Smartphone Gait Authentication:

Recognizing Smartphone Users based on their Gait

A Major Qualifying Project

Submitted to the Faculty of

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

Thar Min Htet

Date: May 7, 2020

Submitted to:

Professor Emmanuel Agu of

Computer Science Dept,

Worcester Polytechnic Institute

Abstract

Gait recognition using smartphone motion sensors such as accelerometers and gyroscopes is relatively underdeveloped compared to those using machine vision. This project explored the various state of the art neural networks-based approaches for accelerometer and gyroscope-based gait analysis and evaluated them. CNN and LSTM neural networks architectures proposed in prior work are replicated to achieve similar results on a gait dataset gathered in the wild. Prior work focused deep learning models for gait recognition on data gathered in controlled user studies.

Acknowledgments

I would like to thank Professor Emmanuel Agu for his insight and guidance throughout this project.

I would also like to thank my family and close friends for their support and Worcester Polytechnic Institute for allowing me the opportunity of working on the project.

Table of Contents

List of Figures

List of Tables

1 Introduction

Human "gait" refers to locomotion achieved through the movement of human limbs. From a technical view, methods for biometric gait recognition can be categorized into three main approaches: Machine-vision based, Wearable sensor-based, and Floor sensor-based [3]. Many successful attempts for gait analysis using videos, (i.e. machine vision) have been made. In one of the machine vision approaches, motion of joints such as ankles, knee, and hips are extracted to assist in the recognition task [11]. Floor sensor-based approaches are relatively less explored due to the dependence on the hardware. Gait analysis using wearable accelerometers is relatively under-researched and robust implementations of machine and deep learning gait analysis models are not only very rare to find but also very limited on the type of sensor.

1.1 Motivation for wearable sensor-based approach

From a consumer perspective, the availability of accelerometer-based gait recognition would be very beneficial and convenient because of the ubiquity of motion sensors that are widely available in mobile gadgets. Mobile gadgets that are equipped with motion sensors include smartphones, smartwatches, fitness trackers, and other gadgets that typically have a built-in accelerometer and a gyroscope. The wearable technology market is expected to grow to over 57 billion USD by 2025 [15]. This doesn't include the market values of smartphones, the most omnipresent gadget. As of February 2019, the Pew Research Center estimated that more than 5 billion people own mobile devices [1]. Among those, 3.5 billion people owned smartphones as of February 2020. This is 45.12% of the whole world population. [12] Despite the huge potential impact they could have, the accelerometers are only commonly used for basic operations such as measuring acceleration and stabilization. This project aimed to utilize

accelerometer and gyroscope data gathered from smartphones for gait recognition. Figure 1.1 shows the wearable tech market opportunity analysis from Tractica.

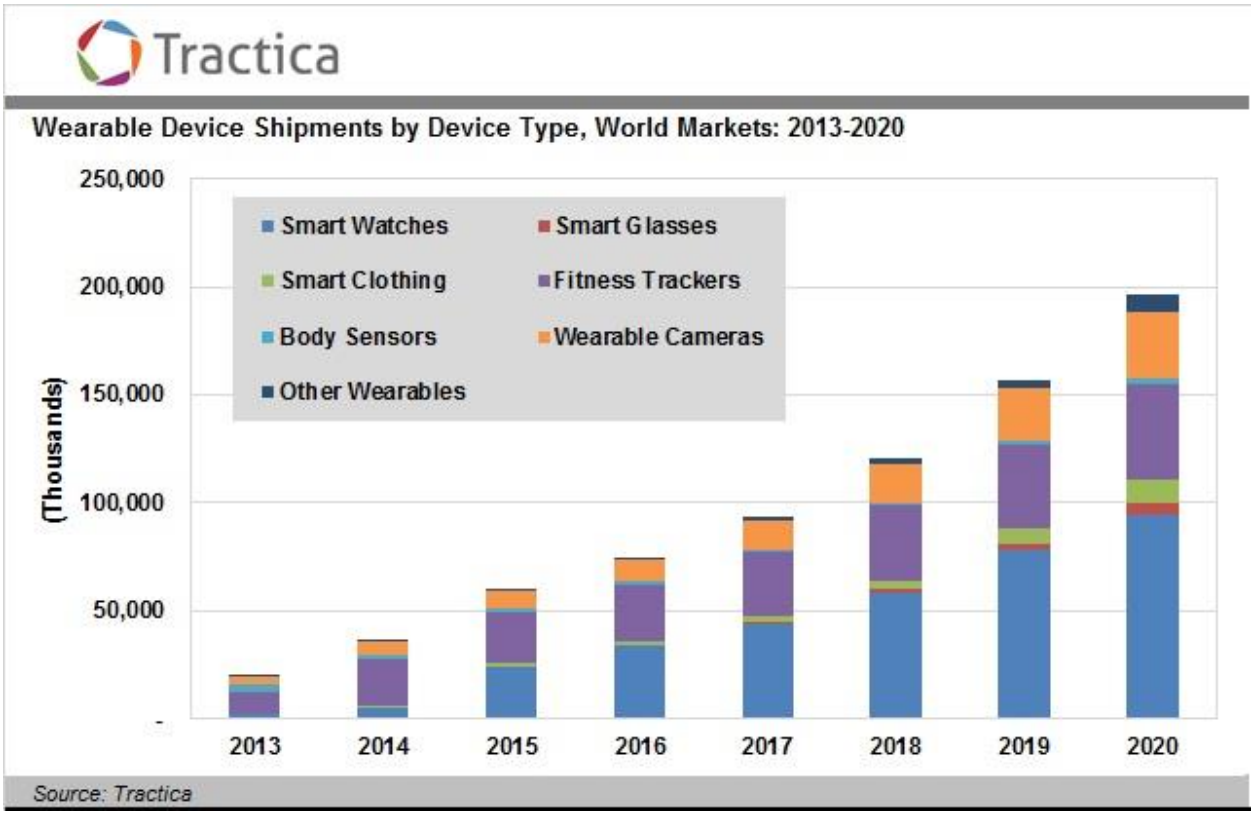

Figure 1.1: Wearable Technology Market Opportunity Analysis. [13]

1.2 Understanding the Human Gait Cycle and Phases

It's essential to understand what a gait means both in the literal and technical forms. Human gait consists of two steps, one from each leg. In technical terms, a step can also be referred to as a single stride. Stride is defined as a long, decisive step [25].

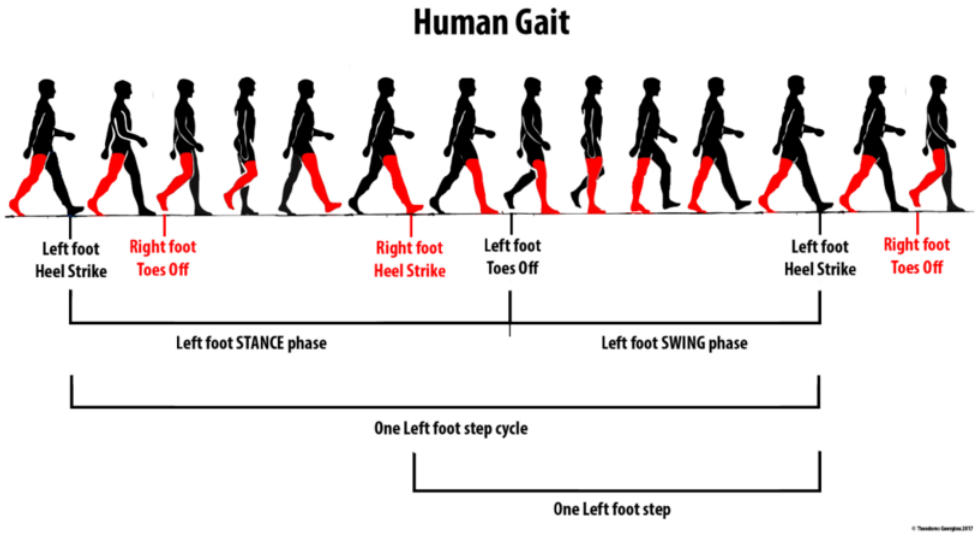

Figure 1.2: One full cycle of Human Gait. The last two states are excluded from the cycle [14]

The accelerometer values generated by the gait in figure 1.2, which is made up of 3 axes, are shown in figure 1.3 in an ideal case. Note that one of the 3 accelerometer axes are shown.

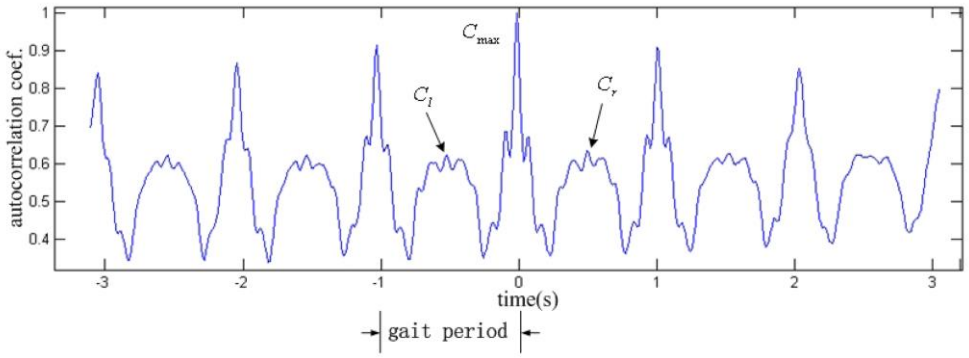

Figure 1.3: The depiction of a gait cycle in terms of accelerometer values [16]

In Figure 1.3, a gait cycle is the distance between two similar peaks. This can either be tall peaks or short peaks. However, in reality, unprocessed smartphone sensor data is very noisy and complicated compared to the data used above. One of the most challenging parts of accelerometer gait analysis is deducing what a cycle means from the accelerometer values. It's

impossible for a human to manually detect cycles on real-life complicated data. Figure 1.4 is sample data from the WISDM dataset, an unprocessed real-life dataset. The WISDM dataset consists of raw time-series accelerometer values collected on subjects who performed various movement activities such as walking and jogging.

