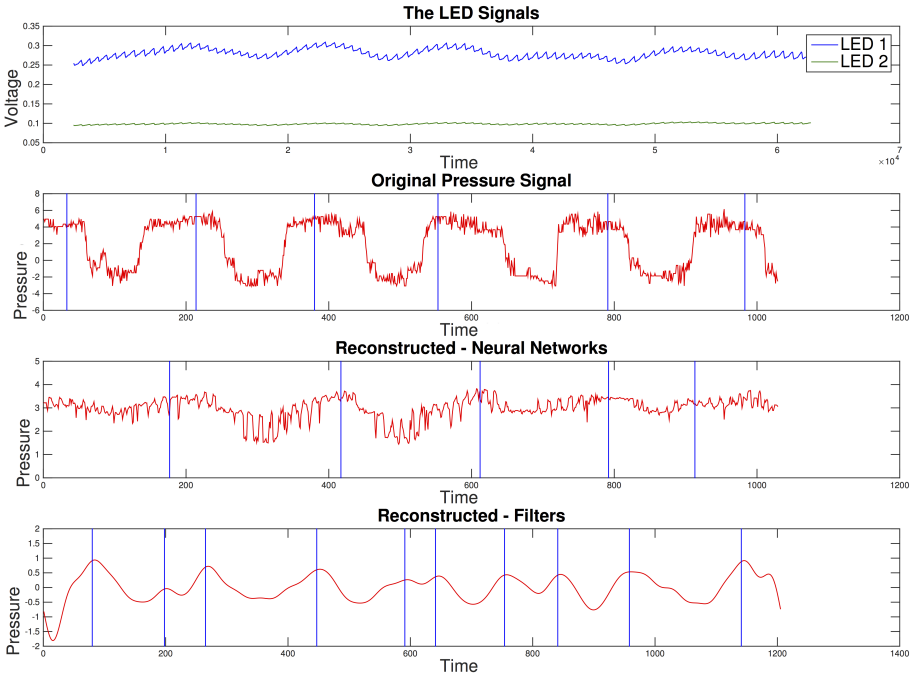

Figure A.4: Comparing the reconstructed breathing signals on human subject 4.

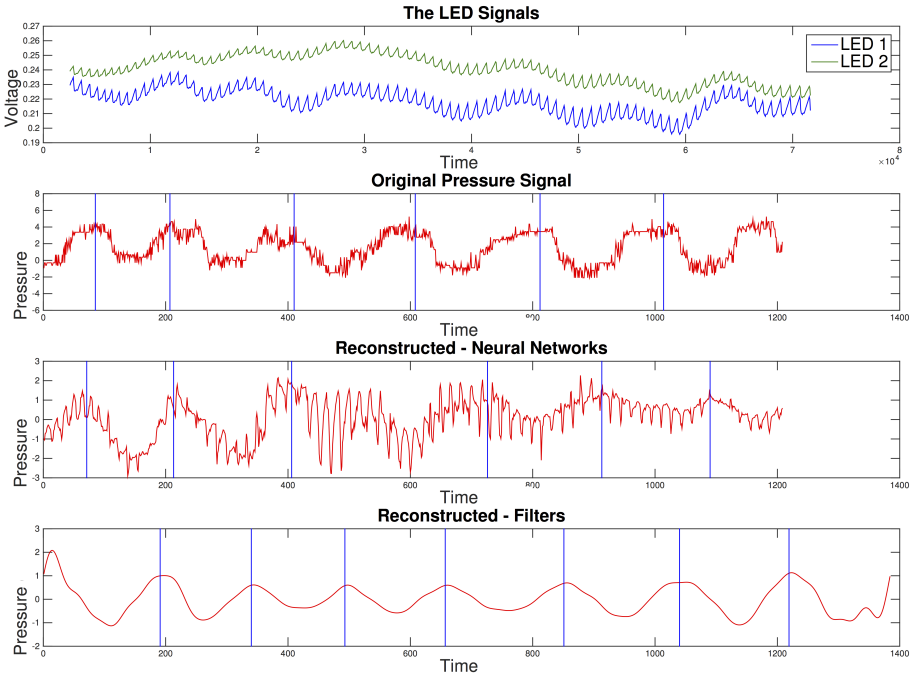

Figure A.5: Comparing the reconstructed breathing signals on human subject 5.

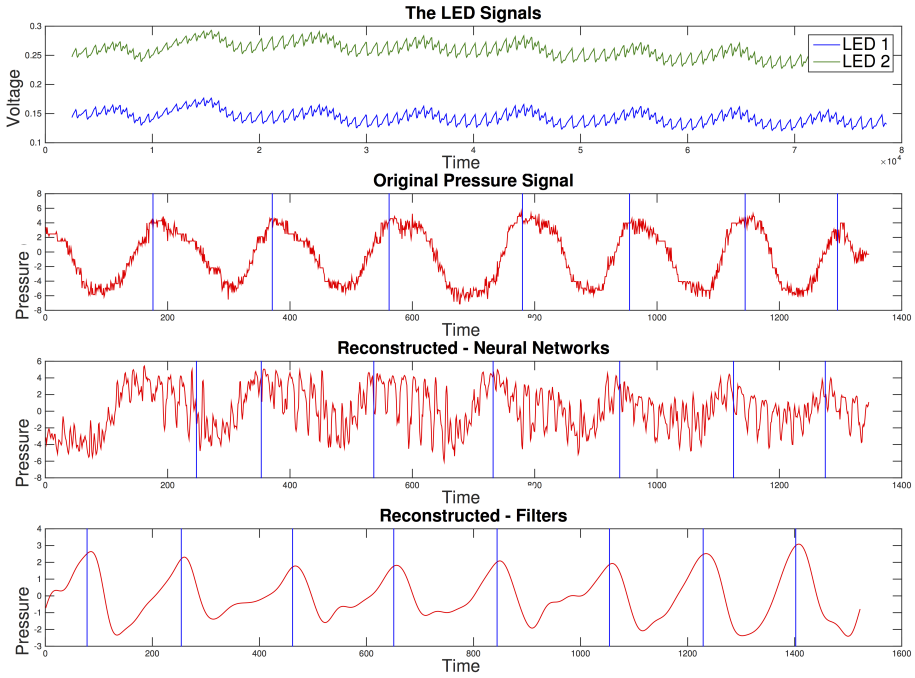

Figure A.6: Comparing the reconstructed breathing signals on human subject 6.