DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING, SECOND CYCLE, 30 CREDITS

STOCKHOLM, SWEDEN 2020

Gait verification using deep learning models, accelerometers and gyroscope data.

ERIFILI ICHTIAROGLOU

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Gait verification using deep learning models, accelerometers and gyroscope data

Erifili Ichtiaroglou

Master of Science Thesis

Machine Learning
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology
Stockholm, Sweden

- January 2020

Examiner: Professor Hedvig Kjellström
Supervisor: Associate professor Pawel Herman

Abstract

Biometrical authentication systems, with its advantages over the traditional methods such as tokens and cards combined with secret passwords and pins, are becoming popular these days. In addition recent advances in machine learning helped improve the performance of biometrical systems, that can now be used commercially. The scope of the present thesis work is to investigate what is the minimum sample length for users' authentication based on monitoring of walking manner (gait).

In the context of the thesis three deep learning models of different architectures were built and trained using 6 dimensional inertial- sensor data, to assess the accuracy and the reliability of the proposed methods. The models were trained using 1, 2, 4 and 8 step gait patterns. The results showed that the most accurate model was the CNN+LSTM for short gaits, while for longer gaits the performance of the models seemed to converge. At the same time all the models performed better for longer gaits, with relatively small differences.

Sammanfattning

Biometriska autentiseringssystem, med dess fördelar jämfört med de traditionella metoderna som poletter och kort i kombination med hemliga lösenord och pinkoder, blir mer och mer populära i dag. Dessutom har de senaste framstegen inom maskininlärning bidragit till att förbättra prestandan för biometriska system, som nu kan användas kommersiellt. Räckvidden för detta examensarbete är att undersöka vad som är den minsta sample-längden för användar-autentisering baserat på övervakning av gångstil (gait).

I samband med examensarbetet byggdes och tränades tre deep learning-modeller med olika arkitekturer för 6-dimensionella tröghetssensor-data för att bedöma precisionen och tillförlitligheten hos de föreslagna metoderna. Modellerna tränades på 1, 2, 4 och 8 stegs gång-sampel. Resultaten visade att den mest exakta modellen var CNN + LSTM för kortare gång-sampel, medan modellerna för längre gång såg ut att konvergera. Samtidigt presterade alla modeller bättre för längre gång-sampel, med relativt små skillnader sinsemellan.

Acknowledgements

I would like to thank my supervisor at KTH Pawel Herman for his insights and guidelines during this project and my supervisors at Assa Abloy Ryd Gustav and Fredrik Einberg for their patience and guidance. I would like to express my gratitude to all, as they supported and motivated me by helping me overcome any problem in the way.

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

MEM	Microelectromechanical (systems)
HAR	Human Activity Recognition
DNN	Deep Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Networks
LSTM	Long Short-Term Memory
PCA	Principal Component Analysis
ANN	Artificial Neural Network
ANOVA	Analyis of Variance
DET	Detection Error Trade off

Chapter 1

Introduction

In our rapidly increasing networked society, security issues are becoming more and more important. Most of us nowadays, have a number of digital accounts, for which we need to reliably authenticate ourselves. Until recently, the most popular methods for managing authentication systems required users to own tokens and cards, or remember secret passwords and pins or often combine both [1]. As a result, users ended up having in their possession a large number of tokens and passwords, each one fitting for a specific task e.g. access control in buildings, email checking, e-banking logins, money transactions etc. This can be overwhelming for many users, who often repeat their authentication credentials (passwords, secret phrases), in an attempt to not forget them, reducing eventually the levels of security [2].

However, the danger of forgetting the passwords is not the only deficiency of the aforementioned traditional methods. Cards and tokens can be stolen or lost, while passwords and pins can be copied and sent out and thus, they can be used by anyone even if the actual owner is not present [3]. To overcome all of these problems, biometrics systems are employed nowadays, and becoming popular due to their liability and their efficiency [3]. According to Liu and Silverman [4] the term biometrics, refers to the science of statistical analysis of biological characteristics. Biometrics is thus used for identity authentication based on physiological or behavioral characteristics. Common biometric modalities include fingerprints, palm geometry, retina, iris, facial characteristics, signature, voice and gait [4].

In contrast with the traditional methods biometrics cannot be stolen, forgotten or borrowed and require that the user is present during the authentication procedure. They are very easy to use and handle, and it is almost impossible to forge them. Due to these properties, a lot of biometrics have been studied extensively and used commercially with great success, up until today, e.g. iris, face, fingerprints [5]. Recent advances in deep learning and wide spread

applications of machine learning promise even higher accuracies which might be able to ensure in the future higher levels of security and more personalized authentication procedures.

1.1 Problem description and motivation

As it has already been mentioned, biometrics are gaining more ground compared to traditional authentication techniques, for the aforementioned advantages. However, even among the biometric modalities, some appear to have more benefits than others[5]. This is the reason why gait biometric is becoming more and more popular. This modality refers to users' identification or authentication based on their walking manner and it appears to be unobtrusive. Some biometrics require that the users place their fingers on a certain device to capture their fingerprints or look persistently into a camera for their face or eyes features to be recorded [5]. In these cases, the users might find the procedure inconvenient or annoying and may also feel that their privacy is being invaded. On the other hand, the authentication based on the gait patterns requires only that the users walk casually without being asked to perform any additional task. Furthermore, it is hard to impersonate gait features, which we can also capture while being far away for the users, in contrast with other biometric modalities [6].

Authentication based on gait biometric can be performed by analyzing two different types of data [7]. Video analysis of human motion has been the most popular approach [7]. However, the extensive usage of smartphones equipped with sensors (accelerometers and gyroscopes) that capture motion patterns offered another possibility less expensive and less intrusive [6]. Data collection is simplified significantly and users privacy is not invaded by cameras, which record their movements, face and body, possibly without their consent.

In the context of this project the inertial sensor approach will be followed. Accelerometer and gyroscope data will be used to train neural network models in order to address the gait authentication task. However, for the gait biometric to be exploited commercially, it is necessary that the authentication procedure is not computationally heavy and that it is as accurate as possible. Building a system that requires a lot of computational power or memory is not appropriate for real-time scenarios like the gait authentication task. In addition, it is quite important to identify what is the appropriate period of time that we need to record the users' walking pattern to confidently authenticate them. A system that would need, for instance, even one minute of a walking pattern to perform the authentication is of no commercial value. Furthermore, if the users are asked to walk for a long period of time in order to gather the needed data the gait biometric is stripped of its unobtrusive characteristic.

1.2 Research question

The purpose of this thesis is to examine the suitability of deep learning approaches to gait authentication task. In particular, it is of interest to investigate the minimum recording time of gait pattern needed to successfully authenticate the identity of the users. The reason for choosing deep learning approaches is the fact that this area of machine learning has demonstrated remarkable performances, in other complex pattern recognition problems like visual perception and speech recognition [5].

Specifically we will focus on two types of neural networks: long short term memory networks (LSTMs) and convolutional neural networks (CNNs). LSTMs are suitable for processing time series data in a recursive way, like voice and gait[5]. CNNs convole the input data in space domain and are suitable in handling data arranged in 2-dimensional grids, such as images [5]. Both networks can promise high performances in their domains and since inertial sensor data can be arranged in 2-dimensional time series, we decided to explore how deep learning models can tackle the gair verification task.

The research questions can be formulated as follows:

1. What is the minimum period of time needed for gait monitoring to reliably authenticate the user's identity?

2. How does it depend on the pattern recognition method in use?

To answer the aforementioned research questions we explicitly set the goals of the thesis as follows:

- Finding the minimum period of time, that we need to monitor the users' gait, which at the same time will guarantee an accurate prediction.

- Finding a suitable neural network model that will predict users' identity with high accuracy. The model should be able to generalize to users that were not used for training and distinguish among them.

1.3 Thesis outline

The chapters of this report are organised as follows:

- Chapter 2, *Background*, provides knowledge about the type of data used for the purpose of the present thesis and the background theory of deep neural networks, giving emphasis on the 2 types of neural networks that were implemented. In addition, previous research work related to the topic of the thesis is presented.

- In chapter 3, *Methods*, information is provided about the data collection, preparation and datasets formulation. In this chapter the architecture of the models chosen is also presented in detail.
- The following chapter 4, *Experiments and evaluation* presents the results of all the experiments that were conducted.
- In chapter 5 *Discussion*, the experimental results are discussed and some ethical and societal aspects of the project are presented.
- Finally, the last chapter 6, *Conclusion and future work* presents a short conclusion on the results and suggests some ideas for future work.

Chapter 2

Background

This chapter will provide the reader with some useful background knowledge in order to make this work easily comprehensible. First, it will give an explanation of what the accelerometers and the gyroscopes are and how they are used. Later, we will briefly introduce the theory of neural networks and we will go through the architecture of some popular networks. Following we will present a summary of previous research, focusing on the task of gait identification.

2.1 Accelerometers and gyroscopes

As mentioned in Chapter 1 the data for this task were acquired using inertial sensors embedded in smartphones; specifically accelerometers and gyroscopes.

Accelerometers are sensors that can measure physical acceleration of an object, due to inertial or static forces, like the constant force of gravity, or dynamic forces, caused by moving or vibrating the accelerometer. Nowadays the most commonly used accelerometers are the microelectromechanical (MEMs) accelerometers, due to their low cost, small size and robust sensing [8]. The functionality principle of the accelerometers can be explained by a single mass, that is attached to a spring. When a linear acceleration occurs, a force equal to mass times the acceleration causes the mass to deflect and stretch the spring. Subsequently, this stretching is converted by a suitable system to an equivalent electrical signal [9]. The use of more that one sensors can offer multi-axis acceleration detection, with the three-axis being widely used in smartphones. Some of the target applications of accelerometers in smartphones are dangerous driving detection, gesture recognition, screen rotation and phone hacking [10].

The gyroscope or gyro is a sensor that estimates or maintains the orientation based on the principles of angular momentum. It detects the angular velocity of an object, that indicates how fast it rotates around an axis. Traditional gyroscopes

consist of a disk in which the axle moves freely and assumes any orientation. When torque is applied on a spinning disk, due to an external force, the orientation of the gyro changes. However, providing a high spin rate of the disk guarantees a large angular momentum, that keeps the axle nearly fixed, regardless of the external torque [10]. Today in smartphones microelectromechanical (MEMs) gyroscopes are being used. They measure the Coriolis force of a rotating body in every axis, which is proportional to the angular velocity [11]. In smartphone applications, where an accurate estimation of the direction is needed, gyroscopes are employed. A calculation based only on accelerometers is not precise, due to jerky movements that might occur to the phone [11]. For this reason, in most of the applications mentioned before, both accelerometers and gyroscopes are being exploited.

2.2 Deep learning

The idea behind deep learning was to tackle problems, that are easy for people to perform, but difficult to describe in a formal way, e.g. distinguishing a car from an animal [12]. For these intuitive problems, deep learning perspective suggests that computers would learn from experience, i.e. data, in a hierarchical manner; starting from learning simple patterns and going towards more complicated ones, as humans do.

Traditional machine learning models are based on solving a convex optimisation problem with strong convergence guarantees. However, for some more complex learning tasks solving a simple convex problem does not provide accurate predictions [13]. In addition, extracting the appropriate features from raw data, to train machine learning models, is not always feasible. In many cases, it is difficult to know what are the most representative features to extract, for example in the task mentioned before (distinguishing a car from an animal) [12]. These problems were overcome by the use of *deep neural networks* (DNNs). DNNs consist of multiple layers of linear and nonlinear transformation in order to create high-level abstractions of the input data and map them to their corresponding output [13]. There exist many types of neural networks with different architectures, but the main principles are very similar. Multi-layer feed-forward neural networks were the first ever to be used and can explain the basics of deep learning.