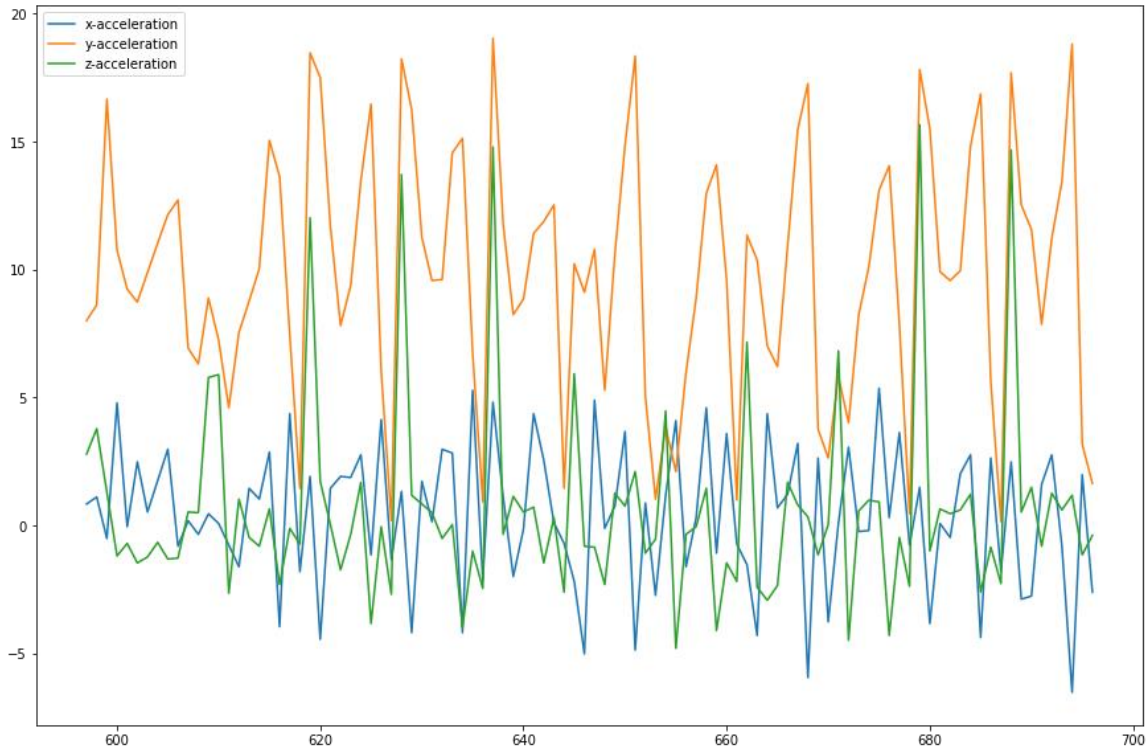

Figure 1.4: Plot of 100 continuous accelerometer values from a single subject

As seen in Figure 1.4, it's not practical to pinpoint where a gait cycle starts and ends. Thus, a significant amount of effort involved in this project was spent on methods for automatically deducing the start and end of a gait cycle. There has been extensive research done on this task. To keep the introduction simple, the specific algorithms and methods used to detect a gait cycle in this project will be discussed further under methodologies.

1.3 Machine Learning

Machine learning is a branch of Artificial Intelligence (AI) that is based on the theory that machines can learn algorithms from example data without being specifically programmed to perform certain tasks [17]. Machines can do so by recognizing reliable patterns within the data. There are four major types of machine learning: (1) Supervised learning, (2) Unsupervised learning, (3) Semi-supervised learning, and (4) Reinforcement learning [18]. In supervised learning, algorithms are trained using labeled examples, such as an input where the desired output is known. Unsupervised learning is used when dealing with data that has no labels or correct answer. Semi-supervised learning lies in between supervised and unsupervised learning and is usually used for the same applications as supervised learning. Lastly, reinforcement learning is often used in robotics and gaming, where the computer learns through trial and error.

In this project, we mainly used supervised learning where the input is a set of smartphone accelerometer and gyroscope data points that represent a user's walking pattern and the labeled output is the identity of the person represented as integers. Neural networks are designed to mimic the neurons of a human brain and contain neurons that are interconnected to process the input and calculate the output. Neural networks are also the foundation of deep learning, which is the major AI technique used in this project.

1.4 Deep Learning

Deep learning is a subset of machine learning that excels when the amount of data is massive and the complexity of the data is high[23]. Deep learning has exploded in popularity in the past decade due to the availability of data in the digital era. Deep learning utilizes a hierarchical level of artificial neural networks, for machine learning.

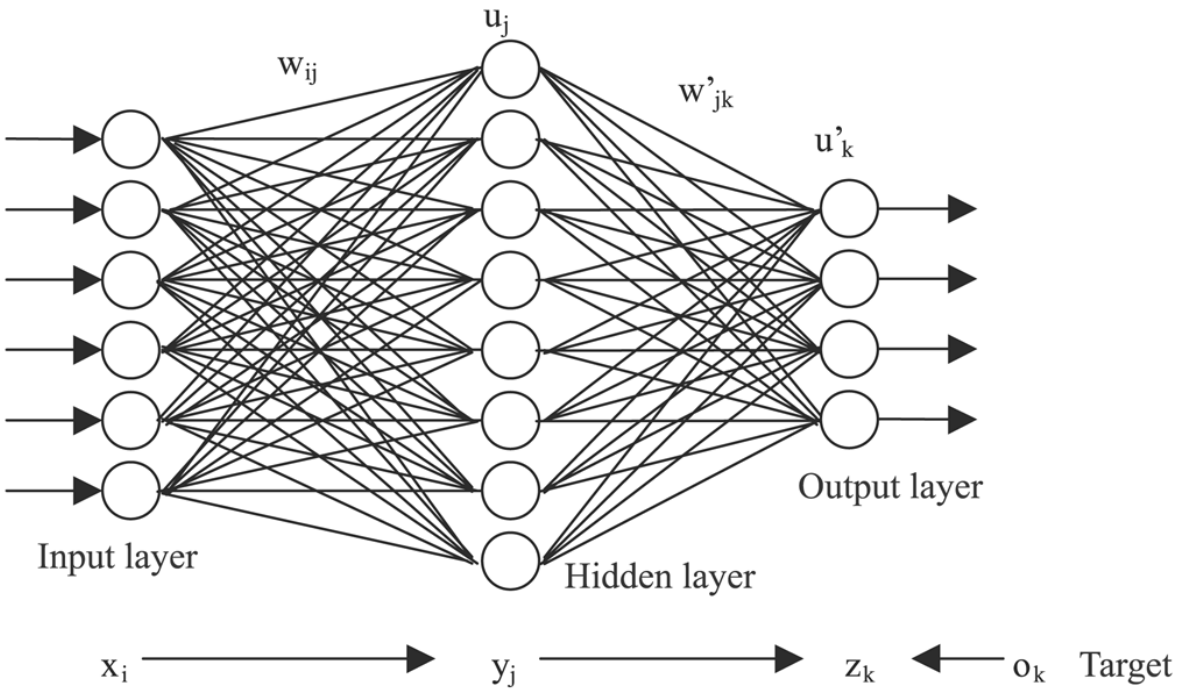

Figure 1.5: Artificial neural networks

The neural network depicted Figure 1.5 has an input layer, a hidden layer, and an output layer. The input layer is where the data comes in. The hidden layer is where the calculation happens and the output layer is where the predicted result is output. However, typical deep learning architectures have more hidden layers than this, and many more nodes or neurons involved in the architecture. Neural networks are used in gait pattern recognition because they are extremely good at recognizing complex patterns even when it is impossible for humans to perform the task. Convolutional neural networks (CNN) and Recurrent Neural Network(RNN) are used in this project. CNNs are good at feature extraction and RNNs excel in processing time-series data such as accelerometer data. The details of the specific architecture of the convolutional neural network and the Recurrent neural network utilized in this project will be further discussed under methodologies.

1.5 Gait Recognition Dataset

The datasets of this project are provided by another research group that authored the paper "Deep learning-based gait recognition using smartphones in the wild" [6]. This dataset was collected in controlled, laboratory conditions. Subjects walked while carrying their phones in a scripted fashion. In their paper, the researchers successfully achieved implementing identification and authentication of subjects based on their gait with high accuracy. In this MQP, we implemented the identification aspect based on the original paper. In identification, the classifier matches 1 to N, meaning it classifies the identity of a person out of N people. In authentication, on the other hand, the classifier compares the input existing data to deduce the similarity between the two and verifies the identity of the person. The basic goal of that original project was to distinguish one person's gait from another using data in the above dataset that was collected while the subjects were "walking". There are 8 different datasets included in this project, but only 2 of them will be analyzed for gait identification. Dataset #1 has 118 subjects and is preprocessed based on the step-segmentation algorithm breaking them into two-steps or one gait cycle. Dataset #2 was collected from 20 subjects and the gait curve is divided into two-step samples and interpolated into the length of 128 using linear interpolation function (Equation 1.1). The data was analyzed using various approaches that ranged from machine learning/deep learning to statistical modeling.

$$\frac{y - y0}{x - x0} = \frac{y1 - y0}{x1 - x0}$$

Equation 1.1: Linear Interpolation Function

1.6 The Goal of the MQP

The project investigates neural networks architectures proposed for gait recognition and evaluates them a smartphone dataset gathered in the wild. The project focuses on reproducing the results CNN and LSTM architectures from published in academic papers on the dataset discussed in Section 1.5. The neural networks described in the paper were implemented and all the results from various approaches are compared in the following sections.

