CNN-Based Speed Detection Algorithm For Walking and Running Using Wrist-Worn Wearable Sensors

Venkata Devesh Reddy Seethi
deveshreddy27@gmail.com

Follow this and additional works at: https://huskiecommons.lib.niu.edu/allgraduate-thesesdissertations

Part of the Computer Sciences Commons

Recommended Citation

This Dissertation/Thesis is brought to you for free and open access by the Graduate Research & Artistry at Huskie Commons. It has been accepted for inclusion in Graduate Research Theses & Dissertations by an authorized administrator of Huskie Commons. For more information, please contact jschumacher@niu.edu.
In recent years, there have been a surge in ubiquitous technologies such as smartwatches and fitness trackers that can track human physical activities effortlessly. These devices have enabled common citizens to track their physical fitness and encourage them to lead a healthy lifestyle. Among various exercises, walking and running are the most common activities people do in everyday life, either to commute, exercise, or do household chores. While performing these activities, the speed at which a person walks or runs is an essential factor to determine the intensity of activity. Therefore, it is important to measure walking/running speed to estimate the burned calories along with preventing the risk of soreness, injury, and burnout. Existing wearable technologies use a GPS sensor to measure speed, which is highly energy inefficient and does not work well indoors. To solve this problem, we design, implement, and evaluate a convolutional neural network-based algorithm that leverages data from accelerometer and gyroscope sensors in a wrist-worn device to detect speed with high precision. We have also evaluated various other machine learning algorithms to compare our results.
CNN-BASED SPEED DETECTION ALGORITHM FOR WALKING AND RUNNING USING WRIST-WORN WEARABLE SENSORS

BY

VENKATA DEVESH REDDY SEETHI
© 2021 Venkata Devesh Reddy Seethi

A THESIS SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE
MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

Thesis Director:
Pratool Bharti
ACKNOWLEDGEMENTS

I would like to give my deepest appreciation to my advisor, Dr. Pratool Bharti for his deep-rooted knowledge, articulate ideas, and brilliance in teaching that stirred my curiosity towards research and made working with him undoubtedly a great learning experience. I am forever grateful and extremely indebted to him for giving me the opportunity to be his research assistant and providing me with computing resources in his lab without which I could not have envisioned my thesis. His kind words, encouragement, sage advice, and piercing reviews on my work and thesis have made my journey most delightful and possible. I would also like to extend my deepest gratitude to Dr. Reva Freedman for her invaluable counsel on writing and her guidance throughout my thesis. I cannot express enough thanks to Dr. Hamed Alhoori, who inspired me throughout my master’s, and for his intuitive ideas that helped me shape my thesis. It was a great pleasure being a part of Data Lab and many thanks to my friends in Data Lab for all the discussions that created a stimulating environment and fun outings.

On the personal side, I am eternally thankful for my family who nurtured the importance of creativity and intellect. I would like to express my most effusive gratitude to my parents for their belief in me, bolstering my ideas and making me the person I am today. My deepest heartfelt thanks to my sister for her constant support, for always being there for me, and most importantly, I would like to thank her for being my guiding light since my childhood. Finally, special thanks to my brother-in-law, whose droll wit and vision have invigorated me thorough all my endeavors.
DEDICATION

To Mom, Dad, Sonu, Kris, and Loki
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF TABLES</th>
<th>vii</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
</tbody>
</table>

## Chapter

1 INTRODUCTION 

1.1 Background 

1.2 Objective 

1.3 Outline 

1.4 Literature Review 

1.4.1 Step Frequency 

1.4.1.1 Methods Based on Time Domain Features 

1.4.1.2 Methods Based on Frequency Domain Features 

1.4.2 Speed Detection 

1.4.2.1 Feature-Based Models 

1.4.2.2 Machine Learning 

1.4.2.3 Neural Networks 

2 DATA PROCESSING 

2.1 Hardware Description 

2.2 Data Collection Procedure 

2.2.1 Ground Truth Collection 

2.2.2 Data Cleansing
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3</td>
<td>Data Preparation</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>2.3.1 Sliding Window Method</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>2.3.2 Final Datasets</td>
<td>22</td>
</tr>
<tr>
<td>2.4</td>
<td>Evaluation</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>COMPONENTS OF SPEED DETECTION</td>
<td>26</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>26</td>
</tr>
<tr>
<td>3.2</td>
<td>Step Frequency</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>Step Frequency Algorithm</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>3.3.1 Procedure for Step Frequency</td>
<td>27</td>
</tr>
<tr>
<td>3.4</td>
<td>Step Length</td>
<td>29</td>
</tr>
<tr>
<td>3.5</td>
<td>Results and Discussion</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>SPEED ESTIMATION</td>
<td>31</td>
</tr>
<tr>
<td>4.1</td>
<td>Speed Estimation Using Machine Learning Algorithms</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>4.1.1 Regression</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>4.1.2 Hybrid Two-Stage Classification-Regression</td>
<td>33</td>
</tr>
<tr>
<td>4.2</td>
<td>Speed Estimation Using Neural Networks</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>4.2.1 Recurrent Neural Networks (RNNs)</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>4.2.2 Convolution Neural Networks (CNNs)</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>4.2.2.1 Training Procedure of CNN Model</td>
<td>37</td>
</tr>
<tr>
<td>4.3</td>
<td>Results</td>
<td>38</td>
</tr>
<tr>
<td>4.4</td>
<td>Making the Speed Detection Model Compatible with Edge Devices</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>4.4.1 Separable Convolution Layers</td>
<td>41</td>
</tr>
<tr>
<td></td>
<td>4.4.2 Pruning</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>4.4.3 Quantization</td>
<td>42</td>
</tr>
<tr>
<td>Chapter</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>5 CONCLUSION.</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>5.1 Discussion and Summary</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td>5.2 Limitations</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>5.3 Future Work</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>REFERENCES</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>2.1 Handcrafted time domain features in Dataset-2</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>2.2 Handcrafted frequency domain features in Dataset-2</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>2.3 Confusion matrix</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>4.1 Results for regression algorithms for speed detection</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>4.2 Results for classifier algorithms to classify between low and high speeds</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>4.3 Results for hybrid two-stage classification and regression algorithms for speed detection</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>4.4 Results of our speed detection algorithms</td>
<td>41</td>
<td></td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Wearable shipments reported according to International Data Corporation (IDC)</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Conceptual workflow</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Participant wearing a Shimmer device on the wrist while running on a treadmill (left); with a Shimmer device (right)</td>
<td>17</td>
</tr>
<tr>
<td>2.2 Accelerometer and gyroscope sensory data captured from a participant at different speeds on the treadmill</td>
<td>19</td>
</tr>
<tr>
<td>2.3 Sliding window method</td>
<td>21</td>
</tr>
<tr>
<td>3.1 Step frequencies for three participants while walking/running at different speeds</td>
<td>29</td>
</tr>
<tr>
<td>3.2 MAPE of step frequency (on y-axis) at different speeds</td>
<td>30</td>
</tr>
<tr>
<td>4.1 Effect of handcrafted features on random forest regression</td>
<td>32</td>
</tr>
<tr>
<td>4.2 Top 33 important handcrafted features for training random forest</td>
<td>33</td>
</tr>
<tr>
<td>4.3 Architecture of CNN-based speed detection algorithm</td>
<td>36</td>
</tr>
<tr>
<td>4.4 Results of CNN-based speed detection algorithm against train-test-evaluation split and leave-one-out cross-validation strategies</td>
<td>39</td>
</tr>
<tr>
<td>4.5 True vs. predicted speed for train-test-evaluation split (top) and leave-one-out cross-validation (bottom)</td>
<td>40</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

1.1 Background

The modern revolution of leading a healthy lifestyle is motivating people to incorporate physical exercises into their daily routine. Among various exercise options, walking and running are the most commonly practiced for fitness, health, and leisure [1]. These cardiac activities are also popular among the elderly population. However, regardless of a person’s age, losing weight through exercise is directly associated with the intensity at which they are performed. For example, a person on average burns 1.5 times more calories running at 5 mph vs. 3 mph [2]. Similarly, walking at 2.5 mph burns twice as many calories compared to 1.25 mph [3]. Therefore, it makes fine-grained walking and running speed detection with step frequency very important for people who want to keep track of the calories in any exercise session. Additionally, tracking the speed accurately can help people to understand their fitness levels and encourage them to design custom exercise regimes and control their food intake accordingly.