2.2.1 Multi-layer feed-forward neural networks

A multi-layer feed-forward neural network, also known as fully connected networks is a function approximator that takes the form of a graph with subsets of nodes, neurons, arranged in a sequence (figure 2.1). Each subset of neurons is

called a layer. The first layer is called input and the last output layer. The layers in between are called hidden layers. The neurons of one layer are connected to all the neurons of the next layer [13]. Thus, between two subsequent layers there is a weight matrix of size $h_l \times h_{l-1}$ and a bias vector of size $h_l \times 1$, where h_l is the number of neurons in layer l. An input vector $\mathbf{x} = (x_1, ..., x_d)$ is fed into the network and after the first layer is transformed according to the equations 2.1 and

$$\alpha^1 = \mathbf{W}^1\mathbf{x} + \mathbf{b}^1 \tag{2.1}$$
$$\alpha_j^1 = \sum_{i=1}^{d} w_{ji}^1 x_i + b_j^1$$

where w_{ji}^1 is the weight connecting the neuron i of the input layer to the neuron j of the first hidden layer, b_j^1 is the bias and α_j^1 is the activation of j neuron in the first hidden layer. Each of the activations, α_j^1 is further transformed by a nonlinear, differentiable activation function h, that denotes the level of response of the neurons in that specific layer. The most widely used activation function are sigmoidal, such as *tanh* function.

$$z_j^1 = h(\alpha_j^1) \tag{2.2}$$

This procedure, is repeated through all the layers of the network to propagate forward the information, until we reach the output layer. The activations of the output layer are transformed by an activation function, that is specified by the nature of the data [14]. In case of regression problems the activation is the identity matrix, meaning that $\mathbf{y} = \alpha^L$, where L is the output layer. In case of classification problems, such as the gait verification task, that this project focuses on, the softmax function is used. As shown in equation 2.3 the softmax function turns the values of the output vector $\mathbf{y} = (y_1, ...y_K)$, into probabilities that sum up to 1. Thus, the output layer provides the probability distribution of a discrete variable with as many values as the number of the neurons in the last layer (classification labels).

$$softmax(y_i) = \frac{e^{\alpha_i^L}}{\sum_{j=1}^{K} e^{\alpha_j^L}} \tag{2.3}$$

It has been made clear that the multi-layer feed-forward neural network performs some mathematical operations one after the other, in order to map the input data into their corresponding output. Following in this chapter more DNNs will be discussed, with more complicated architectures, however they are also functions that are performed in a sequence, as in the case of MLP.

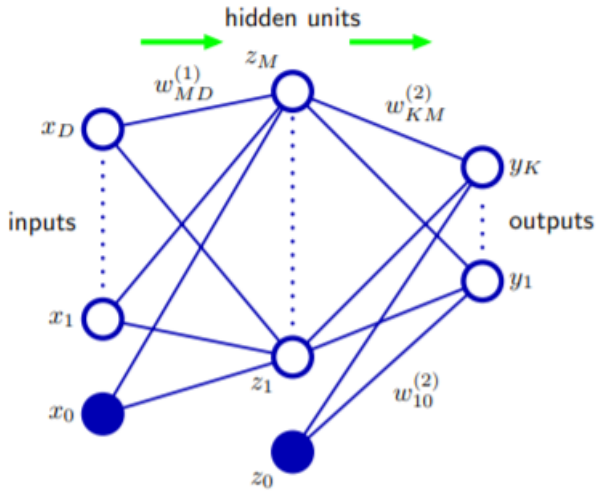


Figure 2.1: Fully connected neural network with 1 hidden layer [14]

2.2.2 Training a neural network

As it was mentioned before, a neural network is a function approximator. It tries to approach the true function f, that maps input values to output ones and is written as $y = f(x)$. Most of the tasks in our everyday life can be seen as such functions, e.g. mapping the face of a person to a name, or a melody to a song's name, or certain words to a specific language.

In the case of classification, neural networks try to approximate f, with another function $\hat{f}$, that performs the mapping $\mathrm{y} = \hat{f}(\mathrm{x};\theta)$. During the training of the network, the free parameters θ (weights W and biases b), need to be adjusted, to bring the predicted output values as close as possible to the real output. *Gradient descent* is the most widely used algorithm for training that tries to find the proper parameters θ, by optimizing an *objective function*.

Objective function

In supervised learning, such as the classification case, neural networks are trained using the *maximum likelihood* estimator, whose goal is to approximate a conditional probability $P(\mathbf{y}|\mathbf{x};\theta)$ of the output $\mathbf{y}$ given the data $\mathbf{x}$. The solution of the conditional maximum likelihood estimator is

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}}\, P(\mathbf{Y}|\mathbf{X};\theta) \tag{2.4}$$

where, $\mathbf{Y}$ represents all the real targets and $\mathbf{X}$ represents all the input data. If the data set consists of N independent, identically, distributed observations (i.i.d.),

then the equation 2.4 can be written as

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \prod_{i=1}^{N} P(y^{(i)}|x^{(i)};\theta) \tag{2.5}$$

and by taking the logarithm we obtain

$$\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} log(P(y^{(i)}|x^{(i)};\theta)) \tag{2.6}$$

Thus, the optimization function, also known as cost or loss function is the negative log-likelihood, which is equivalent to the cross- entropy between the training set and the model distribution (eq:2.8)[13].

$$\mathscr{E}(\theta) = -\mathbb{E}_{x,y\hat{P}_{data}} log P_{model}(\mathbf{y}|\mathbf{x};\theta) \tag{2.7}$$

$$= -\frac{1}{N}\sum_{i=1}^{N} log(P_{model}(y^{(i)}|x^{(i)};\theta)) \tag{2.8}$$

where $\hat{P}_{data}$ is the distribution of the training data and P_{model} is the estimated one. The specific form of the cost function is dependent on the model in use and the assumptions that underline it. For example, if we assume that $P_{model}(\mathbf{y}|\mathbf{x};\theta) = N(\mathbf{y}; f(\mathbf{x};\theta), \mathbf{I})$, then we end up having the mean squared error cost as our objective eq:2.9.

$$\mathscr{E}(\theta) = \frac{1}{2}\mathbb{E}_{x,y\hat{P}_{data}} ||\mathbf{y} - f(\mathbf{x};\theta)||^2 + const \tag{2.9}$$

Gradient descent

Our ultimate goal is to find the parameters θ, that will maximize the likelihood. However as shown in equations 2.2-2.8, maximizing the likelihood is equivalent with minimizing the objective function $\mathscr{E}(\theta)$. Gradient descent algorithm is used for this purpose. According to this algorithm the loss function is minimized by iteratively updating the parameters and moving in small steps towards the direction of the steepest descent, defined by the objective function's negative gradient (w.r.t. the parameters θ).

$$\theta_{\tau+1} \leftarrow \theta_{\tau} - \eta \nabla_{\theta} \mathscr{E}(\theta) \tag{2.10}$$

where τ denotes the iteration number, $\nabla_{\theta}\mathscr{E}(\theta)$ is the gradient of the objective function and η is the learning rate parameter, a scalar that defines the size of the

step. The learning rate is an important hyper parameter, which value should be chosen with caution. Too big learning rate means that we will make large steps and we might "jump over" the minimum, while too small learning rate means that it might take too much time to converge and possibly to some suboptimal local minima. Usually a large learning rate is chosen in the beginning of training and it gets smaller as we reach a local minimum.

Through the years, scientists have proposed some alterations of the gradient descent algorithm that promise faster convergence and more accurate results. One of the first techniques proposed is called *momentum*, that helps accelerate the gradients towards the right direction [15]. In contrast to the simple gradient descent, that updates the parameters according to the current value of the gradient, momentum gradient descent takes into to consideration the gradients of all previous update steps, iterations (equation 2.12).

$$\theta_{\tau+1} \leftarrow \theta_\tau - \eta V_\tau \tag{2.11}$$

$$V_\tau = \beta V_{\tau-1} + (1-\beta)\nabla_\theta \mathscr{E}(\theta) \tag{2.12}$$

where V_τ is the moving average of the the previous and current gradients an β is a coefficient that weights the importance of previous and current gradients.

Recently more optimization methods have been proposed that combine momentum with other more sophisticated techniques and further improved the existing algorithms. *AdaGrad* [16] and *RMSprop* both divide the learning rate by the square root of the sum of the previous and current gradients, while RMSprop uses weights to adjust the importance of the past and the current gradients. *Adam* [17] is an optimization algorithm that is commonly used, because it computes adaptive learning rate for each parameter, combining both the momentum and RMSprop methods.

However, to perform the updates of the parameters we need first to compute the gradient of the cost function with respect to them, $\nabla_\theta \mathscr{E}(\theta)$. *Backpropagation* [18] is an efficient algorithm, that is employed in order to permit the information of the cost to travel back into the network and compute the gradients. As it has been mentioned before, neural networks are a series of mathematical operations that are performed one after the other and lead to the computation of the loss. Thus, backpropagation needs to make use of the chain rule, in order to compute all the gradients from the last layer to the first. In deep neural networks with many layers and neurons, that would be time consuming and costly. However backpropagation instead of calculating the gradients of each layer separately, it saves and reuses some computations of the gradient of one layer, for the computations of the gradient of the previous layer.

Mini-batch training

It is important for neural networks to be trained on large datasets, in order to guarantee high performance. As shown before, in order to calculate the error function of a set of independent observations using the maximum likelihood, we need to compute the sum of individual costs for each observation.

$$\mathscr{E}(\theta) = \sum_{n=1}^{N} \mathscr{E}_n(\theta) \tag{2.13}$$

In order to make an update to the parameters, we need to calculate the above sum, which might be time consuming, considering a large scale of data . Thus, instead of using the whole dataset to make a single update to the weights, we compute the cost on a smaller subset of the samples, the *minibatch*. The initial dataset is split into chunks, or minibatches, used iteratively to update the parameters until all the data are used. This procedure is repeated as many times as needed to minimize the error. It would be useful to mention that an update of the weights over a minibatch is called a *learning iteration*, while the update over the whole dataset is called an *epoch*.

2.2.3 Convolutional neural networks, CNNs

As discussed above in regular fully connected networks the activation of each layer results from general matrix multiplication. In convolutional neural networks though, the activation occurs by employing the linear mathematical operation *convolution*, instead [12].

In mathematics and engineering convolution is defined as the integral of the product of two functions, when one of them is reversed and shifted 2.14. Convolution is used to express how the shape of one function is modified because of the other.

$$(x * w)(t) = \int x(\alpha) w(t-\alpha) d\alpha \tag{2.14}$$

or in the discrete case:

$$(x * w)(t) = \sum_{\alpha=-\infty}^{\infty} x(\alpha) w(t-\alpha) d\alpha \tag{2.15}$$

In the above equations 2.14, 2.15, x is considered to be the input function, w is considered to be the kernel function and the output of the convolution is often called feature map.

Convolution in machine learning

In machine learning the input and kernel functions are often multidimensional arrays of finite size of data and trainable parameters respectively. Usually the convolution is implemented in more than one dimensions at the same time. For example, in case of a 2-dimensional image I and a 2-dimensional kernel K the convolution would take the following form [12]:

$$(I * K)(i, j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.16}$$

If we make use of the commutative property of convolution we can write the equation 2.16 as:

$$(I * K)(i, j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{2.17}$$

Using the equation 2.17 is more convenient as the range of the indices m and n is shorter in this case (the kernel size is always smaller than the size of input array). However, in most machine learning libraries the convolution is implemented as the cross-correlation function [12]:

$$(I * K)(i, j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{2.18}$$

The figure 2.2 shows in a graphical way how the convolution is implemented for a two-dimensional input image of size 10x10 and a kernel of size 3x3. The hidden neuron, in figure 2.2, is not connected to all the neurons of the input image (as in the case of fully connected networks), but rather to a small square region of the image. This kind of interactions, also referred to as sparse connectivity, is the main characteristic of the CNNs [12]. It allows the network to capture small and meaningful spatial or temporal features of the input data. Following, as figure 2.3 shows, the same kernel is slid over the input image according to a specific step size (stride). In this way sparse connectivity decreases also the number of parameters and thus the storage requirements of the model.