2 Related Work

Numerous research has been conducted in the area of gait analysis. Most of the research is based on computer science and mathematical modeling principles. The research papers reviewed below have been utilized directly or indirectly to complete the current project.

Mohammad Omar Derawi [3] explains the problem of biometric gait analysis into different categories and categorizes existing approaches to accelerometer-based gait analysis, a sub-category of wearable sensor-based gait recognition. The paper initially presents the signal processing flow of the method for gait identification. The flow consists of (1) acquiring data from accelerometer, (2) signal acquisition and preprocessing, (2) data analysis, and (4) comparison. Then, data analysis is further broken down into different methods such as segmentation, and feature extraction in the time and frequency domains. Different comparison approaches are also discussed featuring statistical modeling and classification approaches. This paper and the main steps it outlines have been utilized as the foundation of this project as it reviews all the approaches that existed at that time.

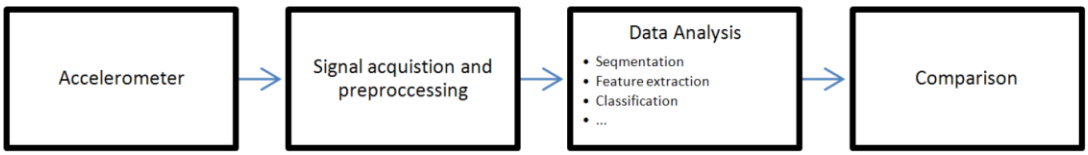

Figure 2.1 Signal processing flow of the method for gait verification/identification [3]

"Identifying People from Gait Pattern with Accelerometers" [4] directly relates to the goals of the current project. The approach from this paper involves 1 step cycle detection algorithm and average cycle detection. This algorithm is used to calculate a 'template' model, in which subsequent data can be compared to see if they are similar. Then, new signals from the accelerometer are collected and the same algorithm is applied to get the data in the same format

as the template. The similarity is calculated using cross-correlation (C). If C is greater than the predetermined threshold, it's accepted as the same person. They achieved an equal error rate of 6.4% in experiments with 36 test subjects. This method was utilized as the first approach explored in this MQP.

"Gait recognition under non-standard circumstances" [5] discusses how different circumstances influence the ability to accurately perform authentication. The goal of the paper is to see if it's possible to recognize users under different circumstances such as walking in different terrain from the data. This project utilized weighted moving averages, a signal processing approach utilized in that paper.

2.1 IntoxiGait Deep Learning

Intoxigait was developed by Worcester Polytechnic Institute students as Major Qualifying Project (Bremner, Cheung, Lam & Huang, 2018). The goal of their study was to compare deep learning to existing machine learning methods in its ability to predict the blood alcohol concentration of a user by training a convolutional neural network and creating a mobile app that could accurately determine the intoxication level. They collected data from 38 participants over 12 weeks, gathering accelerometer and gyroscope data simultaneously from both a smartwatch and smartphone. They then preprocessed the data and fed it into their neural network resulting in 64% on the test set and 69% on the training set for five BAC ranges. Figure 2.2 demonstrates the confusion matrix for the main output in Intoxigait.

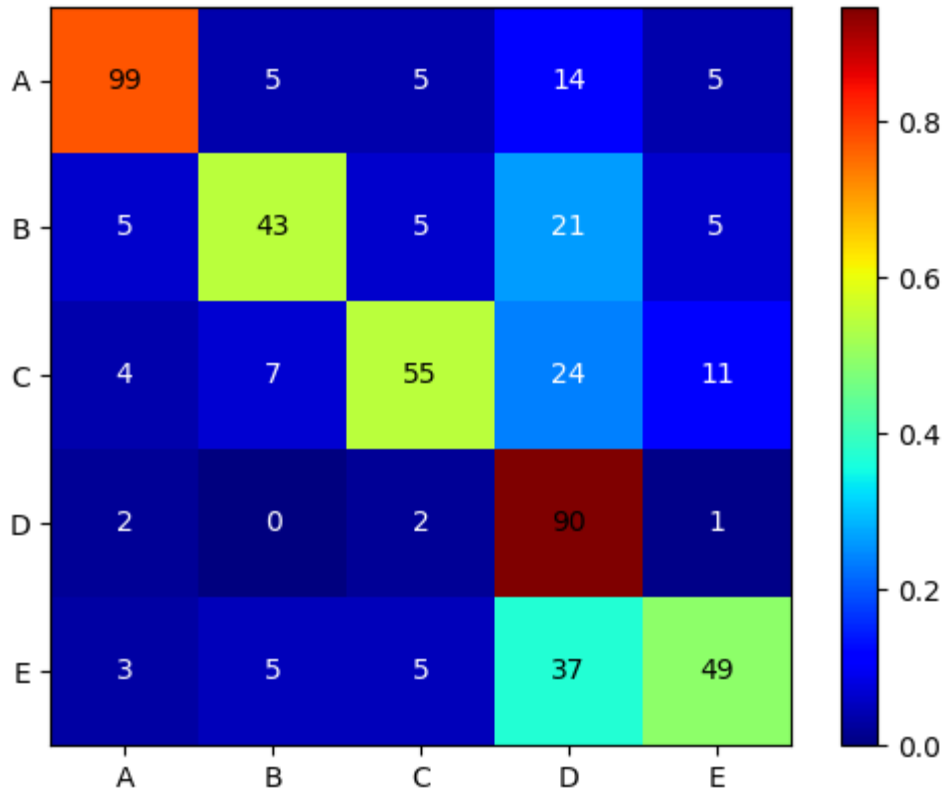

Figure 2.2: Confusion matrix for the main output in Intoxigait [19]

The team used an architecture that contains two inputs, both of which are fed into individual convolutional neural networks, which is served to extract features. Then the layers are merged to then fed again into a multilayer perceptron classifier with dropout for regularization, which helped in reducing overfitting and enhanced the model's ability to generalize better.

2.2 Smartphone Gait Inference

Smartphone gait inference was developed by Worcester Polytechnic Institute students as Major Qualifying Project (Arnold & LaRose, 2015). The goal of the project was to test different machine learning approaches to predict someone's intoxication level from smartphone accelerometer data. They collected accelerometer data from seven individuals over two weeks. They then extracted features such as cadence and applied multiple machine learning classifiers to predict the number of glasses of alcohol consumed. They applied techniques such as random

forest and were able to train the model to get 56% on the training set and 70% on the testing set. They then proceeded to incorporate the machine learning model into a mobile application that could measure smartphone users' levels of intoxication. Table 2.1 compares the accuracy, F-score, and ROC area across different users.

User	Accuracy %	F-Score	ROC Area
User 1	58.435	0.597	0.612
User 2	52.784	0.513	0.538
User 3	62.947	0.640	0.631
User 4	67.249	0.651	0.683
User 5	73.854	0.754	0.749
User 6	33.333	0.410	0.551

Table 2.1: Report of Usability Study Descriptive Statistics on Classifier Improvements

2.3 Identifying people from gait pattern with accelerometers

This paper was written by Heikki Ailisto, Mikko Lindholm, Jani Mäntyjärvi, Elena Vildjiounaite, and Satu-Marja Mäkelä, VTT Technical Research Center of Finland faculties. The objective of the study was to analyze accelerometer data and identify people within their dataset. Instead of relying on techniques such as deep learning, the team focused on more traditional methods such as gait cycle detection and a novel algorithm that they created. The approach works by dividing the signal into parts representing steps, normalizing and breaking down into a "gait code", a term they used to represent x, y, and z data. Then for identification, they used a special algorithm to find a correlation value and then make decisions by comparing that value with a predetermined threshold.

Their experiment contained 36 subjects and 3 trials resulting in 108 total trials. They also produced 1296 false trials. They were able to achieve a False Rejection Ratio of 6.4%, Total Error Rate of 5.4%, and Equal Error Rate of 6.4%.

2.4 IDNet

In IDNet (Gadaleta, Rossi, 2017), Rossi *et al* used Convolutional Neural Networks for user authentication framework from smartphone-acquired motion signals. Data was acquired from devices worn in pockets of user's trousers. Unlike the goal of this MQP, which is identification, the goal of the paper is to authenticate users. They used simple Convolutional Neural Network architecture and a one-class support vector machine to accurately perform the authentication.

Their experiment utilized motion data from 35 subjects which were used to train the CNN feature extractor. They were able to train the model to be both accurate and fast, having false-positive rates and false negative rates that are smaller than 0.15% for an appropriate choice of hyperparameters and thresholds.

2.5 Gait Recognition Using Smartphones

This paper was written by Qin Zou, Yanling Wang, Qian Wang, Yi Zhao, Qingquan Li. The goal of their project was to build a deep learning classifier that is robust to conditions such as the environment and activity [6]. They conducted their research on 118 subjects collecting accelerometer and gyroscope values. They investigated many approaches ranging from CNN to LSTM to a combination of both. They also did both identification and authentication on the data. The unique aspect of their paper was gait extraction, which makes the model able to correctly detect non-walking data such as running and climbing. This is the main inspiration for this MQP and a lot of our approach will be based on their studies and experiments.

3 Methodology

The project initially involved lots of pilot experiments using a lot of trial and error of various existing techniques to determine which approaches worked best for our goal of gait recognition. The results for most approaches explored will be presented in subsequent sections.