In clinical settings, monitoring walking can often provide valuable insights on a person’s health condition. For example, changes in walking speed can be an indication of early symptom’s of dementia [4] and even depression [5].

Nevertheless, these facts and observations create an absolute demand for a device that can measure walking and running speed precisely in a swift manner. For maximum outreach, the device should be accurate, pervasive, wearable, and culturally acceptable among all age
groups. Although a myriad of devices [6] that aim for precise speed detection are commercially available, their practical usage is limited. The majority of these devices [7] are based on the Global Positioning System (GPS), which relies on radio-wave communication through satellites and don’t work well indoors due to signal interference. This can be problematic for people who live in colder places where walking outdoors is difficult. Alternatively, Indoor Positioning System (IPS) identifies a person’s position using Wifi strength [8], radio frequency identification (RFID) tags [9] or infrared beacon [10] technologies present in the infrastructure of a building. These solutions suffer from noise when large crowds are present in the building and are constrained for usage in buildings that meet the infrastructure needs. On the other hand, vision-based solutions are not pervasive [11], that do not work as stand-alone devices, and directly invade user privacy. Other common problems with existing solutions are either in accuracy of the devices or their energy efficiency.

In this realm, a solution based on a wrist-worn wearable can be promising considering its unobtrusive nature, pervasiveness, and convenient wearability. These are commercially available off the shelf at a modest price and have recently gained a lot of attention from the health and fitness communities. According to a recent study [12], International Data Corporation (IDC) reported an estimate of 69.3 million smartwatches shipped in 2019, which increased to 91.4 million by 2020, and they have further projected 156 million units to reach worldwide markets by 2024, as shown in Figure 1.1. Other wearables such as smart glasses, and smart belts are not as ubiquitous as wrist-worn wearables. There are many earwear devices in the market, but they lack activity tracking capabilities due to lack of motion tracking sensors. Instead they are used for natural language translation. These statistics and facts show that wrist-based solutions are the future of fitness and healthcare. A study in 2019 by Santos-Gago et al. [13] addressed the prevalence of wrist-worn wearables in the fitness industry, utilized by both amateurs and professionals. Further, the study underpinned the
necessity of developing precise health and fitness applications by leveraging sensors embedded in the wearables.

Figure 1.1: Wearable shipments reported according to International Data Corporation (IDC)

Modern wrist-worn wearables such as smartwatches and wristbands are equipped with inertial measurement unit (IMU) sensors such as an accelerometer, a gyroscope, or a magnetometer that measures linear accelerations, angular velocities, and magnetic strengths respectively in the x, y, and z directions of the Cartesian coordinate system. These inertial sensors are cheap, energy efficient, and miniature enough to fit in a smartwatch/wristband to capture the wrist movement in real-time with minimal sensor-based error. In comparison to GPS-based solutions, IMU sensors work well outdoors as well as indoors. In this thesis,
we employed tri-axial sensors, namely, accelerometer and gyroscope sensors, in a wrist-worn device to detect walking and running speed swiftly and precisely.

1.2 Objective

The objective of my thesis is to design an algorithm to determine walking and running speed along with step frequency by leveraging sensory data from wrist-worn devices.

1.3 Outline

This thesis is divided into five chapters. The aim of Chapter 1 is to visit and discuss the literature and methodology relating to the thesis. Then Chapter 2 introduces the wearable device used in our study and explains our data collection protocol and the datasets used in thesis. The datasets discussed in Chapter 2 are used in Chapters 3 and 4. To determine walking/running speed, we have taken two routes, which are divided into Chapters 3 and 4. Chapter 3 shows a traditional approach by finding individual components of speed and presents a step frequency algorithm. The traditional approach of Chapter 3 did not work for us, so we adopted a direct approach, discussed in Chapter 4. Finally, Chapter 5 concludes the thesis with a summary of the study and provides insight on future directions for the research.

The conceptual workflow for this study is briefly shown in the Figure 1.2. We collected our data from 15 participants while they were walking or running on a treadmill. The data was captured through an Android application using a mobile device. This data was then sent to a computer for further analysis and modelling the algorithm.
1.4 Literature Review

Inertial sensors, namely the accelerometer and gyroscope, can be leveraged for a diverse spectrum of tasks, ranging from industrial fault detection to human motion monitoring. In industrial plants, inertial sensors are used to ease the protracted process of pinpointing the exact location of faults. For example, inertial sensors such as accelerometers can be embedded in complex mechanical systems to perform vibration analysis that automates detection of faults such as liquid flow error and mechanical issues related to control valve aging. Venkata and Rao [14] used a uni-axial accelerometer to identify the faults at a control valve with an accuracy of 97% by using support vector machines (SVMs). Navigation systems in micro-aerial vehicles (MAVs) rely on inertial sensors or sensor fusion-based inertial-visual systems to navigate in the absence of GPS signals [15]. Scientists at the National Aeronautics and Space Administration (NASA) designed Astrobee [16], a free-flying robot to facilitate basic utility
functions for the astronauts in the International Space Station. The position estimation of Astrobee is carried out by sensor fusion of accelerometer and gyroscope data through an augmented-state extended Kalman filter.

Inertial sensors are also used for human activity recognition (HAR). Researchers have extensively explored several methods for HAR-related applications by leveraging the sensors based on smartphones, body-worn wearables, and wrist-worn wearables. Bharti et al. [17] have classified 21 complex in-home activities leveraging multi-modal sensors available in smartphones. Human activity classification in the medical field can be used to monitor Parkinson’s patients by detecting gait freezes using a waist-worn [18] or wrist-worn IMU sensor [19]. In clinical studies, Bharti et al. [20] used wrist-worn wearables to detect self-harming activities attempted by patients in psychiatric facilities.

In physical fitness and sports, sensing technologies can be used to gain insight on performance and the amount of work done. To detect intensities, photoplethysmography-based (PPG) heart rate monitors are a popular choice. However, PPG has high energy demands that limit the long-term usage of wearables that have a limited battery span. McConville et al. [21] developed an energy-efficient, online model for detecting heart rate using an accelerometer that uses active learning to provide insights on patient recovery after a heart intervention. Dissanayake et al. [22] proposed a solution based on energy-efficient and cheap inertial sensors that used 3.75 times less energy than PPG sensors. To capture the fitness-specific motion artifacts, several algorithms were proposed for counting repetitive exercises [23], assessing volleyball skills [24], and analyzing stroke mechanics in swimming [25].

The use of inertial sensors for walking detection can be reduced to estimating three fundamental parameters of walking: step counts [26], step length [27,28], and walking speed [6,11,29–32]. In previous research relating to speed detection, sensors were mounted either on the shank [6], the thigh [33], or the wrist [29,32]. However, in performing walking-related activities or daily house chores, a sensor placed on the wrist was found to be advantageous
due to its high precision and recall rates [34]. The precision along with the prevalence of wrist-worn devices discussed in Section 1.1 dissuades the use of body-worn devices over wrist-worn devices and hence we used inertial sensors in a wrist-worn device in this research.

1.4.1 Step Frequency

In the domain of step detection, several pedometer applications were proposed using accelerometer and gyroscope sensors. The most common device to detect walking steps is smartphone, but these devices are not precise, as the point of placement of the device is inconsistent. For example, a smartphone can be carried in the hand, a pocket, or a backpack, but sensors placed on the shank, wrist, or leg are more stable, as the placement of the device is consistent. For all of these wearable devices and smartphones, there is one commonality, which is the use of either an accelerometer or a gyroscope to estimate the number of steps taken by a person. The research in this area has mainly taken two routes. The first route is to use time domain features and the second uses frequency domain features, which can be explained on the basis feature types. These two routes to find step frequency or step counts are elaborated in the following sections.

1.4.1.1 Methods Based on Time Domain Features

Time domain features are derived from time series data obtained in the form of a raw acceleration signal. These signals are first subjected to de-noising to remove frequencies irrelevant to human movement, followed by an optional transformation phase that involves transformations such as calculating the resultant acceleration or the energy of the signal. The final stage is composed of an algorithmic stage by which the existing studies can be
categorized into thresholding, peak detection, zero crossing, auto-correlation methods, and machine learning methods.