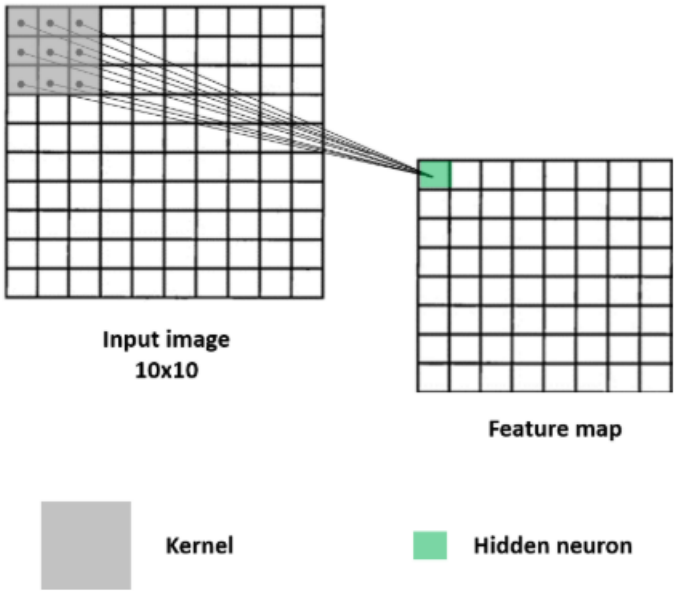


Figure 2.2: Convolution between 3x3 kernel and 10x10 input image [19]

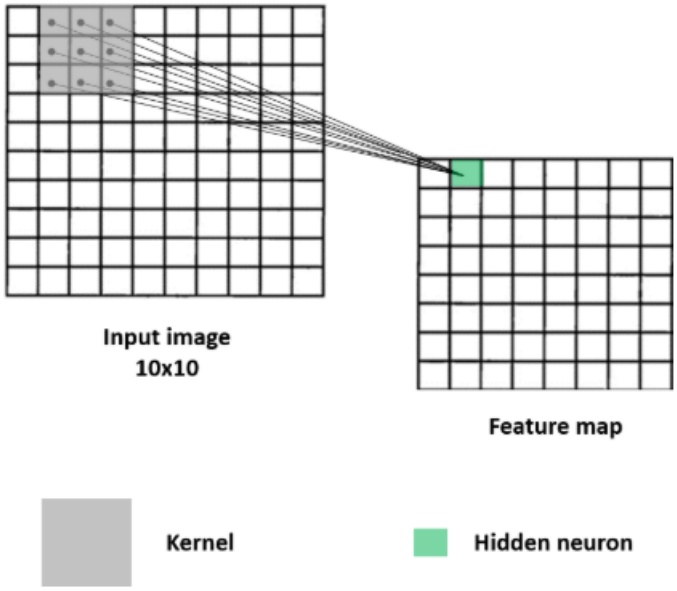


Figure 2.3: The kernel has shifted by 1 unit. [19]

Once the feature map is extracted, it is then fed into a rectified linear unit (ReLU) layer. The ReLU activation function is presented in the figure 2.4. This

layer introduces non-linearity to the model and it is preferred over other activation functions, like sigmoid, as it speeds up the training process by eliminating the vanishing gradient problem. The last stage of a typical CNN includes the pooling layer. This layer replaces a certain output of the feature map according to the value occurred by some statistical operations on the nearby outputs [12]. These operations might include computing the average of the neighboring outputs, or finding the maximum among them. The purpose of this layer is to make the network invariant to small changes in the input, as well as further decrease the number of parameters[12].

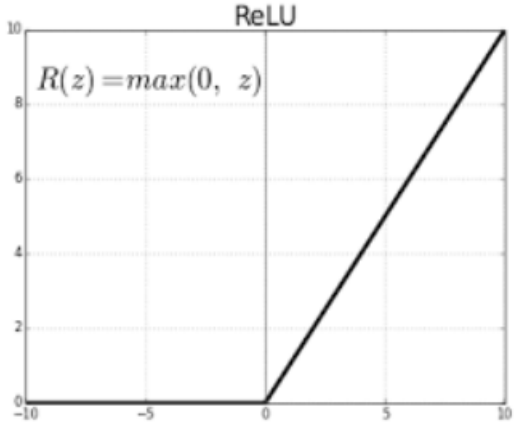


Figure 2.4: Graphical representation of ReLU. [19]

2.2.4 Recurrent neural networks, RNNs

In contrast with CNNs that were initially developed to process grids of data such as images; recurrent neural networks, RNNs, were developed to process sequential data, such as time series, music or video and natural languages. RNNs in principle take into consideration the entire history of all previous inputs to compute each output. In other words, each output is a function of all previous outputs.

To achieve this RNNs allow cyclical connections among the nodes of the network, in order to create a "memory" of inputs, of previous time steps, that flows through the network and thus influence the output [20]. These cyclical connections are achieved by feeding the activation of a node at the previous time step to the same node at the current time step. Thus, RNNs, share parameters across different time steps, by feeding new activations to a network with the same parameters that are kept throughout the sequence, as figure **??** illustrates. The equations that describes the RNN can be written as:

$$\alpha_t = U \times X_t + W \times H_{t-1} + bt \tag{2.19}$$

$$H_t = tanh(\alpha_t) \tag{2.20}$$

where H_t is the activation of the hidden node. The output of the recurrent network can be either a sequence, as in the figure **??**, or a single value according to the task, that the RNN is built for. For instance if the task is to classification of sequential data then the output can be a single value, however in case of translation tasks the output should be a sequence.

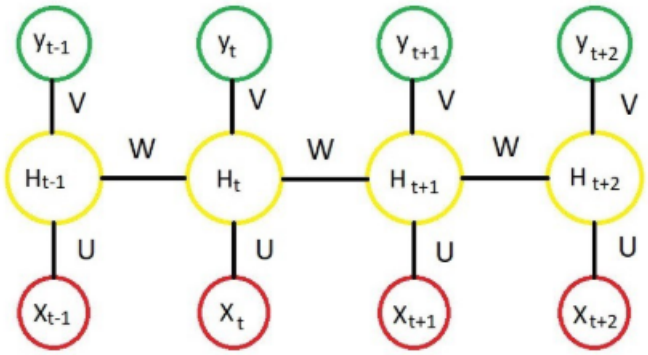


Figure 2.5: Unfolded computational graph of RNN. W, U and V are respectively the hidden-to-hidden, input-to-hidden and hidden-to-output matrices of the network that remain the same throughout the chain [21].

Difficulty in learning long-term dependencies

As helpful as the cyclical connectivity of the network is, it has a quite important disadvantage, which is the problem of vanishing or exploding gradients, when trying to learn long-term dependencies. During back propagation, the computation of the gradients depends exponentially on the number on time steps that are part of the procedure. As the number of time-steps increases the computation of the gradients involves the multiplication of an increasing number of Jacobians, that may lead the gradients to vanish or explode.

Consider we want to compute the gradient of the loss function with respect to the W weight matrix, as shown in figure 2.5. Using the back propagation through time algorithm, BPTT, and taking into consideration that the overall loss is a summation of the losses at every time step, we obtain the equations [22]:

$$\frac{\partial \varepsilon}{\partial W} = \sum_{t=1}^{T} \frac{\partial \varepsilon_t}{\partial W} \tag{2.21}$$

$$\frac{\partial \varepsilon_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial \varepsilon_t}{\partial H_t} \frac{\partial H_t}{\partial H_k} \frac{\partial H_k}{\partial W} \tag{2.22}$$

$$\frac{\partial H_t}{\partial H_k} = \prod_{i>k}^{t} \frac{\partial H_i}{\partial H_i - 1} = W^{t-k} \prod_{i>k}^{t} diag(tanh'(H_{i-1})) \tag{2.23}$$

It is clear that in order to compute the gradient of the loss for the specific time step t, we need to compute the product of equation 2.23. It is clear that for k«t the multiplication of $t-k$ Jacobians can cause the explosion or the vanishing of the gradients, according to their values. It is for that reason that the RNN cannot learn long-term dependencies.

Long short term memory networks, LSTM

The long short term memory networks were introduced as a new approach over the RNNs, in order to overcome the long-term dependencies problem. The idea was to create paths deep through time with gradients that do not vanish or explode.

LSTM networks are similar to RNNs, but in addition they have internal recurrence, self-loops, that makes the weights conditioned to the context of the current time step [12]. This internal reccurence is controlled by a hidden unit called LSTM cell. Thus, the LSTM cell will decide which information to forget, which to take in and which to output.

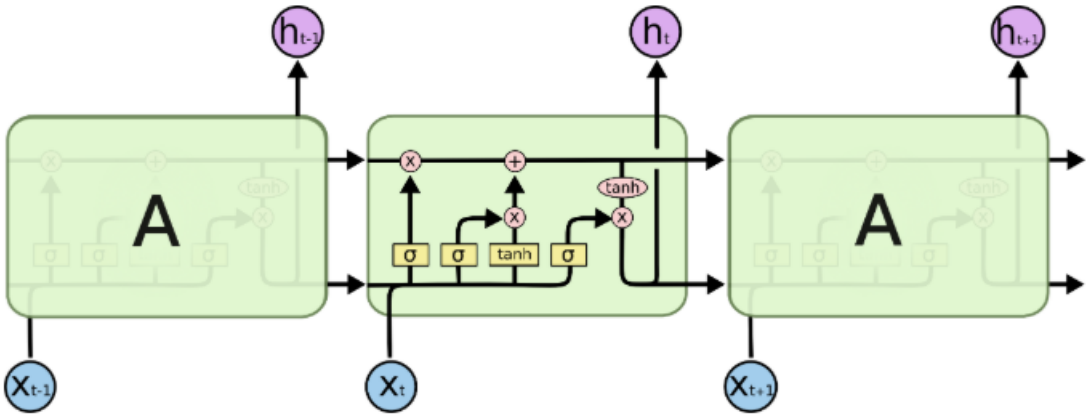


Figure 2.6: Unfolded computational graph of LSTM. [23].

The structure of the LSTM cell is shown in figure 2.6 The pink circles indicate point-wise operations, such as vector multiplication and addition, and the yellow boxes neural network layers with different activation functions.

The cell state is the horizontal line than runs through the top of the diagram and keeps the information flowing through the chain. The cell state represents the memory of the network that updates or forgets the input information of the cell according to three internal gates: the forget the input and the output gate. As figure 2.6 illustrates the inputs to the cell are the hidden node of the previous time step h_{t-1} and the input at the current time step x_t. The combination of the inputs is regulated through the activations and the multiplications of the gates in order to update the cell state only with relevant information. After the cell state is updated the LSTM uses the nesw information stored in the cell state to compute the output of the current time step. Thus the LSTM prevents the gradient problems that occurs by irrelevant inputs and outputs [24]. The equations that describe the LSTM are the following:

$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f) \tag{2.24}$$

$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i) \tag{2.25}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot tanh(W_c h_{t-1} + U_c x_t + b_c) \tag{2.26}$$

$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o) \tag{2.27}$$

$$h_t = o_t \cdot tanh(c_t) \tag{2.28}$$

where f_t, i_t, o_t are the forget, input and output gates respectively and c_t and h_t are the cell and hidden state for the current time step. The matrices denoted with W are the hidden-to-hidden matrices, those denoted with U are the input-to-hidden matrices and b are the bias vectors of the gates.

2.2.5 Siamese neural networks

A commonly used approach for any verification task is the implementation of siamese neural networks, that were introduced in 90s by Bromley et. all [25]. Siamese neural networks are used to compare the similarity between two different input data. Siamese neural networks consist of two separate, but identical sub-networks that perform a non-linear mapping from the input domain and

extract high level representations of the input data, figure 2.7. The extracted representations are then compared and the network learns using loss functions based on similarity method, such as cosine similarity or euclidean distance [25], [26].

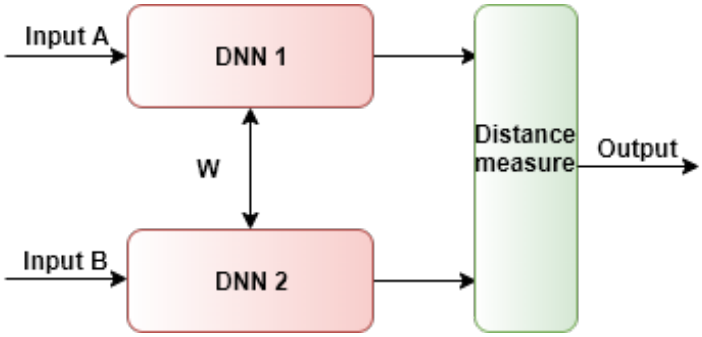


Figure 2.7: Basic architecture of a siamese neural network.

The idea behind the shared parameters (W), is that the network can extract comparable representations that provide valuable information about how similar or dissimilar the two inputs are.

2.3 Related work

Gait analysis has attracted a lot of attention from the biometric community, as a non-obtrusive and spoofing-resistant authentication method. There are two main approaches to accomplish the gait recognition task and these are the video-based approach and the inertial-sensor-based approach. During the past few years a lot of studies have been conducted using the inertial sensor approach for identification, authentication or activity recognition.

In [27] Semwal et.al. proposed a fully connected artificial neural network, ANN, for gait identification purposes. The input data were a fusion of vision-based and sensor-based data. PCA was used to reduce the dimensionality and extract only the principal features that were then used for training. The results of the model were compared with results obtained from two other classification techniques (KNN and K-means) and reported better performance for the ANN approach.

The authors of [28] implemented deep, convolutional and recurrent networks in order to investigate their suitability for human activity recognition (HAR) problems. A lot of experiments took place, using accelerometer and gyroscope data, which were gathered by people performing everyday activities. The experiments were conducted on three benchmark datasets, to enable the comparison

with other models. According to the results, different models were suitable for different tasks. RNNs were able to outperform CNNs and ANNs, for activities that were long in duration, while CNNs were more suitable for tasks short in period. The ANNs performance could approach the performance of CNNs and RNNs, however were found sensitive to the hyper-parameter settings.