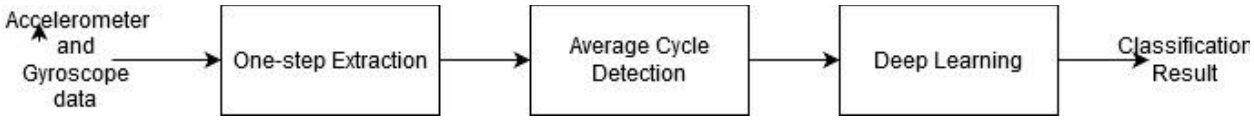

Figure 3.1: Flowchart for Methodology with Preprocessing and Deep Learning

The flowchart in figure 3.1 summarizes the steps in our methodology, which will be discussed extensively as follows.

3.1 Preprocessing Data

This approach tries to define what part of the accelerometer signal data makes up a single step. The flow of the approach involves signal processing, extracting templates using cycle detection algorithms, and comparing using cross-correlation. Figure 3.2 is the diagram for the process excluding signal processing.

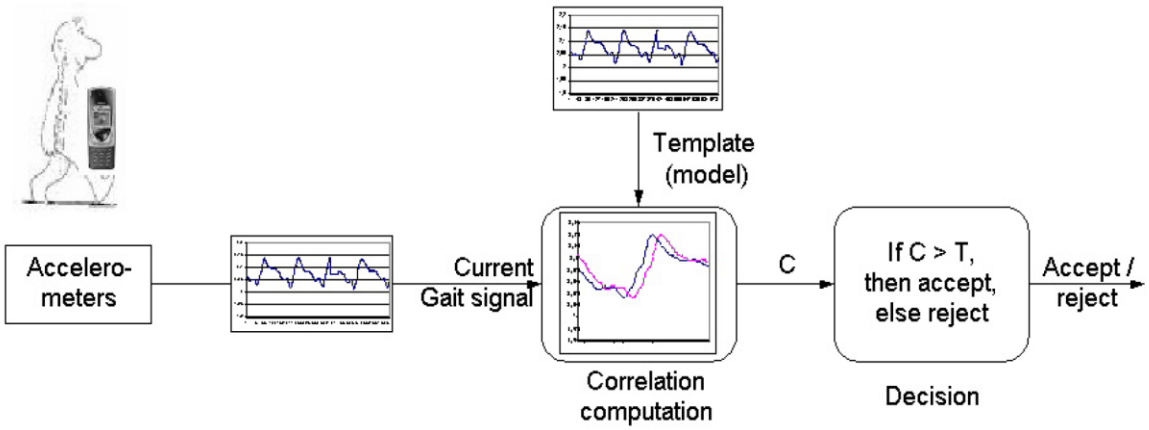

Figure 3.2 Block diagram of the gait based verification method [4]

Figure 3.2 shows the flow diagram of the signal processing to decision making. The first step of the approach is signal processing. The accelerometers are very sensitive to movement, resulting in a lot of unwanted noise. Weighted moving averages is applied in this case to smoothen the data, reducing the spuriousness of the data.

$$S_t = ax_t + (1 - a)S_{t-1}$$

Equation 3.1: Weighted Moving Average

Where S_t, S_{t-1}= Current/Previous raw data points, x_t = Previous process data point,

a = smoothening factor

We selected the Weighted Moving Average (WMA) over the Simple Moving Average (SMA) to determine the number of previous data points that impact the current data point. WMA is a simple statistical technique that puts more 'weight' on more recent data and less on the older data. It's very widely used in other types of data that are spurious such as stock data (See figure 3.3).

Figure 3.3: Comparison of the impact of WMA and SMA over stock data [20]

In the project, the optimal weight of the WMA is calibrated several times to get the optimal value. Figure 3.4 is a comparison of 100 x-accelerometer values of a particular subject before and after 0.8 is applied as the alpha (a) of WMA, which corresponds to the weight of five previous data points.

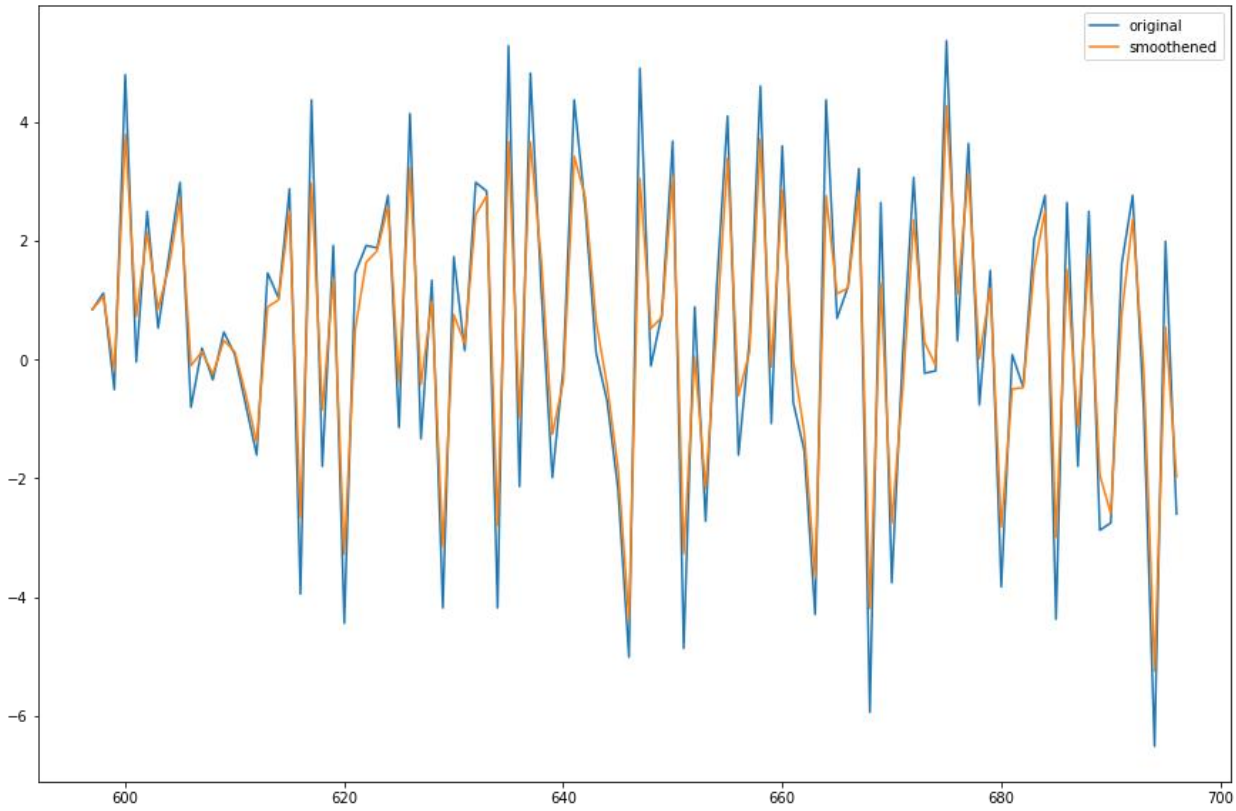

Figure 3.4: Comparison of data before and after WMA (x-acceleration)

The next step is to extract gait cycles from the data. One-step extraction and average cycle detection are used in this step to form a pre-calculated 'template' that future steps can be compared to measure similarities. This computation is done in the time-domain. The details of the algorithm are summarized as follows.

1. Divide the signal to parts representing steps.

2. Normalize the parts, so that their amplitudes and lengths are equal

3. Average "a-steps" and "b-steps" of signals x and z to form the gait code [xa xb ya yb za zb].

The left and right steps are separated and treated differently because they are not symmetrical. The steps can be found in each signal x (forward),y (sideways), z (vertical) by finding local 'minima' and local 'maxima'. The gait code [xa xb ya yb za zb] corresponds to the x, y, and z acceleration and left/right step pairs. The gait code can also be referred to as a 'template'. The original paper disregards y-acceleration due to its reduced impact on the template.

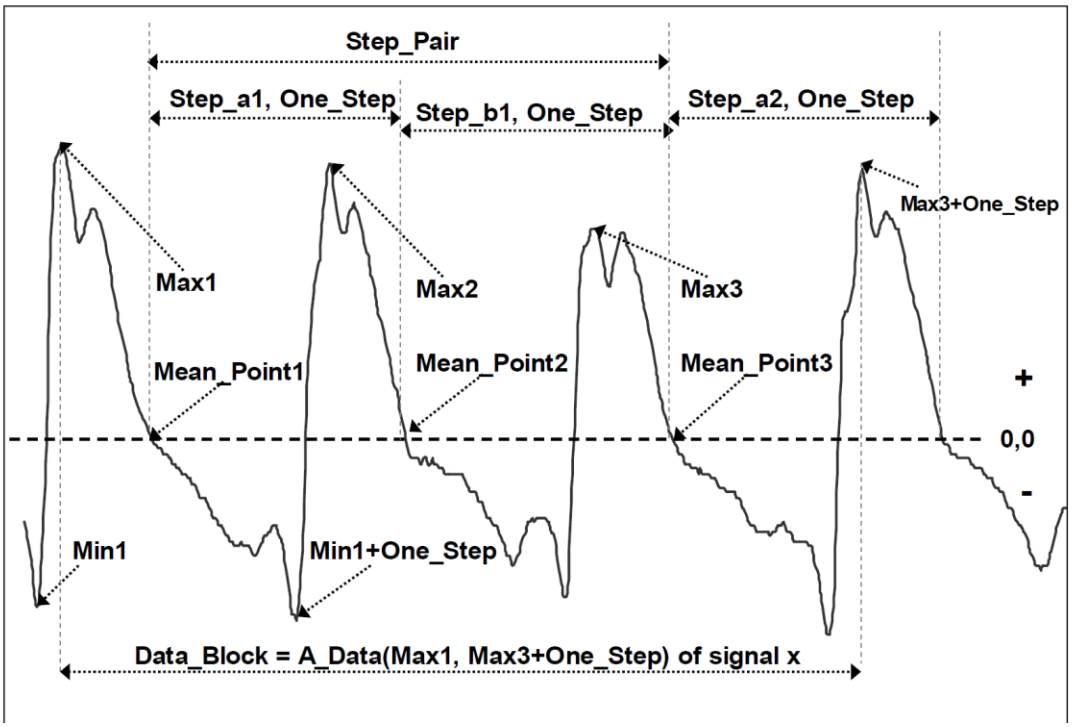

Figure 3.5: A time-domain signal x of acceleration gait data with the main parameters used in gait code extraction [4]

After the template is calculated from the signal using the cycle detection algorithm, the identification phase begins. The steps for the identification phase are as follows.