The thresholding method is one of the earlier methods that has received a lot of attention. Thresholding in time series data is applied using a certain cutoff threshold for the peaks in a signal, such that whenever the signal crosses a certain threshold it is interpreted as a step count. Herrera et al. [35] experimented with the resultant acceleration using the energy of the signal sampled at 100 Hz and 40 Hz to calculate the total number of steps taken by a person. In the results presented, the thresholding method could not be generalized well for all participants due to different styles of walking. In addition, the algorithm cannot be applied to lower speeds as the signal tends to become more noisy for lower speeds. Thresholding methods also fail where new patterns of hand movements are observed within the same person or different people.

In conjunction to setting a threshold on the acceleration signal peaks, Jayalath and Abhayasinghe. [36] identified the zero cross-overs and applied the threshold on the peaks at every cross-over. The authors used a gyroscope and a bandpass filter to capture frequencies in the $0.9 - 3$ Hz range. However, zero cross-overs are hard to identify due to the presence of noise in the signal. This problem was addressed in a more complex approach, where Ma et al. [37] fused the sensor data from accelerometer and gyroscope with an additional pressure sensor placed on the ankle, to detect the zero velocities or zero cross overs more precisely between each toe off and heel strike. But these methods require an additional sensor to be mounted on the ankle, which is not suitable for everyday usage.

In contrast to thresholding methods, auto-correlation methods find the correlation between the current sample of a signal and the preceding one by comparing standard deviations. Santos-Gago et al. [38] used auto-correlation method to demonstrate the cyclic nature of accelerometer signal during walking. In temporal domain, auto-correlation can measure the
degree of similarity between the given time interval and the previous version in all successive intervals. This iterative correlation function can then be used to count steps.

### 1.4.1.2 Methods Based on Frequency Domain Features

During walking, periodic movements occur at the legs as well as at the hands through to and fro motions. At the end of each to or fro motion, a peak in acceleration is observed. The continuous movement of a hand will then create multiple overlapped sine waves, with each sine wave having a different amplitude and frequency. Among these multiple sine waves, some low-frequency components are created due to other body movements and some high-frequency components are created due to environmental noise. Hence, in frequency analysis, the goal is to decompose the time-acceleration signal to frequency-amplitude signal and identify the frequency that is indicative of steps taken. Previous research that uses frequency analysis is shown below.

Continuous or discrete wavelet transformations (CWT/DWT) transform the mother signal into a set of wavelets. The wavelets are formed by applying convolutions in the form of translations and dilations of the mother function [39]. Wang et al. [40] have built a model to classify walking-related activities such as walking, upstairs/downstairs walking, and jogging. The first stage is a non-overlapping windowing technique to subsample the accelerometer signal into smaller durations. In the second stage, features are generated by applying DWT on the input signal to decompose the signal into wavelet coefficients having eight levels. Then, statistical features such as power, energy, mean, variance, and energy of neighbor difference are computed from the DWT coefficients. The final stage is a classifier, where the authors have achieved an accuracy of 93.32\% with logistic regression.
Fourier transforms play an important role in frequency domain analysis, as they can decompose a signal into a collection of signals with unique frequency and signal power. The Fourier transforms when applied to shorter intervals are referred to as short-time Fourier transforms (STFT). During calculation of FFT or STFT, the major challenge is the tradeoff between temporal and frequency resolution. A study in walking step detection [41] posits that there is a loss in spectral resolution while using STFT, and FFT can only be used on long data segments. Qi and Huang [42] show that FFT outperformed STFT in the detection of walking activities. In our case, since all participants walked/ran on the treadmill, they were subject to a degree of automacity where they tended to have similar step frequencies and stride lengths when they walked/ran at constant speed. Therefore, since the step counts were similar during a given session of walk/run, we could use FFT on larger segments without losing temporal resolution.

In the approach shown by McCamley et al. [43], the acceleration is first integrated and then differentiated twice using the Gaussian CWT with a Gaussian mother wavelet of order one (corresponding to the first derivative of the Gaussian wavelet). The locations of the minima of the smoothed acceleration signal obtained after the first differentiation identify the initial contact (IC), while the locations of the maxima of the smoothed jerk signal obtained after the second differentiation provide the final contact of the foot (FC). This method showed a relative independence from sensor alignment during the acquisition. Feature-based machine learning models for clustering (such as k-means and hidden Markov models) and classifiers (such as random forests and decision trees) have performed well for classifying different activities for walk detection but they fail in case of counting steps.
1.4.2 Speed Detection

There are several methods ranging from feature-based to deep learning models from which speed can be inferred using either or both the accelerometer and the gyroscope. These methods are briefly summarized in the following section.

1.4.2.1 Feature-Based Models

For feature-based models, the raw data is statistically transformed to construct a definite linear or non-linear relation with running speed using time or frequency domain features. Bertschi et al. [32] used a wrist-worn device with handcrafted features such as hand movement frequency, height, and weight of a person to estimate the distance covered in a run. Fasel et al. [30] adopted a personalized approach by using four handcrafted features with k-means clustering followed by a linear regression model to estimate instantaneous speed with different surfaces, slopes, constraints, and environments. A recent study by Soltani et al. [31] demonstrated a least squares approach with six handcrafted features to make a personalized model. However, they state that their algorithm would not work well for different populations due to the variance in walking styles and thus they use the personalized approach.

In lieu of feature-based models, integration methods can be used to find quantitative intensity descriptors such as distance travelled [44] and walking speed [32, 44]. In the integration method, walking speed estimation requires a one-time integration of the acceleration signal, whereas walking distance estimation requires a double integration. Sabatini et al. [44] built a heuristic-based algorithm with an ankle-worn device based on a bio-mechanical assumption of leg movement. On the other hand, Bertschi et al. [32] built a custom wrist-worn
device that converts the accelerations of the accelerometer into speed. Although integration method does not require training data and is simple to implement, it is prone to errors in practical usage. A study by Diez et al. [45] posits that for any wearable device, a non-negligible bias and noise of the micro-electromechanical system (MEMS) in accelerometers and gyroscopes make the error in walking distance grow cubically boundless. For example, to find the speed of walking $v(t)$, we integrate the observed accelerometer signal, i.e., $a(t)$. During the integration procedure as shown in Equation 1.1, there is a bias or error $e(t)$ that is induced due to sensor limitations. Therefore, on integration of the accelerations, it will introduce a bias $E$, as shown in Equation 1.2, that reduces the accuracy of the model. The sensor error and bias can be eliminated with the expense of using higher quality sensors, according to Yang and Yi [46]. But commercial devices use different sensors and it is expensive to build a sensor with less measurement error. In addition, accelerations from the accelerometer have to be mapped onto the navigation frame from the direction of movement [45], which can be a challenging task.

$$\int_0^t a(t) dt = v(t) \quad (1.1)$$

$$\int_0^t (a(t) + e(t)) dt = v(t) + E \quad (1.2)$$

Bishop and Li [6] mounted a pair of IMU sensors (each having a bi-axial accelerometer and uni-axial gyroscope) to the lateral side of the mid-shank and estimated the walking speed while walking on treadmill. The authors took an inverted pendulum approach to estimate walking speed from the translatory movement of the shank. In the inverted pendulum approach, the center of mass (CoM) of the body is assumed to move in a cyclic path while the foot point acts like a pivot of the pendulum. Therefore, the authors placed the sensors parallel to the sagittal plane, which is in the same direction as the CoM. In this study, all treadmill
walk sessions were segmented into shorter stride cycles, where the start of each stride cycle was recognized when the shank was situated parallel to the direction of gravity. For each stride cycle, the differential tangential accelerations from both of the accelerometers was recorded. Then the shank’s angular velocity was computed by integrating the accelerations until the next start point occurs. At the onset of a new stride cycle in the stance phase, the velocities are made zero. With this procedure, the authors achieved a percentage RMSE of 8% for the walking speed range of 0.8 – 1.8 mph.