In a more recent research[29], Giorgi et.al. were focusing on identification through gait recognition, based on acceleration data and deep learning. The dataset used was a collection of 5 sensors and was built by 153 people. Initially the data were pre-processed to extract gait cycles and to reduce the noise by filtering and normalization. Following, the 5 sensor data were used to train a CNN model. The task was formulated as a classification task and cross-entropy loss was used for the training. The results were quite promising as the method reached 95% on subject identification accuracy.

The same year a similar work of Dehzangi et.al. [30], was proposed for the gait identification task, using again 5 sensor accelerometer and gyroscope data on a set of 10 people. In this study, however, the gait cycles were extracted and then transformed into the time-frequency domain. Next two different techniques were followed to accomplish the identification. In the first case data from all the sensors were combined in one matrix and used to train one CNN classifier, as in [29] (early fusion approach). In the second case separate CNNs were trained based on the data from one individual sensor and the classification was achieved by combining the scores generated by all the individual CNNs (late fusion approach). According to the results the late fusion approach reported better performance than the early fusion one.

In [31] the authors employed a random forest approach for the person recognition task, using smartphone accelerometer data. The data were collected from 10 people and split into intervals of 100 samples. Subsequently features were extracted combining time and frequency domain characteristics, like mean, median, spectral centroid etc. A random forest classifier was used for the identification task, that consisted of 64 decision trees. Each tree used a random sub-sample of the dataset (bootstrap samples), and the final result was determined by aggregating the output of the individual trees. The results were compared with those obtained from Logistic Regression, SVM and Decision Tree classifiers and the best performance was reported on the Random Forest approach.

As mentioned before, gait analysis can also be used for authentication purposes. In this case the model is responsible of deciding if a specific identity matches the biometric sample, instead of assigning an identity to the sample [32].

In [5] the authors investigated both the identification and the authentication/ verification tasks. They built two different neural networks and used data collected from accelerometers and gyroscopes in smartphones. The people participated in the data collection procedure were 118 and they were allowed to walk freely in

any direction while carrying their phones. The inertial data were split into 128-samples intervals and were combined in a 2D matrix. For the identification task the input data were processed by a CNN to extract spatial features and by an LSTM to extract temporal features. Following the features were concatenated and led to a fully connected layer that perform the classification, using the cross entropy loss. The authentication task was also formulated as a classification problem. Two gait time-series, of the same or different subjects, were processed in parallel by a CNN to extract features. Following they were concatenated and led into an LSTM that performed the classification. The evaluation was achieved by comparing traditional methods (Fourier, Wavelet and EigenGait) with different deep learning approaches (LSTM, CNN, LSTM+CNN). For both the tasks of identification and authentication the deep learning approaches outperformed the traditional ones. Among the deep learning models CNNs performed better than the rest for the identification task while the combination of LSTM and CNN achieved the best results for the verification task.

The authors of [33], Wu et. al. based the authentication on video data. A Siamese neural network was proposed to compare the similarity between two different RGB videos. At each time step the video frames were processed by one CNN and then fed into a convolutional GRU, to extract features with temporal dependencies. Subsequently an average temporal pooling was applied to produce the final video representations. The loss function, used for training the model, was computed as the weighted inner product between the two representations. This formulation of the loss function allowed the proposed model to be applied on sequences of unknown classes. Experimental results in three benchmark datasets showed that the proposed method outperformed state-of-the-art approaches.

A similar approach was proposed in [34], however in this case 2 Siamese networks were combined. RGB frames of 2 different sequences were the inputs to one Siamese, as in [33], and the optical flow images were the inputs to the second Siamese network. Each network consisted of CNN layers leading to a temporal pooling to generate the final feature vector for each input sequence. The Euclidean distance was used to compare the similarity of the inputs to each Siamese network and subsequently these similarity values were combined in one similarity metric, using their weighted average.The results demonstrated that using both spatial (RGB) and temporal (optical flow) representations from different networks can actually improve the performance of the verification task.

Lastly in [35] Zhang et. al. proposed again an approach for gait verification based on video data. The model includes gait energy images (GEIs) and a Siamese neural network trained with the contrastive loss function. The video frames were converted into GEI representations to reduce the noise in the images and a model of 2 parallel CNNs was built and trained using similar or dissimilar pairs of GEIs. During the testing phase one GEI image was passing through one of the trained

CNN to extract the features and following the k-NN classifier was employed to identify a person with similar gait pattern. In [36] a similar approach was presented, where GEIs were used as inputs in a Siamese neural network, for the task of verification. Again the contrastive loss was used for training, however in this case, the testing phase was similar to the training. Two unknown GEIs were fed into the network and by calculating their similarity value (distance), an estimation was generated about whether they belonged to the same person or not.

Chapter 3

Methods

The aim of this project is to perform gait verification using accelerometer and gyroscope data and to investigate how the period of time, which we monitor users' gait, affects the accuracy of the predictions. To achieve this, we built classifier models using recurrent, feedforward and hybrid neural network architectures. In addition, we trained the models using gait time series of different durations.

The neural network models were implemented using deep learning platform PyTorch [37] and the training was completed in a virtual machine instance, which was launched on Google Cloud Platform[38]. For the training and testing of the models the dataset used was obtained in the previous work of Zou et.al [5]. The dataset consisted of accelerometer and gyroscope data, captured using the inertial sensors in the smartphones of the users.

3.1 Accelerometer and Gyroscope Dataset

In 2018, Q.Zou et.al [5] conducted a research, which aspired to perform user identification, based on gait biometric. To that end, a dataset of inertial sensor data was collected using accelerometers and gyroscopes from users' smartphones. Due to time constraints that did not allow us to gather our own data, the same dataset was employed for the fulfillment of the present project as well.

During the data collection period, inertial sensor data were gathered by 118 participants that had no restriction on the manner, the place or the speed of walking. The equipment used was the users' smartphones and the sampling rate of all sensor data was set to 50Hz. Among the 118 participants 20 of them provided larger amount of data in a period of 2 days and the rest 98 participants a smaller amount in a period of 1 day. After the data collection period was over, the non-walking parts of the data were excluded and the remaining walking parts were divided into separate steps (a step is completed when the same foot touched the

ground twice).

Construction of gait verification dataset

For the gait verification task, where the purpose is to train DNN models in order to be able to decide if two gait patterns belong to the same person or not, the dataset constructed was the following:

The dataset consists of 66,542 verification samples for training and 7,600 verification samples for testing the models. Each verification sample consists of 3-axis accelerometer and 3-axis gyroscope data (6 features). Out of the 118 participants, the 98 participants were used for the training set and the remaining 20 participants were used for the test set. Each verification sample contains a pair of samples that belong to the same user or to two different users. In order to create the samples the gait patterns were divided into 2-step sections and linear interpolation was used in order to fix the number of data-points per sample to 128.

	# of users	**# of samples**	**Sample duration**	**# of data-points per sample**
train set	98	66,542	2-steps	128
test set	20	7,600	2-steps	128
overall	118	74,142	2-steps	128

Table 3.1: Detailed information about verification dataset 1.

Table 3.1 provides in compact form all the information about the dataset 1 and the figure 3.1 provides an insight on how the accelerometer and gyroscope data look like.

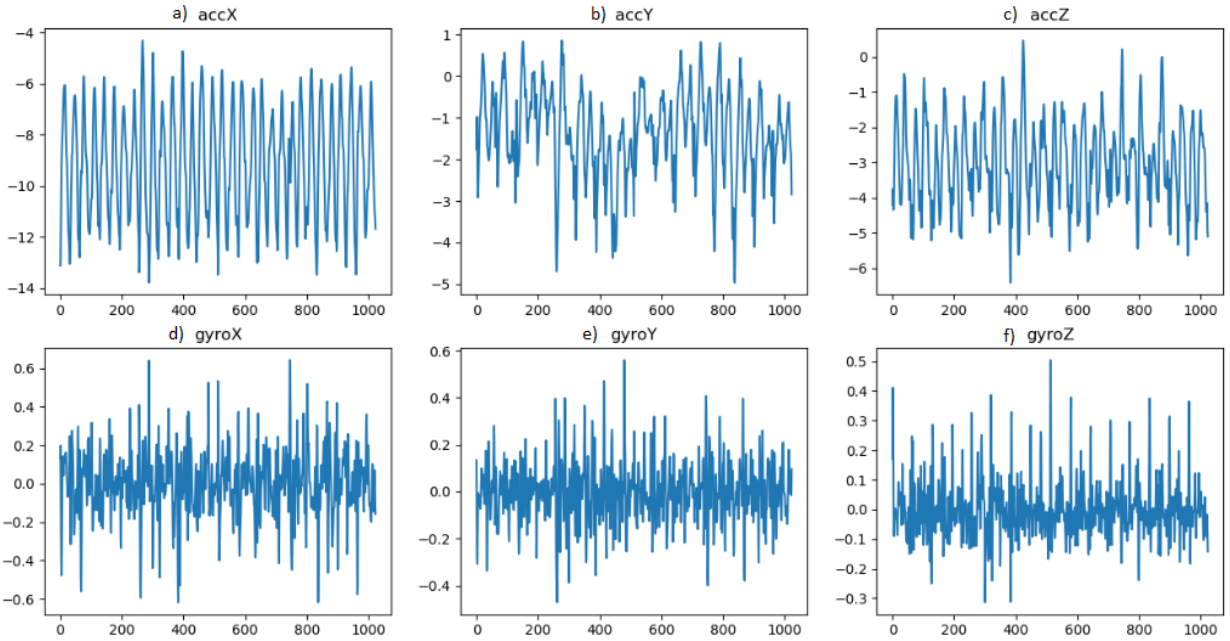


Figure 3.1: Acceleration and gyroscope data captured from the inertial sensors in the smartphones. a) Acceleration along X axis, b) acceleration along Y axis, c) acceleration along Z axis, d) angular velocity along X axis, e) angular velocity along Y axis, f) angular velocity along Z axis.

Creating datasets with samples of different lengths

For the purpose of the present project, which was to investigate the most suitable temporal duration, in order to accurately verify a user, we had to further manipulate the dataset.

In order to be able to train and test the DNN models with data samples of different lengths, we initially created samples by dividing the gait patterns into sections of 1, 4 and 8 steps. Continuously, we paired two gait samples, that contained 3-axis accelerometer and gyroscope data and created a verification sample, as explained before. As follows, we ended up having 4 different datasets, as shown in tables 3.1, 3.2, 3.3 and 3.4. All four datasets consisted of 66,542 verification samples for training and 7,600 samples for testing.

	# of users	# of samples	Sample duration	# of data-points per sample
train set	98	66,542	1-step	64
test set	20	7,600	1-step	64
overall	118	74,142	1-steps	64

Table 3.2: Detailed information about verification dataset 2.

	# of users	# of samples	Sample duration	# of data-points per sample
train set	98	66,542	4-steps	256
test set	20	7,600	4-steps	256
overall	118	74,142	4-steps	256

Table 3.3: Detailed information about verification dataset 3.

	# of users	# of samples	Sample duration	# of data-points per sample
train set	98	66,542	8-steps	512
test set	20	7,600	8-steps	512
overall	118	74,142	8-steps	512

Table 3.4: Detailed information about verification dataset 4.

Data pre-processing

In raw form gyroscope data are centered around zero with a maximum radius of 0.6 (figure 3.1). However, this does not apply for the accelerometer data, where we noticed a greater variance between the values of the 3 axes. For this reason, we normalized the data by rescaling the training set, per feature, according to the mean and standard deviation using the equation 3.1

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

where z are the normalized data, x are the raw data, μ is the mean and σ is the standard deviation of the x. The mean and standard deviation were computed using only the samples in the training set and were used for normalizing all samples in both training and test sets.

Further manipulation of the datasets

In the context of this project it was desirable to examine as well, whether the performance of the proposed approach could be improved by excluding any potential noise in the data. According to Han et.al. [39] most of the human gait is below 15Hz and a low pass filter can be applied in order to cut-of the information in all other frequencies. Thus, in order to observe how the information was spread in our data, we generated the spectrogram of our inertial sensor data and visualized the spectrum of frequencies and the way the information is spread among them.