1. Repeat all steps from the cycle detection algorithm for the sample c and d steps, resulting in the gait code [xc xd yc yd zc zd]

2. C = Max ((c(xa,xc)+c(xb,xd)+c(za,zc)+c(zb,zd)), (c(xb,xc)+c(xa,xd)+c(zb,zc)+c(za,zd))), where c() is correlation.

3. If C > T (threshold), then accept, else reject

Figure 3.6 The algorithm for cycle detection

3.2 Deep Learning

3.2.1 Framework Selection

The very first step in analyzing the data was to determine which deep learning framework to use. The most mainstream frameworks available are Pytorch, Tensorflow, Keras, Caffe, and MATLAB. Some of the most important factors to consider when choosing a framework are (1) Community Support (2) Ease of Use (3) Academia and (4) Industry. Table 3.1 is the comparison of existing frameworks based on the criteria above.

Criteria	Winner	Runner up
Community Support	Tensorflow	Pytorch
Ease of Use	Pytorch	Tensorflow
Academia	Pytorch	Caffe
Industry	Tensorflow	Caffe

Table 3.1: Comparison of AI Frameworks [21]

After careful research, it came down to using either Pytorch or Tensorflow especially due to the prevalence of self-learning materials online. Another important deciding factor is the framework of the codebases that inspire this project, which is Tensorflow in this case. However, after trying out both Tensorflow and Pytorch, Pytorch was found to be more intuitive for a novice to get started and it was selected. Moreover, we were able to find a valuable resource that could guide us on Pytorch very effectively.

3.2.2 Implementation Overview

Because the datasets that we were using have already been preprocessed, the implementation of our project mostly consisted of implementing deep learning models. A significant effort in the implementation was spent on learning the best practices of deep learning. Then we implemented two types of deep learning architectures CNN, LSTM for gait identification. The details are elaborated under the implementation section (Section 4). To make the model generalize optimally, we also used regularization methods such as dropout with 20%. Then the performance of different architectures is compared under results (Section 5).

3.3 Experimental Hypothesis

A significant number of studies on gait identification were published before deep learning became mainstream, and relied on algorithms that are explicitly programmed. However, time-series data such as accelerometer and gyroscopes values are too complex for an average person to manually understand the patterns and formulate a hard-codable algorithm to analyze them. Therefore, we concluded that deep learning methods are the most effective approach to classify such complex data. After reviewing similar academic studies that utilized deep learning, we hypothesized that the accuracy for identifying people based on accelerometer and gyroscope could go up to around 90%.

4 Implementation

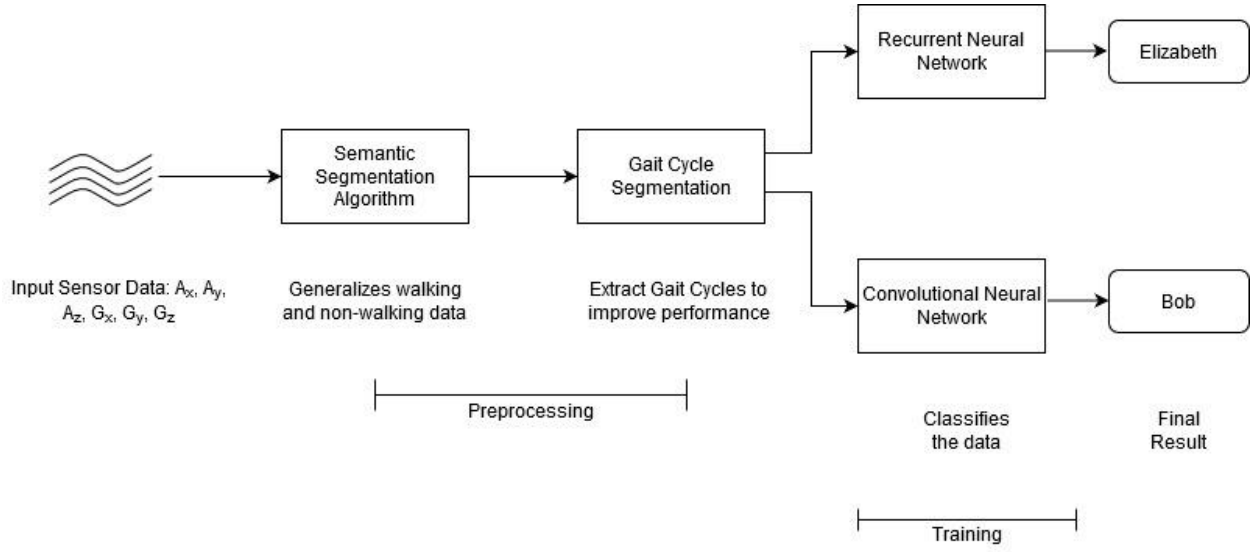

Figure 4.1: Flow diagram for preprocessing and training steps

As seen in figure 4.1, the implementation can mainly be broken down into pre-processing and training steps. The details of these steps are discussed below.

4.1 Data Preprocessing and Manipulation

As explained in Section 3.1, the data is preprocessed before feeding it into neural networks. However, the datasets that we are using have variations that are also preprocessed and manipulated. One dataset is based on step-segmentation similar to the one described in Section 3.1.1. Then each sample is interpolated into a fixed length of 128 data points using the linear interpolation operation. To increase the scale of the dataset, a one-step overlap between two neighboring samples is done for all subjects. In this way, the total number of 36,884 gait samples were collected for one dataset.

Another dataset was generated by dividing the gait curve into two-step samples and again interpolated into the same length of 128. The original dataset had 20 subjects with a much larger dataset than the first dataset resulting in 49,275 samples.

4.2 Datasets

There were two datasets utilized in this project generated by the methods above. Dataset 1 was collected on 118 subjects and contains 36,884 gait samples. The dataset is split 90:10 train to test ratio, with 33,104 used for training the deep learning model and 3,740 for testing it. Dataset 2 involves 20 subjects but it has larger gait samples than dataset 1. The total number of samples in dataset 2 is 49,275 which is split into 90:10 split such that 44,339 is used for training the model and 4,936 is used for testing it.

4.3 Neural Networks

4.3.1 Background and Justification of Choice of Algorithms

Due to the complexity of data, we decided to use deep neural networks to extract the characteristics of motion data. The two types of neural networks used to build models in this project are Convolutional Neural Networks (CNNs) briefly mentioned in the introduction section and the Long Short Term Memory (LSTM).

Deep Convolutional Neural Networks (DCNN) is a type of deep neural network that is often made up of several convolutional layers, activation layers (typically ReLU), pooling layers, and fully connected layers. CNN are well-known for their ability to extract features from images in recent years. Figure 4.1 depicts a possible architecture of CNN made up of the layers described above. It takes an image of 28 by 28 pixels and predicts the class of the image.

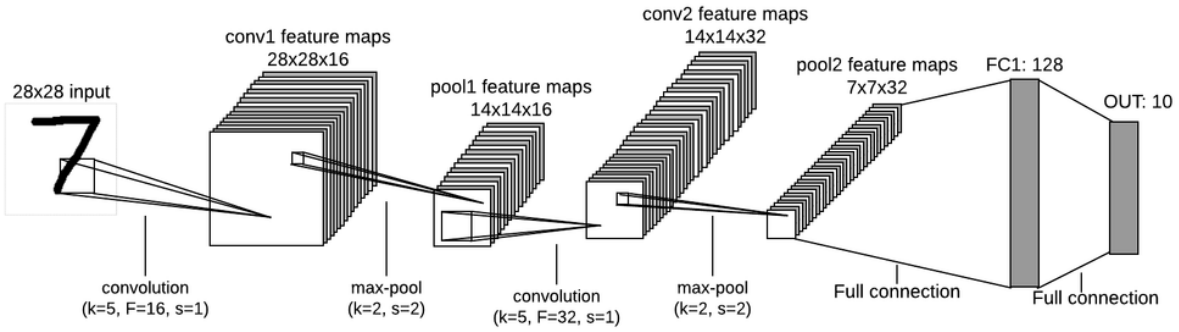

Figure 4.2: Architecture of Convolutional Neural Network [24].