1.4.2.2 Machine Learning

According to a review on gait estimation methods [45], bio-mechanical models are accurate as they are calibrated based on the kinetic features of walking. However, these methods are susceptible to wideband measurement noise and low-frequency drifts affecting the inertial sensors [47]. Mannini and Sabatini [47] evaluated machine learning (ML) regression models against hidden markov model-based (HMM) strap-down integration methods (SDI) and threshold-based SDI and found similar root mean square estimation error of 2.0% and 4.2%, with and without personalization for the ML model, against 2.0% and 3.1% using HMM-based SDI and threshold-based SDI. In a comparison between the two, the ML model showed less intra-subject variability than the SDI method. Although the ML model was not able to generalize inter-subject variability, the authors discuss that subject-specific calibration would immensely benefit the model.

Another approach to detect speed is the use of hybrid models where the speeds are first classified into low and high speeds, followed by training regression algorithm for each class. In these methods, the speeds are categorized based on movement styles related to walking, jogging, and running. Then activity specific regression models are trained for each walking
style. Mannini and Soltani et al. [33] built a classification-regression-based model with three stages. In the first stage, an SVM classifier is programmed to identify the episodes of a person’s movement. The second stage is composed of a cascaded SVM and logistic regression classifier where SVM assigns a feature vector to the walk or run activities and the logistic regression estimates the posteriori probability. Finally, the speed is estimated by implementing a Bayesian soft assignment rule on the input feature vector. This method achieved a root mean square error of 0.3 – 0.7 kmph when the model was trained and interpolated at different speeds between the speed range 1.2 – 9.8 kmph.

Zihajehzadeh et al. [29] proposed a two-stage hybrid model using an SVM and Gaussian random process (GPR) algorithms for classification and regression respectively. In this, the first stage is a SVM classifier that classifies walking into slow (≤ 1m/s) and medium-fast walking (≥ 1m/s). After classification, two separate walking speed regression models using Gaussian process regression (GPR) are trained to estimate the walking speed. For slower walking speeds, the authors achieved an MAPE of 13.5% with classifier and 51.9% without classifier. Similarly, for medium-high walking speeds, MAPE with and without classifier were 12.5% and 17.3%. In both conditions, improved accuracy for the model with classifier shows the benefit of incorporating hybrid models into speed detection.

1.4.2.3 Neural Networks

Neural learning is a novel avenue of ever-growing research. Recently, illuminating studies were published by researchers that use the power of deep learning for human activity recognition [17,48–50], gait authentication [51], and even estimation of knee joint forces in sports [52]. In the context of walking/running speed detection, the first study that used
artificial neural networks (ANNs) for speed detection was proposed by Song et al. [53] in 2007.

In 2016, Hannink et al. [54] used convolutional neural networks to estimate eight gait characteristics subdivided into step-length-based characteristics (stride length, width, and medio-lateral change in foot angle) and step time characteristics (stride, swing and stance, and heel and toe contact times). This CNN method outperformed a double integration method proposed by Rampp et al. [55]. Subsequently, a second model was proposed by Hannink et al. [27] that took a similar approach to [54] but with a smaller model size to estimate the step length. In 2019, a one-dimensional CNN architecture was proposed by Wang and Xu [56] using accelerometer data to estimate walking speed, which achieved MAPE of 7% and 18% on running and walking speeds respectively.

From this perspective, we demonstrate a precise running speed detection algorithm by leveraging deep convolutional neural networks and a wrist-worn sensor. As discussed above, CNN architectures were used for stride length estimation [27] and speed detection [56], where both studies proposed a similar architecture by using input data only from an accelerometer. The Wang and Xu [56] method used a one-dimensional CNN architecture with a kernel length of five. Since the scope of our current work is to determine the speed, we have only compared our results against [56].
CHAPTER 2
DATA PROCESSING

2.1 Hardware Description

In this study, we have employed the Shimmer [57] IMU sensor as a wearable device for collecting the accelerometer and gyroscope data from the participants. Due to its small form factor, it can be easily worn on the wrist like a smartwatch (as shown in Figure 2.1). The Shimmer is widely used in the research community to study human body movements [49,58]. This device is based on the TinyOS firmware, with the core element a low-power MSP430 CPU with 24 MHz clock rate that controls the device operation. The CPU has an integrated 16-channel 12-bit analog-to-digital converter that is used to constantly sample and capture tri-axial acceleration and rotational signals from the inbuilt accelerometer and gyroscope sensors respectively [57]. The accelerometer has two settings, low noise, and wide range with ranges of ±2g and ±16g (where g is the gravitational acceleration) and the gyroscope has a range of ±2000 dps (degrees per second). In this study, low noise accelerations from the accelerometer in the range of ±2g and angular velocities from the gyroscope in the range ±1000 dps were collected, where both of these sensors were sampled at 51.1 Hz. During the experiment, sensory data from the device were streamed to an Android phone (OnePlus 5T) via an inbuilt radio module using the Shimmer Connect Android application. Data from the phone was later transferred to a computer for post-processing and modeling.
2.2 Data Collection Procedure

We collected the sensory data from 15 healthy adults (8 males and 7 females) in this study. These participants belonged to the age group of 22±2 years and had an average height of 168.5 ± 13.5 centimeters. The experimental protocol was approved by the Institutional Review Board (IRB) office at Northern Illinois University. All participants signed the IRB consent forms and indicated that they were not fatigued and did not have any physical injuries or disabilities at the time of the experiment.

Before the experimental procedure, the Shimmer was mounted on the participant’s preferred wrist. Although, we intended to collect the data in the range of 3 – 7 mph, participants were allowed to set their own limit within our range. Each participant performed walking/running on the treadmill for a 45-second session at a fixed speed. Speed was gradually increased by 0.5 or 0.7 mph in every new session, based on the participant’s preference. The participants recruited in our study had different fitness levels. On an average, participants were able to walk/run for 15 ± 5 minutes and during this time completed 9 – 16 sessions at
different speeds. In total, 205 sessions were recorded from 15 participants, which comprised 9,225 seconds and 313,650 samples of accelerometer and gyroscope sensor data.

### 2.2.1 Ground Truth Collection

At the start of each session, we captured the data using the Shimmer Connect Android application on an Android phone (OnePlus 5T). The data from the Shimmer device was streamed onto the Android phone through a Bluetooth connection, where a new CSV (comma-separated value) file was created for each 45-second session, tagged with a unique participant ID and their running speed. Sample raw data from a participant at different speeds are shown in Figure 2.2. For step counts, we used a stopwatch to count the steps taken during a 3-second interval and repeated this for three times during the 45-second session. To reduce the human error in counting steps, we averaged the step counts calculated in the three 3-second intervals, mean of step counts in three intervals was calculated and stored with participant ID in a data sheet.

### 2.2.2 Data Cleansing

Data cleansing is an important aspect of data processing to remove the noise before further analysis. During the experiment, some participants were unable to run for 45 seconds and, in such event, the readings were discarded to maintain consistency in the size of data samples at each speed. Additionally, to discard the noise associated with the start and stop of each experiment session, the recorded data of 45 seconds was trimmed by discarding 2.5 seconds of data from the beginning and the end of recording, making each session effectively a length of 40 seconds.
Figure 2.2: Accelerometer and gyroscope sensory data captured from a participant at different speeds on the treadmill

Like other sensors, IMU suffer from bias, alignment, and sensitivity errors that affect the accuracy of the signals they produce. The Shimmer provides calibration of sensory data to minimize these errors. We leveraged the calibrated value of the tri-axial accelerometer and gyroscope data to train our algorithms to detect the speed as well as to detect step frequencies.
2.3 Data Preparation

In this section, we detail the data splitting procedure followed by the preparation of two datasets. Firstly, we split the data into smaller portions using sliding window procedure, followed by final dataset preparation into two categories.

2.3.1 Sliding Window Method

After pre-processing the signals, we prepared 205 sessions (each 40 seconds long) of data at different walking or running speeds. Both tri-axial accelerometer and gyroscope sensors were sampled at 51.1 Hz, which makes the total dataset dimension [418200, 6]. Next, we segmented the data into 3-second moving windows with 50% overlap. That converts the dataset to a dimension of [5465, 153, 6], where total input samples are 5,465 and each input dimension is [153, 6]. This data is given as input to the sliding window method shown in Algorithm 1.