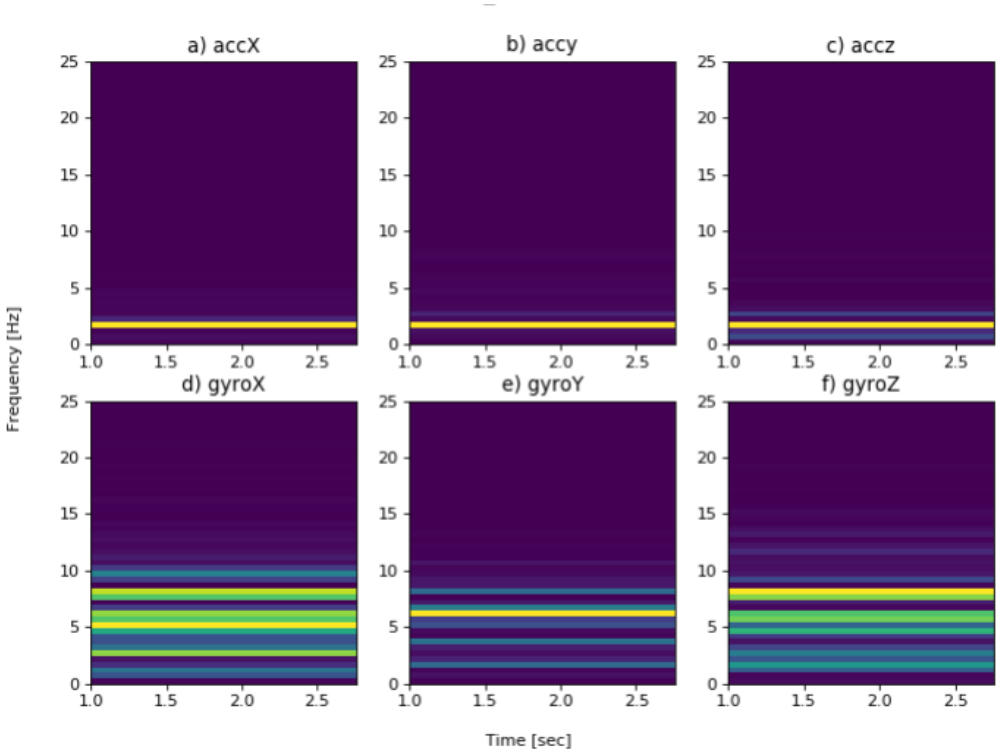


Figure 3.2: Spectrogram of the inertial sensor data that shows where the most information is gathered. The image shows the intensity of the gait signals, and how it is distributed among different frequencies. Plots a), b), c), d), e) and f) show the spectograms of the acceleration and the angular velocity along the axes X, Y and Z, respectively.

Figure 3.2 shows the corresponding frequencies. The accelerometer data have a compact form and according to the figure 3.2 the higher intensity of the signal is gathered around the frequency of 2Hz. Thus, we can conclude that most of the information in accelerometer data is around the frequency of 2Hz.

On the other hand, the information of the gyroscope data is spread in a wider range of frequencies (1-11Hz). The intensity of the spectral components above 8Hz is low, so we decided to exclude these frequency components. To that end, we used a low pass filter and set the cut off frequency equal to 8Hz, in order to keep all the information below that frequency and exclude the rest. Figure 3.3 shows the inertial sensor data before and after the low pass filter was applied.

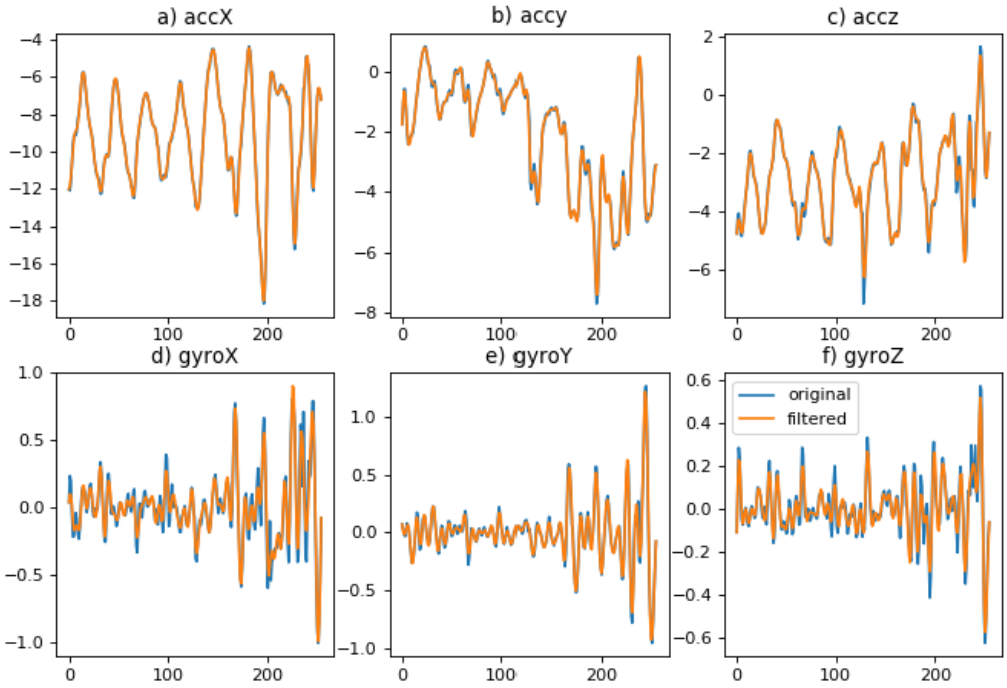


Figure 3.3: Inertial sensor data before and after the low pass filter. The orange color denotes the filtered gait signal and the blue color denotes the original signal. Plots a), b), c), d), e) and f) show the original and filtered signal of the acceleration and the angular velocity along the axes X, Y and Z, respectively.

3.2 Models

The gait verification task is approached as a classification problem. Three different DNN models were built and were trained in a supervised way in order to decide if 2 gait patterns belong to the same user or 2 different users.

For the current project, 3 deep neural networks were built inspired by the work of Q.Zou et.al [5]. However they were implemented using the siamese architecture. Siamese neural networks are commonly used for verification tasks as it is mention in the section 2.3 *Related work*. The 3 models are listed below:

- Siamese CNN
- Siamese LSTM
- Siamese CNN+LSTM

3.2.1 Siamese CNN

This model consists of two identical convolutional networks of the structure shown in the figure 3.4. There are 4 layers of 1D CNNs followed by ReLU activation functions after each other to add non linearity to the model. In all of the 4 layers dropout regularization is applied with a probability of 30% and 1D max pooling layers are apllied after the first and the third layer, according to the suggestions in [5]. Following the CNN layers the features extracted are flattened into 1D vectors and the pair-wise absolute difference of them is computed. Lastly 2 fully-connected layers were applied with ReLU activation and dropout of 30% chance, that were trained on the aforementioned absolute difference. The output layer of the model was implemented with a softmax activation. For the optimization of the model the Adam optimizer was chosen, both for the reasons that it was suggested by [5] and because it was designed specifically for training neural networks [17].

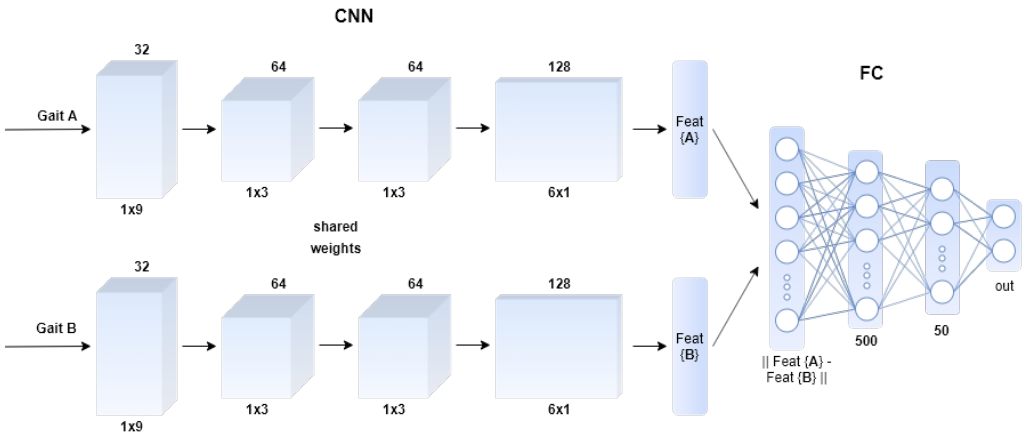


Figure 3.4: Architecture of the siamese CNN model.

The table 3.5 below, shows in detail the structure of the CNN layers in the siamese CNN model and the ouput size of each layer in case were the input is $6 \times 128 \times 1$. Since our data are time-series of sensors measuring the acceleration and angular velocity over time we used 1D CNNs, meaning that the stride and

padding for each layer extends in 1 dimension as well, in the direction of each feature separately. For this model the input data are arranged in a 2D vector of size $6 \times data\_points$, were 6 is the number of features per sample and $data\_points$ is the length of the sample (length of 2, 4, or 8 steps that is 128, 256, 512 data points respectively).

	N. Kernels	**kernel size K**	**stride S**	**paddin P**	**output size (W-K+2P)/S+1**
conv1	32	1x9	2	4 (horiz.)	6x64x32
max_pool1		1x2	2		6x32x32
conv2	64	1x3	1	1 (horiz.)	6x32x64
conv3	64	1x3	1	1 (horiz.)	6x32x64
max_pool2		1x2	2		6x16x64
conv4	128	6x1	1		1x16x128

Table 3.5: Details of the CNN layers. The calculations of the output size were made using the sample length of 2 steps i.e. W = 128 data points.

3.2.2 Siamese LSTM

The second model follows the siamese architecture as well, however the 2 identical CNNs of the previous case are replaced by 2 identical LSTMs neural networks. Figure 3.5 shows the temporally unfolded architecture of the model. The siamese LSTM model focuses on capturing the temporal relation of the input data. It consists of 2 LSTM layers each of which has a hidden layer with a number of 64 hidden nodes. Dropout regularization is applied after each layer with probability of 30%.

The input data for the second model are arranged in a 2D vector of size $data\_points \times 6$ (meaning 128×6, 256×6 or 512×6) and each row of the vector is fed one after the other into the model, representing subsequent time-steps. The hidden state of the last time-step, T, includes the temporal information of the whole time-series and is used as the extracted feature vector. Following the feature extraction, the absolute pair-wise difference of the 2 feature vectors is computed and fed into a fully-connected layer with the same formulation as in the previous model. Adam optimizer and softmax activation for the output layer were used for this model as well.

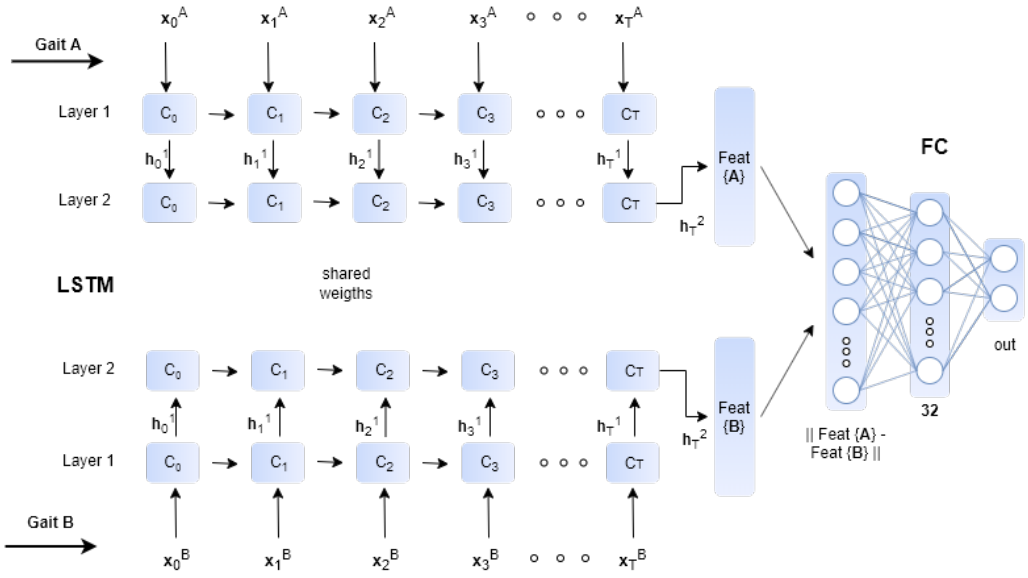


Figure 3.5: Temporally unfolded architecture of the siamese LSTM model.

3.2.3 Siamese CNN+LSTM

The final model is a combination of the 2 aforementioned models. Here two identical CNNs are used in order to extract spatial features from the input data. The features are arranged into 2D vectors of size 16×128, 32×128 or 64×128, depending on the size of the input data (2, 4 or 8 steps respectively). However this time, instead of computing their pair-wise absolute difference, they are concatenated row-wise into a new 2D vector of size $16/32/64 \times 256$. Following the 2D feature vector is fed row by row into an LSTM network, where the temporal relationship is captured. Lastly the hidden layer of the last time-step, a 1D vector of size 1×64 is used as input to the output layer. The architecture of the individual networks are the same as in the previous models as it is shown in figure 3.6.