Gait data is similar to image data in that they have multiple dimensions. For this reason, CNN can be used to extract the abstract characteristics of the gait data that are very challenging for humans to see. However, studies show that CNN that only processes single time-stamp data tends to ignore the temporality of time-series data that contains useful features for identification. Accelerometer and gyroscope data are such time-series data. [22]

Time-series analysis is where LSTMs comes into handy. Long Short Term Memory (LSTM) network is a variation of a Recurrent Neural Network (RNN). Unlike standard feedforward neural networks, LSTM works using feedback connections. The hidden layer of LSTM designed in a way that allows it to extract the features of time series data. A common LSTM cell consists of an input gate ($\mathbf{i}_t$), an output gate ($\mathbf{o}_t$) and a forget gate ($\mathbf{f}_t$). The three gates regulate the flow of information in and out of the cell. Figure 4.2 depicts the diagram of a single cell of LSTM. Note that $\mathbf{x}_t$ is the input data and $\mathbf{c}_t$ is a state vector)

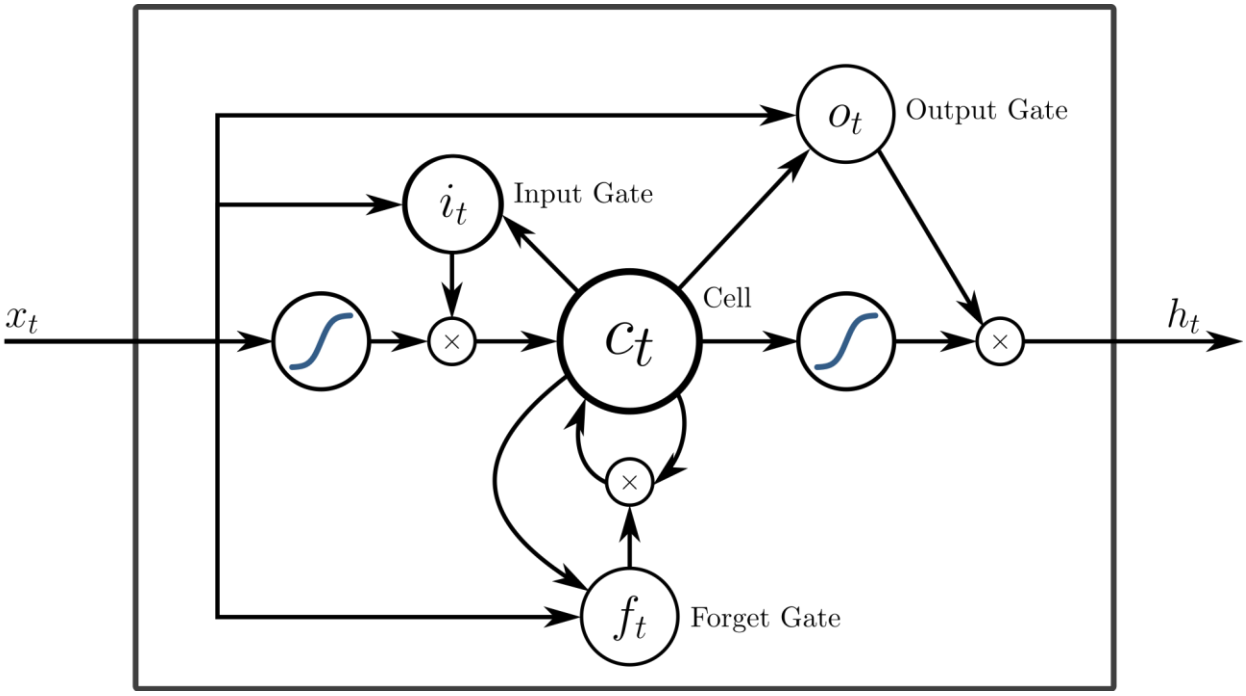

Figure 4.3: A single LSTM cell

The mathematical formulae of the cells and states are as follows.

$$\begin{aligned}
\mathbf{i}_t &= \sigma_i(W_{xi}x_t + W_{hi}^t\mathbf{h}_{t-1}^l + W_{hi}^l\mathbf{h}_t^{l-1} + W_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i),\\
\mathbf{f}_t &= \sigma_f(W_{xf}x_t + W_{hf}^t\mathbf{h}_{t-1}^l + W_{hf}^l\mathbf{h}_t^{l-1} + W_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f)\\
\mathbf{c}_t &= \mathbf{f}_t + \mathbf{c}_{t-1}\sigma_i(W_{xc}x_t + W_{hc}^t\mathbf{h}_{t-1}^l + W_{hc}^l\mathbf{h}_t^{l-1} + \mathbf{b}_c),\\
\mathbf{o}_t &= \sigma_o(W_{xo}x_t + W_{ho}^t\mathbf{h}_{t-1}^l + W_{ho}^l\mathbf{h}_t^{l-1} + W_{co}\mathbf{c}_{t-1} + \mathbf{b}_o),\\
\mathbf{h}_t^l &= \mathbf{o}_t\sigma_h(\mathbf{c}_t),
\end{aligned} \tag{1}$$

where **W, $\mathbf{i_t}$, $\mathbf{f_t}$, $\mathbf{c_t}$, $\mathbf{o_t}$, $\mathbf{h_t}$** are parameters of a particular layer l.

The Fully Connected layer of both LSTM and CNN is made up of a softmax activation function that produces a classification output that predicts the identity of the person who is walking.

$$\textbf{output} = \text{Softmax}(\mathbf{X} * \mathbf{W} + \mathbf{b}) \tag{2}$$

where X is the input, W is the weight matrix and b is the bias vector

The softmax activation function is the function that takes in a vector of K values and outputs a probability distribution of K possibilities.

$$\sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad (3)$$

Where z = X * W +b, e = Euler's number

For the Loss function, we decided to use Cross-Entropy Loss because it is widely used in classification models. Using cross-entropy function can speed up the training speed compared to using the variance cost function. The equation of cross-entropy is as follows.

$$\Lambda(o, o') = \Sigma^{N}(o_i' \ln o_i + (1 - o_i')\ln(1 - o_i)) \quad (3)$$

where **o** is the predicted output and **o'** is the expected output.

4.3.2 Architecture

We took an approach of testing different network architectures to classify gait patterns and tried to reproduce the results of the paper mentioned before. The three major approaches are CNN only, LSTM only, and CNN + LSTM. However, we did not finish implementing CNN + LSTM architecture. We will still be discussing the idea since we attempted.

CNN + LSTM network works by using both CNN and LSTM as feature extractors and feeding the concatenation of both outputs to the feedforward network with a softmax function. The fully connected layer works as a classifier and predicts the class of the input. Figure 4.4 is the diagram of the proposed architecture of the CNN and LSTM hybrid model.

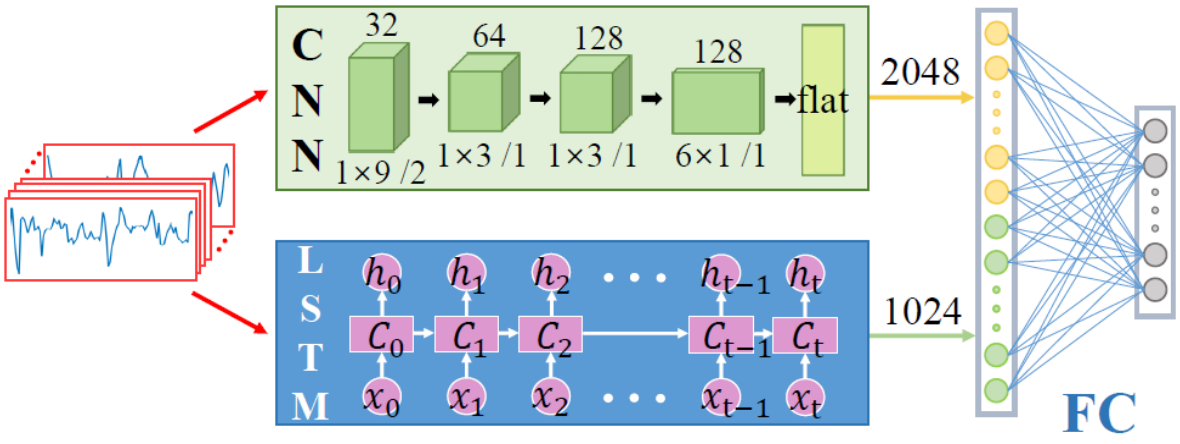

Figure 4.4: Architecture of CNN+LSTM model for gait Identification [6]

The CNN layer consists of 3 convolutional layers and two max-pooling layers. The details of the CNN structure are shown in Table 4.1.

Layer	Kernel Num	Kernel Size	Stride	Output Size
Conv 1	32	1x9	2	6x64x32
Max Pool 1	N/A	1x2	2	6x32x32
Conv 2_1	64	1x3	1	6x32x64
Conv 2_2	128	1x3	1	6x32x128
Max Pool 2	N/A	1x2	2	6x16x128
Conv 3	128	6x1	1	6x16x128

Table 4.1: The Details of CNN architecture

As for the standalone CNN or LSTM models we compared against, we used the same architecture that we use in the hybrid model. However, in this case, instead of having two different inputs concatenated together as one, we have just one input each that we feed into the fully connected layer.

5 Evaluation and Results

In this section, we will be discussing the evaluation of our results. As we discussed in the previous sections, we trained datasets on three different models. 1) A standalone CNN model, 2) a standalone LSTM model and 3) a hybrid model of CNN and LSTM. We will now be comparing and contrasting the results of these three different approaches. For testing we split roughly ten percent of both datasets and tested on those. To calculate the accuracy, we let the models classify each sample and divide the correctly classified samples by the total samples. This is done on both testing and training.

5.1 Standalone Convolutional Neural Network (CNN)

The standalone CNN's testing accuracy is close to 89% on Dataset 1 and 85% on Dataset 2 after splitting the data into 90:10 training to testing data and training for 60 epochs and the batch size of 32. In comparison, the original paper produced the result of 93% on Dataset 1 and 97% on Dataset 2. It's interesting to note that our implementation did worse on Dataset 2 than on Dataset 1 while the original team did it the other way around.