In human activity recognition, previous studies [17, 20] have tested different ranges of $Ov\%$ and $F_{size}$ and chose the one that works best with their algorithm. We took a similar approach in our study by permuting $Ov\%$ between 0% to 75% and $F_{size}$ between 0.5 to 3 seconds (25 to 153 samples). The optimum values for two variables, $Ov\%$ and $F_{size}$ were found to be 50% and 3 seconds respectively (as shown in Figure 2.3). Our sliding window implementation is described in Algorithm 1 and the heuristics of choosing the sliding window sizes are given below:

- $Ov\%$ should be large enough to enable instantaneous tracking of speed per second.
- $F_{size}$ should be wide enough to capture at least one step per window.
After choosing the appropriate $F_{size}$ and $Ov\%$, the sample size of each session is initialized with the variable $total_{sample-size}$ in step 1. As each session was 40 seconds long, $total_{sample-size}$ was initialized with 40. For the first sliding window, the starting index ($I_s$) was initialized with 0 and ending index ($I_e$) was initialized with $F_{size}$ in step 2. In step 3, the total number of sliding frames for each session is calculated as $n$. After initializing the above variables, we iterate until we obtain $n$ sliding frames for each session. In this loop, the data is first sliced in step 5 with $I_s$ and $I_e$. The data slice is then appended and stored in an array as $Segmented_{sliding\_windows}$. After each segment is sliced and stored, the start of the next window, i.e., $I_s$, is assigned as the value of previous $I_e$ adjusted with $ov\%$ in step 7. Subsequently, the next $I_e$ is updated as the sum of $I_s$ and $F_{size}$ in step 8. Finally, the frame count is incremented in step 9 and after the end of the loop, the stored sliding window data is returned as $Segmented_{sliding\_windows}$ in step 11. A visual representation of the sliding window procedure is presented in Figure 2.3.

Figure 2.3: Sliding window method.
Algorithm 1: Algorithm for sliding windows

Input: Dataset data, Frame size $F_{size}$, Overlap ov%.

Output: Segmented data Segmented_sliding_windows.

1: $total_{sample-size} \leftarrow total_{number of samples}$
2: $Frame \leftarrow 0, I_s \leftarrow 0, I_e \leftarrow F_{size}$
3: $n \leftarrow \frac{total_{sample-size} - F_{size}}{F_{size} \times (1 - ov\%)} + 1$
4: while $Frame \neq n$ do
5: $data_{slice} = data[I_s : I_e]$
6: $Segmented_{sliding}_{windows}.append(data_{slice})$
7: $I_s = I_e - round(F_{size} * ov\%)$
8: $I_e = I_s + F_{size}$
9: $Frame = Frame + 1$
10: end while
11: return $Segmented_{sliding}_{windows}$

2.3.2 Final Datasets

After obtaining the segmented sliding windows data from Algorithm 1, we subdivide this data further into two datasets. In the first dataset, Dataset-1, we use the raw data after the sliding window procedure. For Dataset-2, we took Dataset-1 and computed time-domain-and frequency-domain-based handcrafted features for each sample.

Handcrafted features in time and frequency domains have shown to be effective in prior studies in human activity recognition [17,20,59]. From these studies, we have leveraged time and frequency domain features listed in Tables 2.1 and 2.2. Our analysis had 16 handcrafted features that we calculated on the x, y, z axes of accelerometer and gyroscope as well as
their resultant magnitudes. Therefore, by calculating 16 handcrafted features on eight axes we get \(16 \times 8 = 128\) features as input.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum, Maximum, Mean</td>
<td>(\text{Min}(X), \text{Max}(X), \text{Mean}(X))</td>
</tr>
<tr>
<td>Standard deviation</td>
<td>(\sigma = \sqrt{\frac{\sum (X_i - \mu)^2}{N}})</td>
</tr>
<tr>
<td>Variance</td>
<td>(\sigma^2)</td>
</tr>
<tr>
<td>25(^{th}) percentile</td>
<td>(Q1 = \text{Median of lower half of data})</td>
</tr>
<tr>
<td>75(^{th}) percentile</td>
<td>(Q3 = \text{Median of lower upper of data})</td>
</tr>
<tr>
<td>Inter quartile range</td>
<td>(IQR = Q3 - Q1)</td>
</tr>
</tbody>
</table>

Table 2.2: Handcrafted frequency domain features in Dataset-2

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skewness</td>
<td>(\frac{\sum (X_i - \mu)^3}{N} \times \sigma^3)</td>
</tr>
<tr>
<td>Kurtosis</td>
<td>(\frac{\sum (X_i - \mu)^4}{N} \times \sigma^4)</td>
</tr>
<tr>
<td>Shannon spectral entropy</td>
<td>Measure of spectral power distribution</td>
</tr>
<tr>
<td>Fast Fourier Transforms</td>
<td>Top two dominant peaks ((P1, P2)) where (P1, P2 = (F1, A1), (F2, A2)) (Frequency (F) and amplitude (A))</td>
</tr>
</tbody>
</table>

2.4 Evaluation

We evaluated our speed detection model with three evaluation metrics: mean absolute error (MAE), mean absolute percentage error (MAPE), and R-squared \((R^2)\) as described in Equations 2.1, 2.2, and 2.3, respectively, and evaluated step counts with MAPE. MAE measures the average magnitude of the error in the prediction without considering its direction. It uses the same scale as the data and has no bounded range, hence it cannot be interpreted without knowing the scale of real values. In contrast, MAPE measures the percentage error in the prediction based on the predicted and real values. Therefore, MAPE can be interpreted well because its range is bounded between 0 and 100. \(R^2\) (coefficient of
determination) is based on the ratio of the prediction error and the variance in the predicted values. It typically ranges between 0 and 1, where \( R^2 \leq 0 \) indicates that the model explains none of the variability of the data around its mean and 1 indicates it explains all the variability. If \( S_i, \hat{S}_i \) and \( \bar{S}_i \) are predicted, real, and mean predicted speeds respectively, then the equations can be defined as Equations 2.1, 2.2, and 2.3.

\[
MAE = \frac{1}{n} \sum_{i=1}^{N} |S_i - \hat{S}_i| 
\]  

(2.1)

\[
MAPE = \frac{1}{n} \sum_{i=1}^{N} \frac{|S_i - \hat{S}_i|}{S_i} 
\]  

(2.2)

\[
R^2 = 1 - \frac{\sum_{i=1}^{N} (S_i - \hat{S}_i)^2}{\sum_{i=1}^{N} (S_i - \bar{S}_i)^2} 
\]  

(2.3)

The classifier algorithms in Section 4.1.2 were evaluated based on their precision, recall, F-1 score, and accuracy. These metrics can be explained by referring to the confusion matrix in Table 2.3.

**Table 2.3: Confusion matrix**

<table>
<thead>
<tr>
<th>Predicted values</th>
<th>True values</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>True Positive (TP)</td>
<td>False Negative (FN)</td>
<td></td>
</tr>
<tr>
<td>Negative</td>
<td>False Positive (FP)</td>
<td>True Negative (TN)</td>
<td></td>
</tr>
</tbody>
</table>

In the above confusion matrix, the distribution of true values and predicted values are shown in the columns and rows respectively. By using values from the confusion matrix, we can derive the metrics for our evaluation: accuracy, precision, recall, and F1 score as shown in the Equations 2.4, 2.5, 2.6, and 2.7. Among these metrics, accuracy measures the total number of instances that were predicted correctly. Precision tells us the degree of
correctness when a certain class is predicted. On the other, hand recall measures the degree to which a certain class is identified correctly. Finally, the F1 score is an aggregate measure that takes the harmonic mean of precision and recall and presents the general performance of the algorithm. In our analysis, we show the results for micro-averages, where \( TP, TN, FP, \) and \( FN \) are calculated globally, i.e., for both the walking and running speed classes put together.

\[
\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.4}
\]

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{2.5}
\]

\[
\text{Recall} = \frac{TP}{TP + FN} \tag{2.6}
\]

\[
F1 - \text{Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.7}
\]
CHAPTER 3
COMPONENTS OF SPEED DETECTION

3.1 Introduction

Running or walking is a repetitive bipedal motion, where each strike of feet to the ground propels the person forward with a certain displacement. These displacements (step length, \( l \)) along with the rate of strike of feet (step frequency, \( f \)) are two critical components for determining the speed of walking and running. Speed \( S \) can be calculated using Equation 3.1. Therefore, \( S \) during walking/running changes when a variation in either or both of \( l \) and \( f \) is observed.

\[
S = l \times f
\]  

(3.1)

3.2 Step Frequency

Step frequency is one of the key indicators in determining the amount of work done in a day for people who are trying to lose weight. In clinical settings, it may be used for identifying underlying diseases or to track the recovery of an elderly person. Therefore, for walking and running, it is important to track step counts or step frequency along with speed for people trying to be fit as well as elderly people who are recovering from illness.
3.3 Step Frequency Algorithm

In this section, we outline our step frequency algorithm, which uses only data from the accelerometer. The core implementation of the step frequency algorithm is based on fast Fourier transforms (FFT), where FFT is used to transform the signal into the frequency domain and capture the dominant frequency. The algorithmic implementation is detailed in the next section.