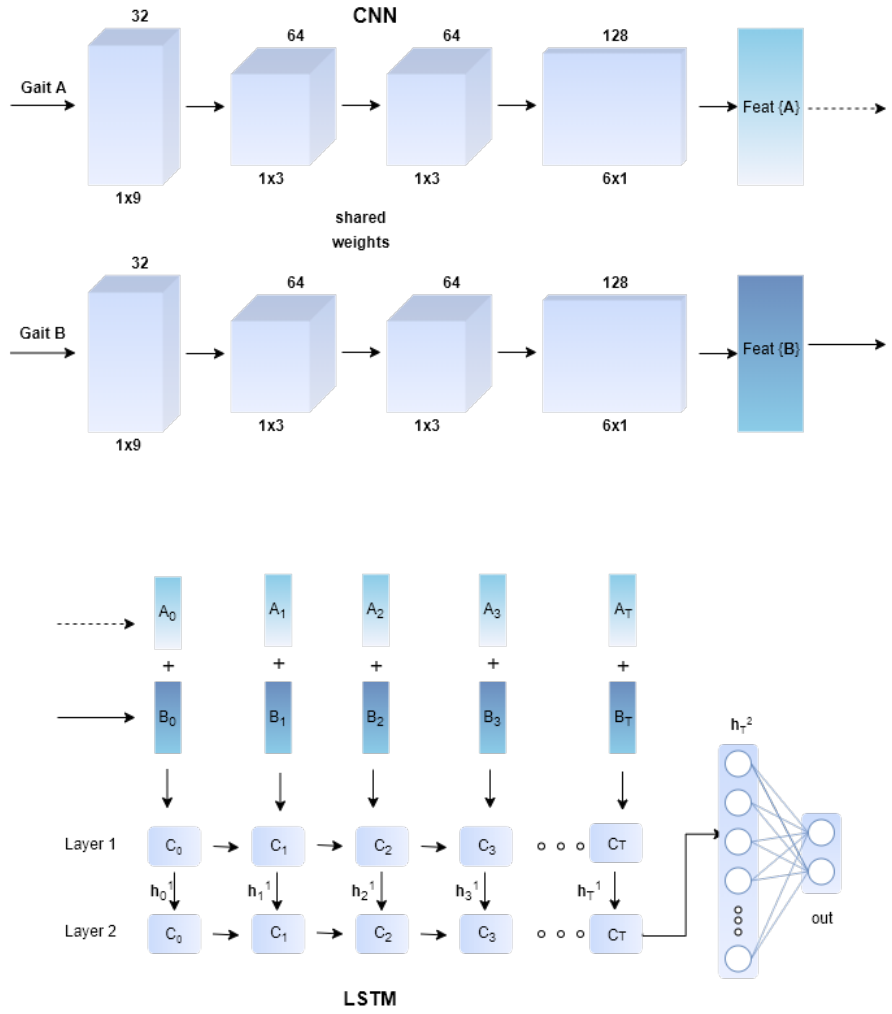


Figure 3.6: Architecture of siamese CNN+LSTM network

3.2.4 Training details

Loss function

As it has already been stated, the task of gait verification is formulated as a classification problem with only 2 classes: the pair of samples either belongs to one user or two different users. Thus, the binary cross entropy loss, equation 3.2, was used for the training of the models.

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i log(out_i) + (1 - y_i) log(1 - out_i) \tag{3.2}$$

In the equation above, N is the number of sample-pairs in the training set, y_i is either 0 or 1 depending on the class where the i_{th} sample-pair belongs and out_i is the probability that the i_{th} sample belongs to the positive class.

The probabilities are computed by the networks. The last layer of every model outputs a 2x1 vector of scores that is fed to the softmax activation function, as it was described above. Softmax then converts the vector of scores into a vector of probabilities that sum to 1 and indicate the likelihood that one pair of gaits belongs to the same person or not. The form of softmax activation function is presented in equation 3.3.

$$out_i = \frac{e^{score_i}}{\sum_{j=1}^{2} e^{score_j}} \tag{3.3}$$

where $score_i$ is the output of the networks before the softmax function.

Early stopping

In order to train the models we used a varying number of epochs extended from two hundred to four hundred. Training a neural network, however, over a large number of epochs might lead to overfitting. This means, that the network is well trained on the training set, but cannot generalize on unseen data. In order to minimize that risk, the early stopping method was deployed. This method guarantees that the training ends, if the performance on the validation set has not improved for a certain number of epochs. Similar to the number of epochs, different number of patience were tried while training the networks extended from 20 to 100.

Hyper-parameters selection

Due to time limitations the hyper-parameters values were selected according to the guidelines from the previous work of Zou et.al [5]. A small search around the area of the suggested values was performed, however not in a systematic way. For the parameters for which there was no reference in the paper (dropout probability and patience), we tested some random values and we kept those that performed the best.

3.2.5 Evaluation metrics

The most commonly used metrics for biometric systems are the *False Acceptance Rate* and the *False Reject Rate*.

False Acceptance Rate

This metric is also known as "Type II" error and refers to the expected proportion of attempts with wrongful claims of identity, that are incorrectly confirmed. It is computed as follows:

$$\frac{false\_acceptances}{all\_impostor\_pairs} = \frac{FP}{FP+TN} \tag{3.4}$$

where FP is false positive attempts and TN is true negative.

False Reject Rate

This metric is also known as "Type I" error and refers to expected proportion of attempts with truthful claims of identity, that are incorrectly denied. This metric is computed as follows:

$$\frac{false\_rejections}{all\_genuine\_pairs} = \frac{FN}{FN+TP} \tag{3.5}$$

where FN is the number of false negative attempts and TP is the number of true positives.

Detection Error Trade-off (DET) curves

We also make use of Detection Error Trade-off curves. These curves show a range of error rates for binary classification systems that perform detection tasks, as the threshold varies to alter the false negative and false positive rates [40] .

Chapter 4

Experiments and evaluation

This chapter will present the experiments that were conducted. The results of the experiments will be evaluated and discussed, in order to conclude on the performance of the various proposed approaches.

4.1 Experimental set-up

In this section we are going to present a list of the different experiments that we conducted and the metrics that were used for the evaluation of the proposed models.

4.1.1 List of experiments

The experiments were divided into 3 categories:

1. **Experiments with unfiltered data**
   This was the main experiment of the current thesis that investigated the research question. In this series of experiments all the models were trained and tested using raw data, as they were captured from the inertial sensors. All the models were tested using the 4 different sample lengths (1, 2, 4 and 8 steps of gait) in order to evaluate how this parameter affects their performance.

   The absence of a second dataset, which could be used to assess the robustness and stability of our models, was a drawback that needed to be overcome. To that end we performed 10-fold cross validation to create 10 dummy datasets and evaluate the performance of the models.

2. **Experiments with low-pass filtered data**
   This experiment was consider to be an additional experiment to further

explore some aspects of the problem. For this set of experiments the models were trained using the low-pass filtered data, in order to investigate if the elimination of possible noise in the raw data could improve the performance of the approaches. As in the first group of experiments, all the models were tested with the 4 different sample lengths.

3. **Multiple trials experiments**
   This set of experiments was also an additional one and was conducted in order to assess the performance of the models, when instead of verifying the users by one attempt, we were performing multiple of them. If at least one of these attempts was successful then the user was verified. According to Conrad et.al.[41] there is always a trade off between the values of FAR and FRR. The purpose of this experiment was to investigate how the FAR and FRR metrics behave when more that one authentication trials were used.

4.2 Results

4.2.1 Experiment 1: Unfiltered data

Tables 4.1, 4.2, 4.3, 4.4 and figures 4.1, 4.2, 4.3 show the results of the experiment with the unfiltered data for the 4 datasets of gait lengths of 1, 2, 4 and 8 steps,and the 3 DNN models. The tables provide the results obtained by performing 10-fold cross validation and the figures provide the results obtained from the test data.

By an initial observation of the test results in figures 4.1, 4.2 and 4.3, a general trend is that the accuracy increases as the length of gait increases from 1 to 8 steps. The accuracy increases from 86.66% (LSTM model for 1 step of gait) to 93.79% (CNN+LSTM model for 8 steps of gait). Another observation is that the LSTM network performed the worst, in terms of the accuracy, for the first 2 datasets of 1 and 2 steps, while the CNN+LSTM network provided better results for the 4 datasets.

When it comes to the FAR and FRR values we know that the lower the FAR and FRR values, the more accurate the system. From the results in the figures we observe that FAR is decreasing as the number of steps increases while FRR fluctuates slightly. However the decrease is not significant. The lowest FAR and FRR values are 0.04 and 0.08, respectively, and are produced by the CNN+LSTM network for 8 steps. Another observation is that the LSTM network tends to provide better FAR and FRR values than the CNN for the last 2 datasets of 4 and 8 steps of gait. In addition, FAR and FRR seem to have comparable values and small differences among the models.

However, these observations are not supported by any statistical evidence and they can be used only to obtain a initial opinion about the trends of the results.

At this point, we make claims about the performance of the models, as there is no clear tendency that would help us reach some conclusions about the most accurate approach.

Even though it is not clear which method is the most accurate, there is at least an indication that the combination of capturing both the temporal and spatial data provides slightly better results than just capturing one of them. However, in order to reach to valuable conclusions we performed statistical analysis.

	10-fold cross validation	
	mean acc (%)	**std**
CNN	98.99	0.12
LSTM	98.12	0.27
CNN+LSTM	96.98	0.46

Table 4.1: Experiments results using the dataset with gait length of 1 step. The table provides the mean accuracy and standard deviation (std) of a 10-fold cross validation test.

	10-fold cross validation	
	mean acc (%)	**std**
CNN	99.28	0.10
LSTM	98.44	0.23
CNN+LSTM	97.77	0.32

Table 4.2: Experiments results using the dataset with gait length of 2 steps. The table provides the mean accuracy and standard deviation (std) of a 10-fold cross validation test.

	10-fold cross validation	
	mean acc (%)	**std**
CNN	99.79	0.09
LSTM	99.31	0.24
CNN+LSTM	98.64	0.39

Table 4.3: Experiments results using the dataset with gait length of 4 steps. The table provides the mean accuracy and standard deviation (std) of a 10-fold cross validation test.

	10-fold cross validation	
	mean acc (%)	**std**
CNN	99.81	0.25
LSTM	99.75	0.10
CNN+LSTM	99.28	0.13

Table 4.4: Experiments results using the dataset with gait length of 8 steps. The table provides the mean accuracy and standard deviation (std) of a 10-fold cross validation test.

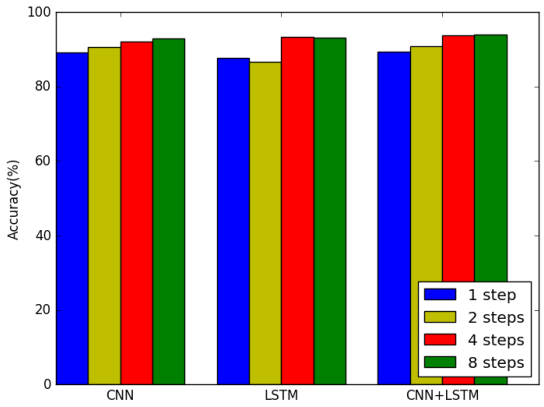


Figure 4.1: Accuracy for all the models and the different number of steps.

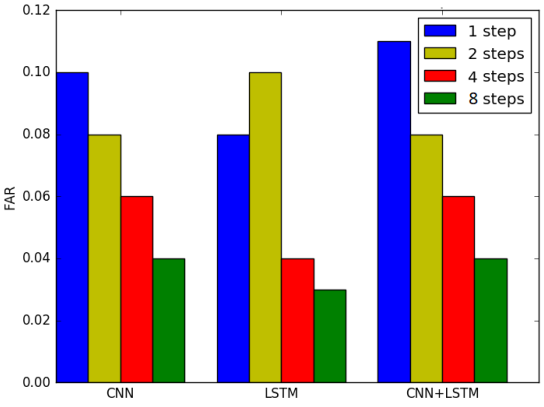


Figure 4.2: FAR for all the models and the different number of steps.

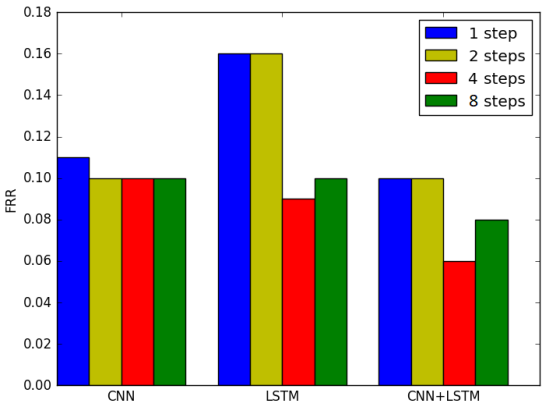


Figure 4.3: FRR for all the models and the different number of steps.