Another uncommon characteristic we found was the graph of accuracy against training loss. Because of the way the program was implemented, it's hard to visualize the loss and accuracy of each epoch in an intuitive. We decided to plot the best accuracy and loss of all the batches in each epoch and plot that against each epoch. Figure 5.1 is a plot of accuracy vs loss. The loss values are scaled down to be within 0 and 1 to make it more intuitive.

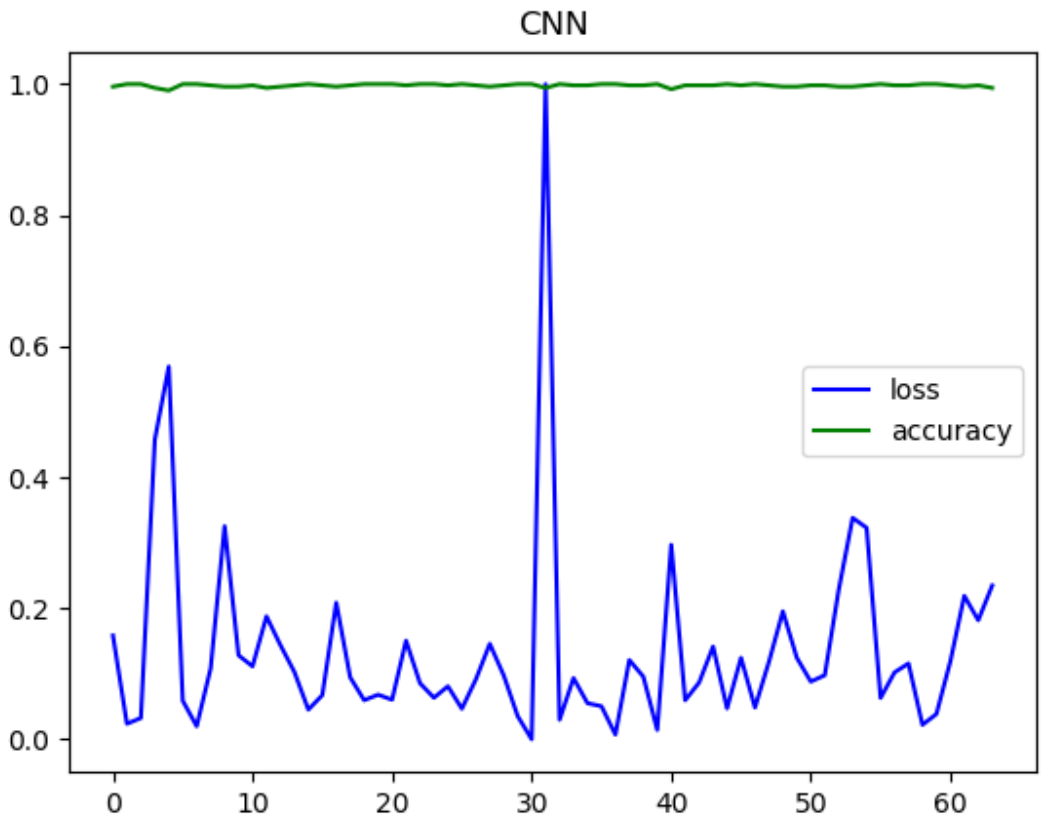

Figure 5.1: Loss vs Accuracy plot for CNN

We can see in figure 5.1 that the model has an accuracy of 100% since the early epochs. This is not an error but an implementation choice. We used 512 for batch size and took the best accuracy of all the batches during each epoch, which is showing an accuracy close to 100% and a low loss value. However, we will see that it's different in LSTM below where we did the implementation slightly differently.

5.2 Standalone Long Short Term Memory (LSTM)

The standalone LSTM's testing accuracy is close to 90% on Dataset 1 and 92% on Dataset 2 after splitting the data into 90:10 training to testing data and training for 50 epochs and

the batch size of 512. In comparison, the original paper produced the result of 92% on Dataset 1 and 97% on Dataset 2. Unlike the CNN approach, we reproduced the results pretty similar to the original paper.

The loss and accuracy correlation of the training process is a lot more intuitive than the plot above. Again, the loss values are scaled down to a value between 0 and 1 to make the graph more understandable.

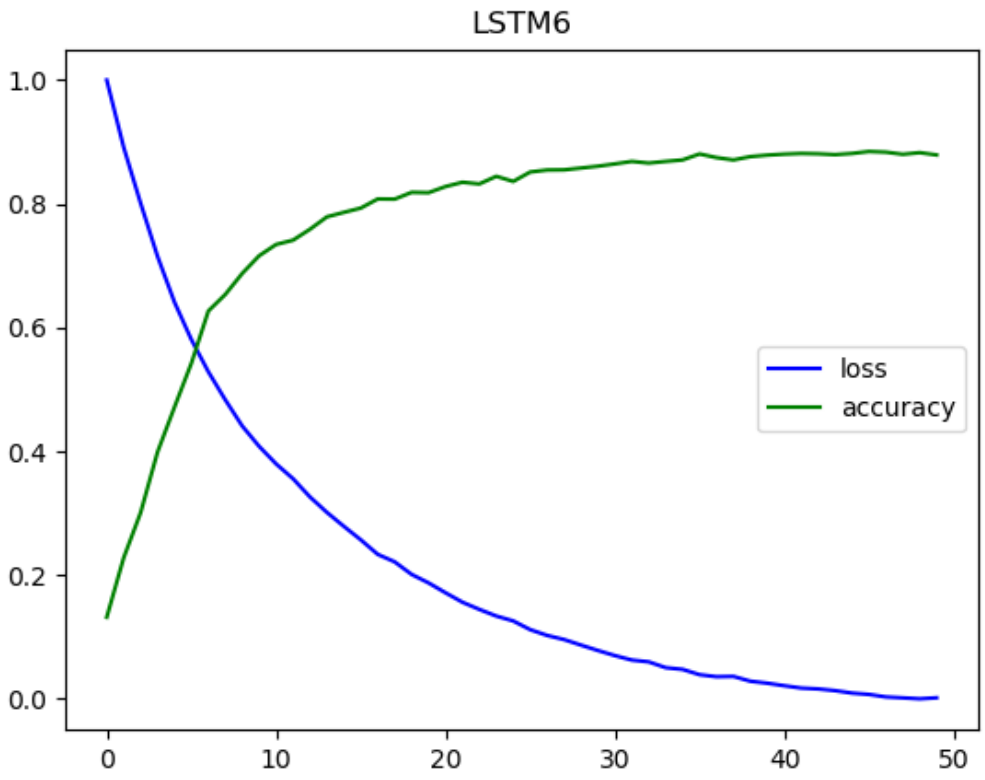

Figure 5.2: Loss vs Accuracy plot for LSTM

As we can note from Figure 5.2, the model learns very rapidly within the first 10 epochs and then continues to learn relatively slower until it plateaus at 40 epochs.

5.3 Comparison of results

In this section, we will be summarizing the results of the approaches and compare them against each other.

Deep Learning Method	Dataset 1 test accuracy	Dataset 2 test accuracy
CNN	89.0%	84.9%
LSTM	90.2%	92.1%

Table 5.1: Comparison of results between CNN and LSTM approaches

Table 5.1 compares the results achieved from two different approaches: CNN only and LSTM only models. This result is different from the results achieved in the original research in that the original implementations in that CNN outperformed LSTM for dataset 1 and the results are roughly the same for both CNN and LSTM for dataset 2.

6 Discussion

In this section, we summarize our main challenges and reflect on our experience of working on this project.

6.1 Limitations and Drawbacks

6.1.1 Deep Learning Related Limitations

Our deep learning models are trained to identify a fixed number of subjects involved in the dataset. For instance, the model can identify "Subject x" from n total subjects from the dataset. However, it cannot authenticate new subjects or users not included in the dataset. One way to solve this problem is to remove the n-1th layer from the trained model and retrain the model with the new identities on the output layer. Deep learning wise this is relatively easy to achieve provided that the new gait-related data can be gathered. However, from the aspect of prototyping the final product, there are more challenges involved with training the model over and over again live on the server. These aspects will be discussed under 6.1.3 Web application setbacks and 6.1.4 Mobile application setbacks. However, if given an extra amount of time, our team would have been able to solve these engineering problems.

6.1.2 Sensor Data Related Limitations

It has been observed by some research that gait analysis results depend highly on the data and the devices used to collect those data[6][8]. For example, the movement data collected on an MR100 sensor is different from those collected using sensors on a smartphone.

Even data collected on different models of smartphones can make a difference. This means that if a model is trained on data collected on device-1, it might not perform as well on

device-2. However, this is just our hypothesis deduced from observations of prior research teams. We did not find any definitive studies dedicated to the impact of gait data on gait data analysis.

Originally, we had a vision of deploying our analysis models on a smartphone. Before anything else, we looked for different datasets collected on smartphones. The first dataset suggested by our advisor was WISDM dataset. However, WISDM dataset only contains accelerometer values. This is a minor setback as a lot of prior research showed that gyroscope values are also important in gait analysis[6][7][10]. Soon after, we found a dataset that contained both accelerometer and gyroscope values as well as is collected using smartphones in an uncontrolled situation. So, we decided to move forward with the latter dataset. This dataset was collected on Samsung, Xiaomi, and Huawei but the specific models are not specified.

To reduce the error margin, it's optimal to find or even collect our data using the device that the prototype will be using. However, due to the limitation of manpower and time, we decided to train our models on the preexisting data. While this is not a major set back, training on-device data has a potential performance boost to our prototype.