Algorithm 2: Algorithm for computing step frequency

**Input**: Tri-axial accelerometer data \((A_x, A_y, A_z)\)

**Output**: Step frequency \(S_f\)

1: \(A_{xyz} = \sqrt{A_x^2 + A_y^2 + A_z^2}\)

2: Adjusted acceleration \(A_r = A_{xyz} - \text{mean}(A_{xyz})\)

3: \(F_A = \text{FFT}(A_r)\)

4: \(F_P = \text{Process} \ F_A \text{ through 2\textsuperscript{nd}-order bandpass Butterworth filter with range (0.5Hz - 4.5Hz)}\)

5: Peaks = Find peaks in \(F_A\)

6: One Step Frequency \(S_f = \text{Peak with the maximum absolute amplitude}\)

3.3.1 Procedure for Step Frequency

Our input to the step detection algorithm (shown in Algorithm 2) is the tri-axial acceleration data captured from the accelerometer in the Shimmer device. In step 1, the resultant magnitude of the tri-axial acceleration is computed as \(A_{xyz}\). In step 2 we eliminate zero frequencies from \(A_{xyz}\) by subtracting the mean, giving \(A_r\). \(A_r\) is now the input for the fast
Fourier transforms (FFT) in step 3, which converts the signal into frequency domain. Equation to compute FFT is presented in Equation 3.2. The frequency domain features can be obtained by decomposing the acceleration values into a discrete Fourier transform matrix, where the FFT uses a divide and rule algorithm to reduce the matrix into sparse factors. Due to this procedure, the time complexity is reduced from order $O(n^2)$ to $O(n \log n)$. FFT then produces a set of real and imaginary values representing amplitude and frequency respectively. The FFT iterates for a total of $N$ steps, where $N$ is the total number of data samples. In this case, with a 51.1 Hz sampling rate and 40 seconds of data, $N = 51.1 \times 40 = 2044$.

$$FFT(A_r) = \sum_{n=0}^{N-1} A_r[n] e^{-j2\pi kn/N} \quad (3.2)$$

In step 4, the signal is sent through a 2$^{nd}$-order Butterworth bandpass filter by limiting frequencies to the range of 0.5 - 4.5 Hz. The main purpose of bandpass filtering is to preserve frequencies that are deterministic of human movement. In this regard, we have set the low and high frequency limits as 0.5 and 4.5 Hz. We chose the low frequency to capture one step taken in two seconds, which is common in very slow walking conditions, whereas many athletes have their higher cadence at 3 Hz, so we set the high frequency limit at 4.5 Hz. Next, in step 5, we find peaks in the frequency domain as $Peaks_A$. In step 6, we then pick the dominant peak from $Peaks_A$ as the step frequency $S_f$. After knowing the step frequency, we can gauge the step counts $S_c$ as $S_c = S_f \times t$. The ground truth for this study was achieved by manual step counting during the procedure explained in Section 2.2.1, which is referred to as the observed step frequency. Figure 3.1 shows the step frequency for all speeds recorded among three participants of different heights.
3.4 Step Length

Step length estimation is a challenging task, as it is hard to collect the ground truth while the participants are walking or running on the treadmill. In previous studies, Fasel et al. [30] and Soltani et al. [31] used a GPS to obtain the ground truth for stride length. This is not realistic in regular usage to carry an additional device with their smartwatch to calibrate step lengths. As an alternative to obtaining the ground truth, we found the step lengths from Equation 3.1 by taking treadmill speed and step counts and training feature-based machine learning algorithms, but the results did not seem promising.

3.5 Results and Discussion

For all participants in the range of speeds 3.0 – 7.0 mph, the mean absolute error of step frequency was 0.12 steps/sec, 0.075 steps/sec for 3 seconds and MAPE was 0.73% for 40-second interval. Figure 3.2 shows the MAPE values for estimating step counts at different speeds while using 40-second intervals. We observed a higher MAPE for lower speeds, i.e.,
at 3 mph and 3.2 mph, caused by the slower movement of hands during walking that makes the sensor susceptible to capture the noise from smaller movements in the body. In addition, steps taken during walking speeds are lower than running speeds, which causes a higher penalty for error at lower speeds than at higher speeds. The variation of step frequency among three participants at different speeds can be seen in Figure 3.1. The step frequencies over different speeds show a gradual increase from 3.0 mph to 5.0 mph and then reach a plateau where they cease to increase with speed. At this point, the only way to achieve a higher running speed is by increasing the step length to compensate step frequency. Since the step frequency is not very indicative of speed and finding the step length is a challenging task, we resort to finding the speed by using the raw data, which is discussed in the next chapter.

Figure 3.2: MAPE of step frequency (on y-axis) at different speeds.
4.1 Speed Estimation Using Machine Learning Algorithms

In this section we evaluate several regression algorithms with Dataset-1 and Dataset-2. Then we build a hybrid two-stage classification-regression algorithm using Dataset-1. For all the machine learning algorithms, we used grid-search hyperparameter tuning to choose the optimal parameters for training each algorithm. For evaluation, we used train-test-evaluation split, where we used 70% of data for training and 30% for testing.

4.1.1 Regression

The regression algorithms used in this section along with their accuracies are shown in Table 4.1. Among these algorithms, random forest regression had a better performance with an MAE of 0.38, which performed twice as well as compared to other algorithms. On the other hand, using handcrafted features of Dataset-2 with regression, we had a further improvement in the performance with MAE as 0.27. After training the random forest on 128 handcrafted features in Dataset-2, we ranked the features based on their importance. Starting with the top two features to 128 features, we iterated and re-trained the algorithm and reported the corresponding accuracy at each step (as shown in Figure 4.1). In this analysis, we found that best performance could be achieved by just using top 33 features. These features along with their importances are shown in Figure 4.2. Among the top 33
features we observed that the most important features are those based on accelerations in the x-axis direction. This bias towards the x-axis was caused due to alignment of x-axis of sensors towards transverse plane of human body as opposed to the coronal or sagittal plane.

Table 4.1: Results of machine learning regression algorithms for speed detection

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$MAE$</th>
<th>$MAPE$</th>
<th>$R^2$</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Regression</td>
<td>0.75</td>
<td>12.5</td>
<td>-1.33</td>
<td>1</td>
</tr>
<tr>
<td>Random Forest Regression</td>
<td>0.38</td>
<td>8.9</td>
<td>0.62</td>
<td>1</td>
</tr>
<tr>
<td>Random Forest Regression</td>
<td>0.27</td>
<td>6.1</td>
<td>0.83</td>
<td>2</td>
</tr>
<tr>
<td>Gaussian Random Process</td>
<td>0.72</td>
<td>16.4</td>
<td>0.37</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.1: Effect of handcrafted features on random forest regression
In the hybrid algorithm, the first stage is a classifier that classifies speeds into low and high speeds. However, low and high speeds are subjective to change for each person, which makes it challenging to define criteria for a threshold to separate low and high speeds. For this, we experimented by training a random forest classifier with different speeds as a threshold in the speed range of 3 – 7 mph and found that 4.2 mph served as a better threshold, with an accuracy of 95%. Therefore, in our final analysis, we deemed speeds \( \leq 4.2 \) mph as low speeds and speeds \( > 4.2 \) mph as high speeds. The results for different classifiers are shown in the Table 4.2.