Decision Error Tradeoff curve (DET curve)

Figure 4.4 shows the DET curve of FRR and FAR values for different decision thresholds. From this figure we are able to obtain a better understanding of the behavior of the models for different number of steps.

As shown in Figure 4.4 (a) all the models appear to have a similar behavior for the dataset of gait with 1 step. CNN appears to have slightly higher values for the FAR and FRR, however the difference with the other models is almost unnoticeable.

For the dataset of 2 steps of gait, 4.4 (b) though, the performance of the LSTM and CNN+LSTM models is improving, while the CNN seems to have almost the same behavior with the previous dataset. The best model for this dataset is the CNN+LSTM, which differentiates enough from the other 2 models.

In Figure 4.4 (c), it can be observed that the performance has improved a lot in comparison with the previous datasets. The values of FRR are kept low enough while the FAR need to be further improved. Once again the CNN model appears to have greater values that the other 2 models, which have comparable performances.

Lastly, for the case of the dataset of 8 steps of gait, we observe that the choice of the model does not really play a significant role in terms of the performance and the values of FAR and FRR. All models provide quite similar values of FAR and FRR metrics, which in this case they have further decrease and are the lowest among all the tests cases.

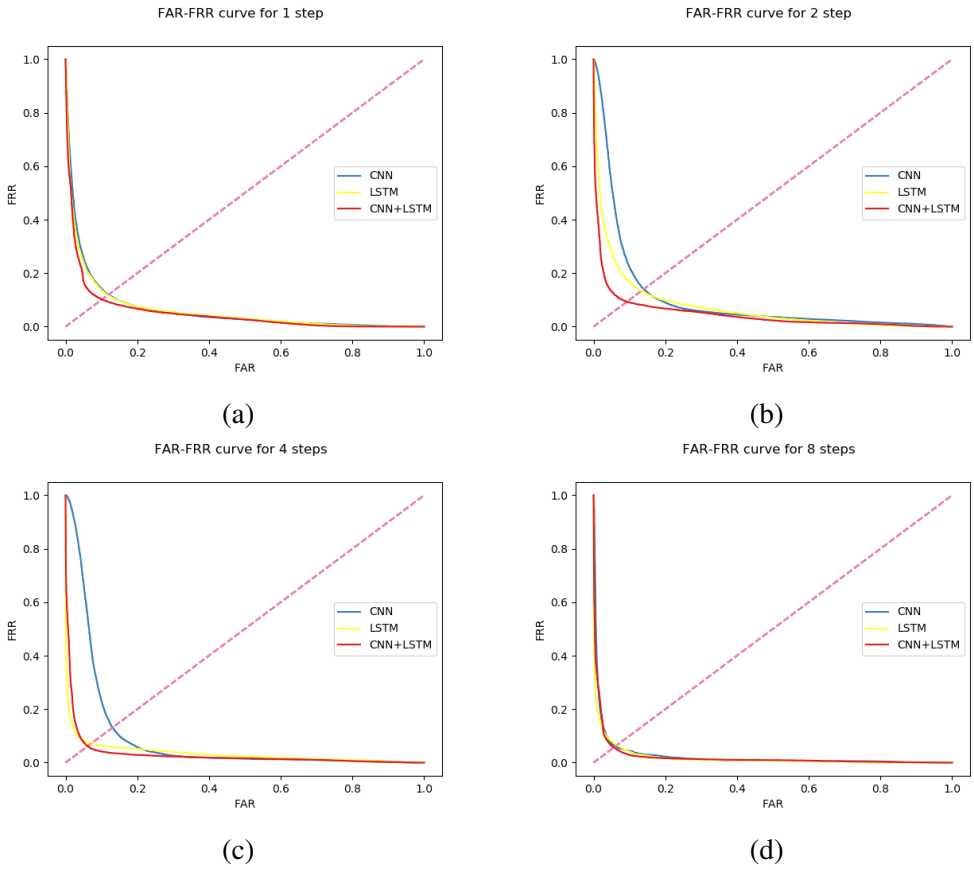


Figure 4.4: DET curves for CNN, LSTM and CNN+LSTM models. Plots (a), (b), (c), (d) corresponds to gaits of 1, 2, 4 and 8 steps, respectively. The figures represents the change of FAR and FRR values for different decision thresholds. CNN is depicted with blue colour, LSTM with yellow and CNN+LSTM with red.

Confusion matrices

Figure 4.5 shows some confusion matrices from tests of different gait lengths. More specifically the figure illustrates which users from the 20 users of the test set were faulty accepted as other users of the test set.

The CNN model had mostly difficulty in predicted correctly participants 0, 1, 2, 7, 9 and 16. The issue persisted even when 8 steps of gait were used to predict a user. However, the number of times that they were mismatched had decreased.

Almost the same pattern was observed in the case of the LSTM model, where the most mismatched users were the users 0, 1, 2, 3, 7, 9 and 16. Once more, the use of gait patterns with 8 steps managed to decrease the divergence, even

though the model was still unable to predict some users. Specifically, user 1 was constantly predicted as user 7, and vice versa while user 17 was predicted as user 16.

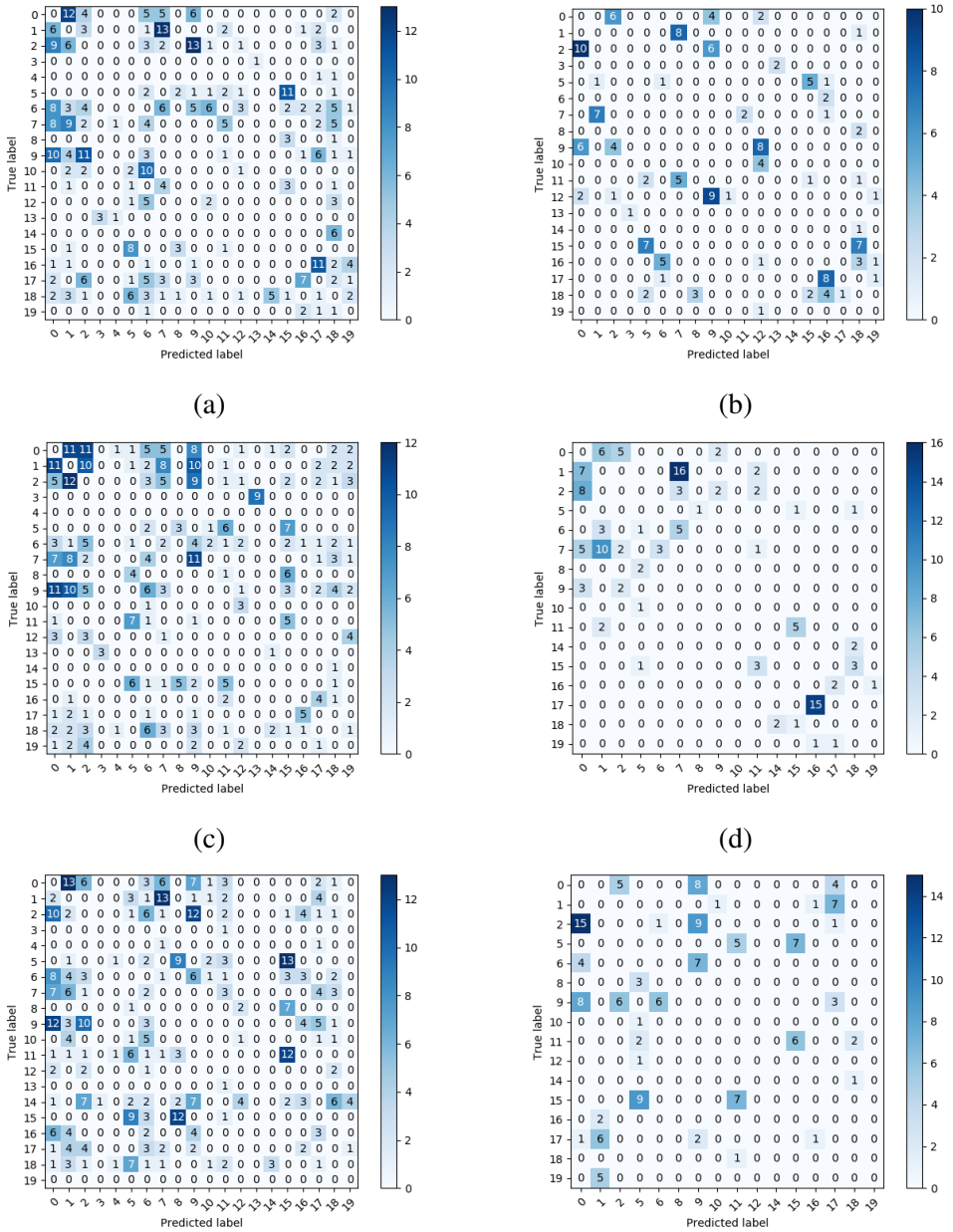
The smaller number of mismatches was observed when the CNN+LSTM model was used. However the decrease in comparison to the other models was not as impressive, according to the confusion matrix in Figure 4.5 (e). In the case of 1 step of gait the number of users being mistaken with other users was comparable to previous models. However, when 8 steps of gait were used there was a sufficient decrease in the number of false acceptances. It should also be mentioned that the users that were mostly mistaken with other users were the same as in the previous cases.

The fact that the same users were mostly mistaken with others (and the same others) in most of the cases, leads us to believe that the aforementioned users have similar gait patterns with each other. According to the results from Figure 4.5, it becomes clear that none of the models were sensitive enough to capture the small differences between their gaits, causing high values of false acceptances. The CNN+LSTM models seemed to perform slightly better than the others, however it is obvious that a more sophisticate approach is needed in order to capture the smaller details.

Analysis of Variance (ANOVA) test

Having only one dataset in our disposal, in order to investigate the gait verification task and obtaining results that were controversial and not as clear as we expected we decided to perform analysis of variance test (ANOVA test) in order to conclude if any of the models performed better and if the choice of the number of gait steps, actually improved the performance.

To set up the ANOVA test we needed to specify the independent and the dependent variable. In our case we had 2 independent variables (factors): *models* and *number of steps* and one dependent: *accuracy*.



(a) (b) (c) (d) (e) (f)

Figure 4.5: Confusion matrices for CNN (a, b), LSTM (c, d) and CNN+LSTM (e, f) models and for gaits of 1 (a, c, e) and 8 (b, d, f) steps.

Values or levels of the *models* factor:

- CNN
- LSTM
- CNN+LSTM

Values or levels of the *number of steps* factor:

- 1 step
- 2 steps
- 4 steps
- 8 steps

Next we fitted the two way ANOVA algorithm with the results we obtained from the 10-fold cross validation test in order to reject or accept the null hypotheses. In the case of 2-way ANOVA there are 3 null hypotheses:

1. There is no significant difference in response (accuracy) with respect to the choice of the models, i.e. all models means are equal.
2. There is no significant difference in response (accuracy) with respect to the number of steps, i.e. all number of steps provide the same means.
3. The is no significant interaction effect between the models and number of steps in terms of the accuracy.

In order to reject the null hypotheses we needed to obtain p-values that are less than the *significant level*, α. In most cases, a significance level of 0.05 works well and it indicates the 5% risk of concluding that an effect exists while in reality there is no effect. Table 4.5 show the results from the 2-way ANOVA.

	Df	Sum Sq	Mean Sq	F value	p-value
Models	2	33.939	16.969	259.203	1e-5
Number of steps	3	46.236	15.412	235.415	1e-5
Models: Number of steps	6	6.179	1.03	15.731	1e-5

Table 4.5: Results from 2-way ANOVA test for the main factors: *Models* and *Number of factors* and the interaction: *Models: Number of steps*

According to the results and the fact that all p-values are close to 0 (less that the significant level) we reject the null hypotheses and conclude that:

1. The different levels of the factors *models* and *number of steps* are associated with differences in the accuracy.

2. There is an interaction effect between *models* and *number of steps*, which indicates that the correlation between *models* and the accuracy depends on the levels of *number of steps*.

These results can be also seen in the Figure 4.6. In this figure we observe that there is and upward trend in the accuracy values as the number of steps are increasing. In addition, the fact that the 3 lines corresponding to the models are not parallel indicate that there is actually and interaction between the 2 factors.