6.1.3 Web Application Related Limitations

We implemented an application server and a web server that was intended to serve our native application as an API server [9]. While Flask serves as a server to receive the incoming gait data and route it to where the machine learning model exists, the results are, as of now, impractical to test because the model can only classify subjects from the dataset. This has been discussed above in the deep learning related limitations. To make the prototype testing plausible, the model needs to successfully accept new identities and retrain based on those. This

requirement comes with a different engineering challenge: to train the model on our server. This is a task suitable for future continuations of the research.

6.1.4 Mobile Device Resource Limitations

In our goal, the role of the mobile application is to record new sensor data, interface the users, and return results from classifying new data. However, in today's situation, the majority of mobile devices are not capable of training deep learning models natively. This fact deters us from developing a native application that is self-sufficient since our models must get retrained. This is the sole reason why we needed a web server: to let a more capable device handle heavier computations.

7 Conclusions and Future Work

7.1 Conclusions

With the rise in demands of personal identification and authentication, biometric systems have gathered a lot of attention from developers and researchers. Among the commercially available biometric solutions, many are considered to be obtrusive as they require users to participate more actively such as requiring users to physically place their fingers on the sensor. To provide a better unobtrusive solution, we attempted to build a system that can identify a person using their gait patterns. To do so, we analyzed the accelerometer and gyroscope sensor values from users' smartphones collected in a mildly controlled manner.

We did a lot of research on academic papers on gait analysis that are promising to our use case. Afterward, we decided to replicate the models of Deep Learning-based Gait Recognition Using Smartphones in the Wild[6]. Following their concept, we implemented two different deep learning models to train on gait data. We achieved results slightly lower but comparable to those of the original implementations.

7.2 Future Work

Our gait recognition project can be significantly improved in many ways. Because of time constraints, there are features left behind to develop. Mobile phones and smartwatches are improving rapidly and it's likely that shortly, they will be able to handle all server work related to the project. However currently, there's no such device and hence on the following section, deployment on the webserver will be discussed instead. There are also aspects related to model and data collection that can be improved further upon. All of these will be discussed as follows.

7.2.1 Model Modification

As elaborated in section 6.2.1 [Deep Learning Related Limitations], the major obstacle preventing the deployment of the model for live use is how the model is trained on. To recite the problem briefly, the model is trained to recognize 'n' people in the dataset and so far it can recognize only those 'n' identities. In other words, the model can only be used by subjects in the two datasets involved in our research. The improvement is to successfully register new users to the model so that the system can be scaled to all users.

This problem can be compared to facial recognition systems. This comparison can help find ways to solve the problem because the same problem for facial recognition has been solved. The new models for gait recognition can be built upon the existing models because the models are trained to understand the general patterns such as what a step means in terms of accelerometer and gyroscope values. This is similar to facial recognition systems in that deep learning models trained on faces learn less significant features such as eyes and noses. Therefore, if training is necessary to make this work, the last layer $(n-1)^{th}$ from the fully connected layers of both CNN and RNN could be removed and replaced with a more suitable layer for a specific usage.

By using the pre-trained models, the training for new users would be very quick compared to training from scratch. Removing the last layer will essentially remove the part which correlates walking patterns to n-users in the dataset.

7.2.2 Deploying the Model to Production

The explored pipeline to model deployment is to create a RESTful API on a Flask web server that will send a sequence of accelerometer and gyroscope data to a route where the pre-trained model is deployed on (e.g. *API/predict*). Afterward, the input is converted to a data

loader and then fed into the model to get a prediction. However, as mentioned before, this can identify only the n-subjects from the dataset.

Once the model is modified to register new users, the model might need to retrain on the live server on the go to consider the new users. The model is no longer just a pre-trained model and it has to evolve to accommodate registering new data. In other words, the model needs to do live training. This is another part a lot of effort can be put into improving the research.

7.2.3 Live Data Collections using Ubiquitous devices

A good candidate for collecting data and interfacing users is a mobile phone. Since the current model is trained on data collected on mobile phones, it would perform the best on data collected on mobile phones. If possible, it's optimal to use the exact mobile phone used in the original research. The engineering aspect of collecting data from a mobile phone is fairly straightforward.

We would like to propose two ways to collect data for future continuations. The first way is to prompt the user to click "Start collecting" and collect a sequence of data for a certain time. This chunk of data will be sent to the webserver via a post request and this will be fed to the pre-trained model to get predictions. The user will have to click the prompt and walk every time they want a prediction. This is a more straightforward way to make an application for testing.

The second way is to collect the data seamlessly as long as the application is on. This stream of data will continuously be sent to the server where the developer can decide how to divide and feed into the server to get a classification. This approach is more desired as it is closer to the actual real-world usage. However, compared to approach one, the division of data can be more complicated and the development aspect deserves better thoughts.

Smartphones are not the only candidates to face the user. Another possible target device is a smartwatch, many of which also have accelerometer and gyroscopes for data collection and can be programmed. Potentially, using smartwatches over smartphones come with some advantages. Smartphones are often not carried on the body of the owner while the smartwatches are worn most of the time. This would allow the program to monitor the users more and potentially make better decisions. However, to deploy the deep learning models on smartwatches, it should be trained on watches' sensor data instead. This is because watch sensor data is more likely to be impacted by the hand-wrist movement while smartphone sensor data is more likely to be influenced by lower-body movements.

References

[1] Laura Silver, Pew Research Center, Smartphone Owner is Growing Rapidly Around the World but not Always Equally https://www.pewresearch.org/global/2019/02/05/smartphone-ownership-is-growing-rapidly-around-the-world-but-not-always-equally/

[2] Zachary Arnold, Danielle Larose, WPI, Smartphone Gait Inference https://web.wpi.edu/Pubs/E-project/Available/E-project-043015-100131/

[3] Mohammad Omar Derawi, Norwegian Information Security Lab. Gjovik University College, Norway. *Accelerometer-Based Gait Analysis, A survey* NISK-2010.

[4] Heikki Ailisto, Mikko Lindholm, Jani Mäntyjärvi, Elena Vildjiounaite, and Satu-Marja Mäkelä, VTT Electronics, *Identifying People from Gait Pattern with Accelerometers*

[5] Kjetil Holien, Gjøvik University College, *Gait recognition under non-standard circumstances*

[6] Qin Zou, Yanling Wang, Qian Wang, Yi Zhao, Qingquan Li, *Deep Learning-Based Gait Recognition Using Smartphones in the Wild*

[7] R. E. Mayagoitia, A. V. Nene, and P. H. Veltink, "*Accelerometer and rate gyroscope measurement of kinematics: an inexpensive alternative to optical motion analysis systems,*" Journal of biomechanics

[8] W. Yuan and L. Zhang, "*Gait classification and identity authentication using CNN*," in Asian Simulation Conference, 2018

[9] Elizabeth Sahakyan, "*Create an API to deploy Machine Learning Models Using Flask and Heroku",* https://towardsdatascience.com/create-an-api-to-deploy-machine-learning-models-using-flask-and-heroku-67a011800c50

[10] E. P. Ijjina and C. K. Mohan, "*One-shot periodic activity recognition using convolutional neural networks*," in ICMLA, 2014

[11] Bouchrika I., Nixon M.S. (2007) "*Model-Based Feature Extraction for Gait Analysis and Recognition. In: Gagalowicz A., Philips W. (eds) Computer Vision/Computer Graphics Collaboration Techniques"*. MIRAGE 2007. Lecture Notes in Computer Science, vol 4418. Springer, Berlin, Heidelberg

[12] "https://www.bankmycell.com/blog/how-many-phones-are-in-the-world" As seen on Feb 26th, 2020

[13] "https://tractica.omdia.com/newsroom/press-releases/wearable-technology-market-is-quickly-diversifying-into-new-device-categories-application-markets-and-services/" As seen on Feb 26th, 2020

[14] Georgiou, Theodoros. (2018). "*Rhythmic Haptic Cueing for Gait Rehabilitation of Hemiparetic Stroke and Brain Injury Survivors.*"

[15] "https://menafn.com/1099294105/Wearable-Technology-Market-2019-Growth-Analysis-Future-Trends-Growth-Factors-Emerging-Technology-Sales-Revenue-Historical-Demands-Sales-by-Forecast-to-2022" As seen on Feb 26th, 2020

[16] Sun Bing, Wang Yang, Banda Jacob, "*Gait Characteristics Analysis and Identification Based on the iPhone's Accelerometer and Gyroscope"*

[17] "https://en.wikipedia.org/wiki/Machine_learning" As seen on Feb 26th, 2020

[18] "https://www.edureka.co/blog/what-is-machine-learning/" As seen on Feb 26th, 2020

[19] Bremner, Cheung, Lam & Huang, Worcester Polytechnic Institute "*Intoxigait Deep Learning*" 2018

[20] What is Moving Average "https://www.fidelity.com/learning-center/trading-investing/technical-analysis/technical-indicator-guide/wma" As seen on Feb 26th, 2020

[21] "https://www.analyticsvidhya.com/blog/2019/03/deep-learning-frameworks-comparison" As seen on Feb 27th, 2020

[22] "https://machinelearningmastery.com/how-to-model-human-activity-from-smartphone-data/" As seen on Feb 27th, 2020

[23] “https://towardsdatascience.com/why-go-large-with-data-for-deep-learning-12eee16f708” As seen on Apr 1st, 2020

[24] “https://www.easy-tensorflow.com/tf-tutorials/convolutional-neural-nets-cnns/cnn1?view=article&id=108:cnn”

[25] “https://www.google.com/search?client=firefox-b-1-d&q=stride” As seen on May 9th, 2020