At the second stage, we trained two regression algorithms for lower and higher speeds. The mean results for lower and higher speeds are shown in Table 4.3. The random forest classifier and random forest regression algorithm showed the best results with an MAE of
Table 4.2: Results for classifier algorithms to classify between low and high speeds

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 – Score</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Vector Machine</td>
<td>0.91</td>
<td>0.95</td>
<td>0.93</td>
<td>93%</td>
</tr>
<tr>
<td>Random Forest</td>
<td>0.95</td>
<td>0.97</td>
<td>0.96</td>
<td>95%</td>
</tr>
<tr>
<td>K Nearest Neighbour</td>
<td>0.93</td>
<td>0.93</td>
<td>0.93</td>
<td>94%</td>
</tr>
</tbody>
</table>

0.33. Here we saw an improvement in the preciseness of the algorithm by leveraging a hybrid approach, but the $R^2$ results were worse compared to using just regression in Table 4.1. This was due to relatively poor performance for the algorithm at lower speeds.

Table 4.3: Results for hybrid two-stage classification and regression algorithms for speed detection

<table>
<thead>
<tr>
<th>Regression</th>
<th>Classification</th>
<th>MAE</th>
<th>MAPE</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest Regression</td>
<td>Support Vector Machine</td>
<td>0.37</td>
<td>14.3</td>
<td>-1.89</td>
</tr>
<tr>
<td>Random Forest Regression</td>
<td>Random Forest Classifier</td>
<td>0.33</td>
<td>12.2</td>
<td>-1.1</td>
</tr>
<tr>
<td>Gaussian Random Process</td>
<td>Random Forest Classifier</td>
<td>0.47</td>
<td>18.1</td>
<td>-3</td>
</tr>
</tbody>
</table>

4.2 Speed Estimation Using Neural Networks

We employed two strategies to train our CNN model and only the first strategy for RNN model. In the first strategy, we used train-test-evaluation split, where we divided the dataset into three subsets: 70%, 15%, and 15% for training, fine-tuning, and final evaluation for the model respectively. In the second strategy, we employed a leave-one-out cross-validation method where the model was trained on the data from 14 participants and evaluated against the data from remaining one participant. We cycled this process 15 times so that each participant’s data was included once in the evaluation and the mean results are shown for
all participants. Using the first strategy, we employed randomized hyperparameter tuning to search for the optimal parameters fitting both RNN and CNN models.

### 4.2.1 Recurrent Neural Networks (RNNs)

Recurrent neural networks (RNNs) can deftly identify the correlations between neighbouring datapoints, and bidirectional RNNs have the capacity to interact with values in the past and future [60]. However, RNNs fail to capture long-term dependencies as their gradients tend to vanish or explode for long sequences, according to Bengio et al. [61]. Gating methods such as gated rectified unit (GRUs) and long short term memory (LSTM) are two kinds of RNNs that have increase in popularity in recent years for sequential data. Both GRUs and LSTMs have been shown to be superior to traditional RNNs, but it is hard to rank the two gating methods [62], as they might be suitable for different applications. In our study, we have used both GRUs and LSTMs and had best performance with a bi-directional LSTM. Our best model had an MAE, MAPE, and $R^2$ of 0.21, 4.9, and 0.93 respectively. For this model, the total number of trainable parameters were 225,300. The large number of trainable parameters increased the overall model size and this led us to switch to convolutional neural networks, which have relatively lower implementation size.

### 4.2.2 Convolution Neural Networks (CNNs)

Convolution neural networks (CNNs) are competent in automatically extracting the relevant features from the input signals [63]. It has achieved promising results in solving image classification [63], speech recognition, and text analysis problems [64]. In this work, we experimented with the different CNN architectures and their sensory inputs as either of ac-
celerometer, gyroscope or both included. Among the three combinations, models trained on both accelerometer and gyroscope reported a better MAPE score compared to single sensor. We showed in the results section that our model outperformed the CNN model with only accelerometer as input [56]. Additionally, we observed that learning from each sensor independently in initial layers and concatenating them later made the model more accurate and compact in size. Therefore, we designed a CNN architecture (as shown in Figure 4.3) that has two symmetrical branches. In the input layer, the first branch takes its input from the accelerometer and second from the gyroscope. Input layer is followed by the convolutional layers in each branch.

Figure 4.3: Architecture of CNN-based speed detection algorithm.

To optimize the hyperparameters, we performed a randomized search [65] on the following range of values: number of convolution layers (2 – 10), number of filters in the convolution layers (10 – 100), number of dense layers (2 – 5) and number of neurons in each dense layer (15 – 500). We randomly sampled 50 settings of hyperparameters and kept the one which performed best. Our hyperparameter search led us to add two convolutional layers; the first layer consists of 27 filters and the second 45 filters, where each filter size is $3 \times 3$. These convolution layers extract both high-level and low-level representations of the data. The output of these convolutional layers is passed to a global max-pooling layer which distills the output of the convolution filters into a salient vector of size 45 by extracting the maximum value.
from each filter. After pooling, the outputs of accelerometer and gyroscope convolutional layers are concatenated to fuse the information from both layers, which produced a vector of size 90. Finally, this vector is fed to the two sequential dense layers. The two dense layers have 180 and 30 neurons respectively. Output of second dense layer is forwarded as an input to the single neuron output layer that estimates the final speed.

4.2.2.1 Training Procedure of CNN Model

Training of the CNN model consisted of three main processes: feedforward propagation, loss function and backpropagation. In the feedforward process, convolution filter weights and neuron weights are first initialized with Xavier initialization [66]. Each data sample is then passed through the architecture and the output after each layer is obtained through a rectified linear unit (ReLU) activation layer [67]. After all data samples pass once through the architecture, one epoch is completed. Then the training loss is calculated with mean absolute error (MAE) loss function, which computes absolute difference between predicted and true speeds. The weights of the architecture are then updated with root mean square propagation (RMSprop) optimizer [68], which works similarly to the gradient descent with momentum that adjusts the weights of all trainable parameters with a learning rate of 0.001. Additionally, to ensure that our model is not suffering from the overfitting problem, we leveraged the dropout [69] regularization technique. We trained our model for 1000 epochs and used an early stopping criteria that stops the training if the validation MAE loss was not decreasing for 10 consecutive epochs.
4.3 Results

For the train-test-evaluation split evaluation strategy, our model achieved 0.18, 4.2%, and 0.94 value of MAE, MAPE and $R^2$ respectively as shown in Figure 4.4. In this evaluation strategy, the same participant data can be present in the train, test and evaluation subsets which can induce bias in the results. To avoid this, we evaluated our model on a more stringent strategy: leave-one-out cross-validation, that achieved 0.48, 9.8%, and 0.62 values for MAE, MAPE and $R^2$ respectively. As shown in Figure 4.5, for both strategies, predictions are fitted close to the regression line at different speeds. At higher speeds, the model has high error rate for the leave-one-out strategy, likely because each participant exhibits a different running posture at higher speeds. Since the participant data that is used for evaluation is not included in the training, it is difficult to capture those patterns. A detailed account of all our best performing algorithms is presented in Table 4.4. We see that random forest gives the best results only with handcrafted features. Better precision can be achieved through neural networks, but most of the LSTM-based neural networks have too many parameters, which make the models bulky to implement in smartwatches. Therefore, we used CNN, which has better precision with less parameters. Finally, we propose a quantized version of a CNN model that shows a considerable improvement in inference time with a minimal loss in precision.

In order to validate our results with previous work, we compare our results with the study by Wang and Xu [56]. Their study used one-dimensional convolution neural networks with kernel size of 6 and achieved a mean error of 7% - 18% for running and walking speeds while we had 4.2% for train-test-evaluation split and 9.8% for leave-one-out cross-validation (they didn’t evaluate on this). Our research displayed higher precision in a thorough evaluation
Figure 4.4: Results of CNN-based speed detection algorithm against train-test-evaluation split and leave-one-out cross-validation strategies.

by leaving one participant out in a condition that depicts a practical use case of a device where a person expects higher precision to plan their fitness regimen.
Figure 4.5: True vs. predicted speed for train-test-evaluation split (top) and leave-one-out cross-validation (bottom).

4.4 Making the Speed Detection Model Compatible with Edge Devices

To make the model compatible with edge devices such as smartwatches or a wrist-worn wearable, we need to make the algorithm adhere to limited computation speed and battery
Table 4.4: Results of our speed detection algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>MAE</th>
<th>MAPE</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forests + Random Forest Classifier</td>
<td>0.33</td>
<td>12.2</td>
<td>-1.1</td>
</tr>
<tr>
<td>Random Forests with dataset-1</td>
<td>0.38</td>
<td>9.8</td>
<td>0.62</td>
</tr>
<tr>
<td>Random Forests with dataset-2</td>
<td>0.27</td>
<td>6.1</td>
<td>0.83</td>
</tr>
<tr>
<td>Bi-directional LSTM</td>
<td>0.21</td>
<td>5.2</td>
<td>0.87</td>
</tr>
<tr>
<td>Conv LSTM</td>
<td>0.20</td>
<td>5.0</td>
<td>0.90</td>
</tr>
<tr>
<td>CNN</td>
<td>0.18</td>
<td>4.2</td>
<td>0.92</td>
</tr>
<tr>
<td>Quantized CNN</td>
<td>0.21</td>
<td>4.8</td>
<td>0.89</td>
</tr>
</tbody>
</table>

life. In order to achieve this, we first use separable convolution layers instead of convolutional layers in our model architecture, followed by post-training quantization in our neural network architecture to crunch the model’s size and reduce the computational need to achieve a better inference speed.

4.4.1 Separable Convolution Layers

This approach was first suggested by Sifre [70] and became popular for computer vision problems that aim for reduced model size for better performance on mobile devices. This concept was first utilized in Xception by Chollet [71], followed by MobileNet [72] and ShuffleNet [73], which achieved state-of-the-art performance in a few applications. The separable convolution layers come from depthwise and pointwise convolutions. A depthwise separable convolution convolves in the spatial domain in the horizontal axis, followed by a pointwise convolution in the vertical axis that projects the channel’s output of depthwise convolution into new space. Therefore, by this approach, we factorize the $3 \times 3$ matrix into $3 \times 1$ and $1 \times 3$ matrices. Using this method, we have reduced the total trainable parameters from 44,341 to 24,973 by using the same hyperparameters as discussed in Section 4.3.
4.4.2 Pruning

Pruning is essential in resource-constrained environments such as smartwatches where bigger computations are taxing on the energy spent. Therefore, methods such as pruning [74] will help reduce the size of the over-parameterized neural network model by compressing the sparse weights in the graphs. This will result in model size reduction with a compromise for the model’s precision. In our model, when we applied post-training pruning, the MAE increased to 0.28 and hence we did not use pruning.

4.4.3 Quantization

Quantization is a process of downcasting floating-point digits from a higher dimension to a lower dimension [75]. Due to this, we can achieve faster inference time and reduce computational overhead on edge devices. In our model, we have quantized the weights from float-32 format to float-16 format at each layer and for each activation function. When we applied quantization technique to the model in Section 4.4.1, the inference time decreased from 172 ms to 82 ms, giving a two-fold improvement in the inference time. The increased efficiency of the model was achieved with a negligible precision loss of 0.03 mph for MAE. The resulting model had an MAE, MAPE and $R^2$ of 0.21, 4.8, and 0.89 respectively.
CHAPTER 5

CONCLUSION

5.1 Discussion and Summary

In this thesis we have collected data using a Shimmer sensor while participants were using the treadmill. We have collected 9,225 seconds of data from 15 participants. During the data collection process, we have followed the protocols given by the Institutional Review Board (IRB) of Northern Illinois University and all procedures were in compliance with the ACM code of ethics [76]. The participants were instructed to walk or run as they would do in everyday life. The participants were free to choose the hand for wearing the sensor; as a result, two participants wore the Shimmer device on the right wrist and 13 on the left wrist. In our data, each participant had a unique style of walking/running and their choice of walking or running at a given speed also varied. The varying style of movement, different heights of participants, and wearing the device on both left and right wrists led us to collect data that is realistic in everyday life. Participants also had different fitness levels and while most of them were able to walk/run at speeds $\leq 6$ mph, only some were able to run at higher speeds ($\geq 6$ mph). Due to this, at running speeds 6.2 mph and 6.7 mph we had only one session recorded, which was discarded in the leave-one-out cross-validation because it was leading to bad results.

The methods in this research were shown to perform well for indoor walking and running conditions on a treadmill. But the applicability of our algorithm was not tested in outdoor conditions. In outdoor conditions, obtaining precise ground truth for speed is challenging...
and is more prone to noise and errors as compared to indoors. However, the availability of both outdoor and indoor walking/running data from a broader range of participants will enable us to fine-tune our algorithm and build a more generic and more precise model.

Our analysis showed that random forest regression with heuristics gave higher precision. Since handcrafted features do not have generalizing capacity, we used convolution neural networks and achieved better precision. Comparing our results with existing literature shows that by using wrist-worn wearables we can achieve similar precision to other body-worn sensors such as shank-mounted [6] or chest-mounted [53] sensors. In addition, we showed that our proposed model achieved better precision in detecting speed on the treadmill compared to the other solutions based on the wrist-worn devices [56].

- We collected tri-axial inertial sensor data from 15 participants. In addition, we used a medical-grade device called Shimmer to capture the data precisely.

- We evaluated a frequency-based step counting algorithm that uses fast Fourier transforms to detect the step frequency without the need for a complex model.

- We proposed a CNN-based speed detection model that has achieved 0.18, 4.2%, and 0.94 values for MAE, MAPE and $R^2$ respectively for all participant's train-test-evaluation model. And for the leave-one-out evaluation, it achieved 0.48, 9.8%, and 0.62 values for MAE, MAPE and $R^2$ respectively.

- We also applied quantization and pruning techniques to improve the inference time of the model, reduce its size, and make it compatible for current edge devices.
5.2 Limitations

Our study has several limitations in regards of applicability in commercially available devices, usability among users of different generation, and understanding the behavioral impacts of speed detection on wearable devices.

We collected the sensory data from Shimmer, a medical-grade device, but commercially available devices are not the same and could lead to different error rates in capturing the data. Thus choosing the type of watch might affect preciseness of our algorithm. Another limitation is based on our dataset; we collected data from college students with different fitness levels, heights, walking styles, and equally distributed genders, but we did not evaluate on much younger and older people. Perhaps data from a diverse set of people from different demographics and ages will help us build a more generalized model. In addition, collecting data from the same participant after a few months might will enable us to see the effect of changing walking style within the same participant.

On the behavioral side, we did not catalogue the emotional impact of the device or track the behaviors of participants in our study. Hence, we were unable to gauge the psychological impact of a speed-detecting device on a participant. Some users might be prone to injuries due to overexertion or due to inaccurate recommendations given by smartwatches. This issue could be tackled by improving human-wearable interaction by proper fitness education and precise fitness tracking. Clermont et al. [77] showed that tracking running data helped recreational runners to find motivation in running, whereas competitive runners used it to prevent injuries and increase performance. In addition, recreational runners showed interest in basic metrics such as speed, distance, and steps taken, whereas competitive runners were interested in advanced metrics such as pace, vertical oscillation, and stride length.
5.3 Future Work

Our speed detection model based on CNN regressor architecture has achieved favorable results in train-test-evaluation split and leave-one-out cross-validation strategies. The higher precision of our solution makes it particularly useful to elderly citizens who are advised to be as active as possible. An error-prone device can discourage them from exercising. Unlike GPS, our IMU sensor-based wearable solution can work well indoors as well as outdoors. Since step frequency doesn’t change significantly when speed increases in walking or running, it becomes difficult to improve the precision. To overcome this challenge, in future study, we will design a two-phase speed detector model, where in first phase, we will classify running vs. walking and then, in second phase, will estimate the speed within that activity. In addition, we will generate artificial data, append it to our dataset to test more complex neural network based algorithms to further improve the precision, and apply different quantization techniques to make the algorithm compatible with smartwatches.
REFERENCES