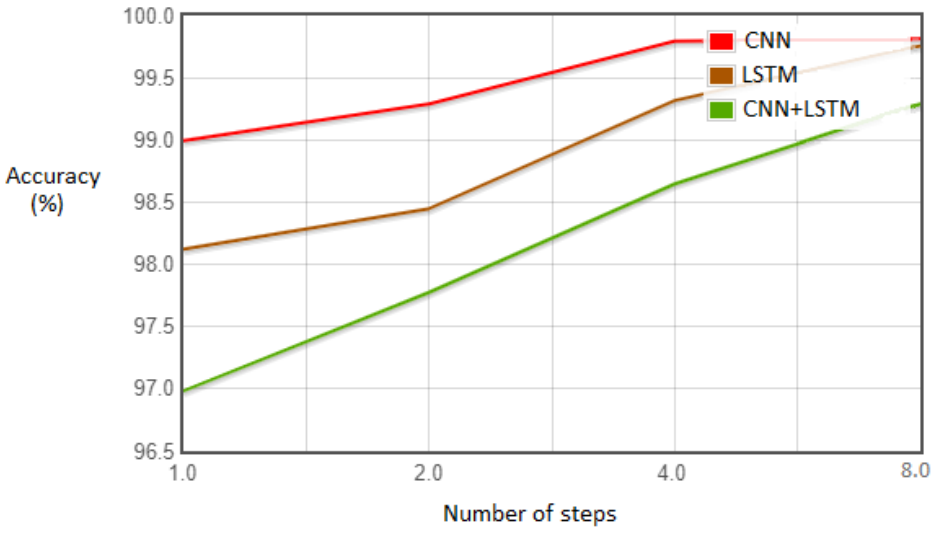


Figure 4.6: Accuracy means for the three different models at different number of steps.

Having accepted that there is actually connection between the accuracy and the 2 factors, the next step was to investigate the origin of differences in accuracies reported by ANOVA. To that end we performed a post hoc test to assess the way these 2 factors affect the accuracy. The Tukey's HSD (Honest Significant Difference) test was applied which compares the means between each level in every factor and each level between the 2 factors. Tables 4.6 and 4.7 show the comparison between the levels of the factor *models* and *number of steps* respectively.

HSD = 0.14	diff	lwr	upr	p adj
CNN+LSTM vs CNN	-1.299	-1.435	-1.163	0
LSTM vs CNN	-0.562	-0.698	-0.426	0
LSTM vsCNN+LSTM	0.737	0.601	0.873	0

Table 4.6: Tukey's HSD results for the comparison between the levels of factor *models*. *Diff* is the mean differences between the levels of the factor. *Lwr* and *upr* are the lower and the upper end points of the 95% confidence interval of *diff*. If the difference between 2 levels is higher than the *HSD* value , or if $p\ adj < .05$ then the difference is significant.

HSD = 0.17	diff	lwr	upr	p adj
2steps vs 1step	0.471	0.299	0.643	0
4steps vs 1step	1.22	1.047	1.392	0
8steps vs 1step	1.586	1.414	1.759	0
4steps vs 2steps	0.749	0.576	0.921	0
8steps vs 2steps	1.115	0.943	1.288	0
8steps vs 4steps	0.367	0.194	0.539	0

Table 4.7: Tukey's HSD results for the comparison between the levels of factor *number of steps*.

As we can see from the results there are significant differences between all the different levels of the factor *models* and the factor *number of steps*. That means that by choosing different models and number of steps we actually affect the accuracy. According to the results the best model appears to be the CNN model and the best choice for the mumber of steps the 8 steps.

However the 2-way ANOVA pointed out that there is interaction between the 2 factors. In order to observe how different combinations of the 2 factors affect the accuracy, we had to compare the pair-wise mean differences for all the combinations of *models* and *number of steps*. Applying the Tukey's test, that led to $4^3 = 64$ pair-wise comparison of which for obvious reasons, we only report the most interesting ones in Table 4.8.

Row	HSD = 0.38	diff	lwr	upr	p adj
1	CNN:2steps vs CNN:1step	0.295	-0.087	0.677	0.305
2	LSTM:2steps vs LSTM:1step	0.327	-0.055	0.709	0.172
3	CNN+LSTM:2steps vs CNN+LSTM:1step	0.749	0.576	0.921	0
4	CNN+LSTM:1step vs CNN:1step	-2.01	-2.392	-1.628	0
5	CNN:2steps vs CNN+LSTM:1step	2.305	1.923	2.687	0
6	CNN:4steps vs CNN+LSTM:1step	2.81	2.428	3.192	0
7	CNN+LSTM:2steps vs CNN:2steps	-1.514	-1.896	-1.132	0
8	LSTM:2steps vs CNN:2steps	-0.841	-1.223	-0.459	0
9	LSTM:2steps vs CNN+LSTM:2steps	0.673	0.291	1.055	0
10	CNN+LSTM:8steps vs CNN:8steps	-0.524	-0.906	-0.142	0.001
11	LSTM:8steps vs LSTM:1step	1.635	1.253	2.017	0
12	LSTM:1step vs CNN+LSTM:1step	1.137	0.755	1.519	0

Table 4.8: Tukey's test results for the interaction of the 2 factors *models* and *number of steps*. Provides the mean differences between the different levels of the 2 factors.

From the results in Table 4.8 we observe (in the first 2 rows) that choosing 1 or 2 steps of gait with the CNN model or the LSTM model does not affect the accuracy of the system significantly. However, it does improve the performance the choice of 2 steps of gait instead of 1 when using the CNN+LSTM model. Another interesting observation is that as the number of steps increases the choice of the model even though it affects the accuracy, it appears to have less impact. This can be seen in rows 4, 7 and 10, where the differences in performance of CNN and CNN+LSTM models for gait of 1 step is higher (-2.01) than for the gait of 2 steps (-1.514) and even higher from the gait of 8 steps (-0.524).

From these results and from the Figure 4.6, we can conclude that the factor *number of steps* has actually greater impact to the accuracy than the factor *models*. Another conclusion is that the choice of the model matters more for gaits of 1 or 2 steps, while it does not make a big difference for gaits of 8 steps.

4.2.2 Experiment 2: Low pass filtered data

Figure 4.7 shows the results of the experiment with the low pass-filtered data. For this experiment we only tested the trained models with the test set of 20 participants in order to observe if there are any differences between the results of the test set in experiment 1.

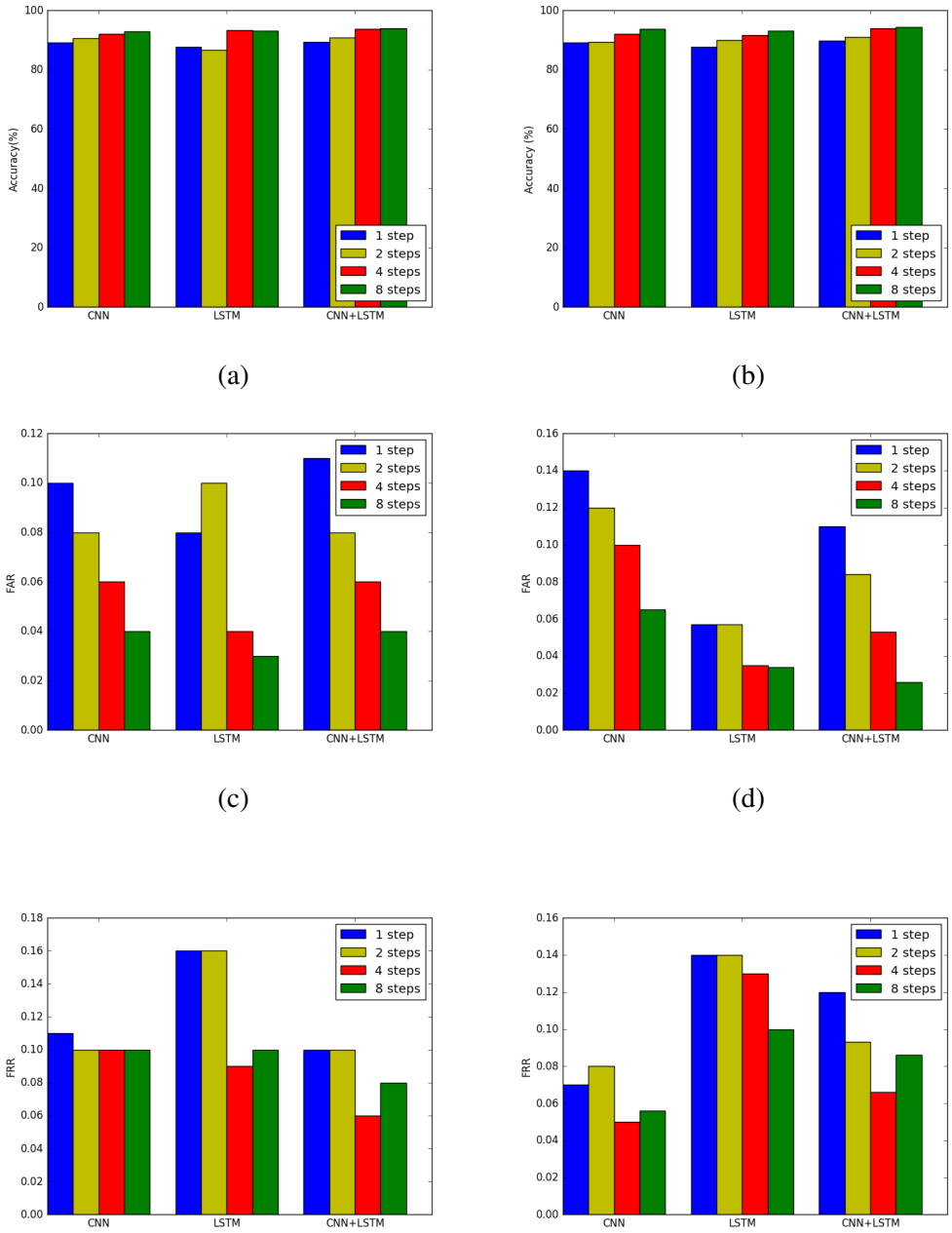


Figure 4.7: Accuracy (a, b), FAR (c, d) and FRR (e, f) plots for all the models and different number of steps using unfiltered (a, c, e) and low-pass filtered data (b, d, f) of the test set.

According to the results of the low passed filtered data, we observe that they

follow almost the same trend as the unfiltered data. More specifically the accuracy of the models increases when the number of steps increases and at the same time the FAR and FRR values decrease. In addition, the model that appears to have the best performance is the CNN+LSTM model like in the first experiment, however there are no strong differences among the values of the metrics that could lead us to a positive conclusion.

The main reason for applying the filter to the data, was to investigate if by excluding some information, that might have been noisy, we could improve the predictive ability of the systems. Although, the results we obtained were slightly better only in some cases when using the filtered data (LSTM-2steps gait, CNN+LSTM-4steps gait) and in some other cases the low pass filtered data performed slightly worse (CNN+LSTM-1step gait, LSTM-4steps gait). Thus, no clear conclusion can be made about the behavior of the filtered data and further investigation is required.

4.2.3 Experiment 3: Multiple trials experiments.

Table 4.9 below contains the results from the last experiment of multiple verification trials. In this experiment we examined how the FAR and FRR metrics behave after performing 1, 2, 3, 4, 5 or 6 verification trials. The set of data that was used for experiment 3 was the test set of 20 participants, as in the previous experiments.

From the table we observe that the FRR values are decreasing while the number of trials increases. As expected, the FAR values are increasing as well, however the rate of the modification in not the same for FAR and FRR. The increasing rate of the FAR values is lower that the decreasing rate of the FRR values, when we use 2 verification trials instead of 1.

As the number of trials increases the decrease rate of the FRR seems to decline as well, although the FRR values for most of the cases approach really low values (magnitude order of -3). On the other hand, in most cases the increase in the number of trials does not seem to really affect the increase rate of the FAR values, which is almost always 0.02 or 0.01.

Another interesting observation is that for gaits of more steps the increase rate of FAR is lower than for gaits of less steps; the increase FAR rate of 8 steps is around 0.01, while the same rate for 4 steps is around 0.3.

According to the above results we could say there is a tendency that multiple trials approach improves the performance of the gait verification system, under the condition that it is sensitive enough to prevent false acceptances, so that the increase in the false acceptances does not affect the performance. However, once more no absolute conclusions can be made without a deeper investigation and statistical analysis.

(a)

	1 trial		2 trials		3 trials		4 trials		5 trials		6 trials	
	far	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**
CNN	0.10	0.11	0.1	0.1	0.16	0.02	0.2	0.011	0.23	0.006	0.25	0.003
LSTM	0.08	0.16	0.076	0.16	0.12	0.04	0.15	0.016	0.17	0.0078	0.21	0.0047
CNN+LSTM	0.11	0.094	0.11	0.093	0.16	0.027	0.21	0.012	0.24	0.0068	0.27	0.0039

(b)

	1 trial		2 trials		3 trials		4 trials		5 trials		6 trials	
	far	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**
CNN	0.08	0.1	0.12	0.02	0.15	0.01	0.17	0.006	0.19	0.002	0.21	0.0015
LSTM	0.1	0.16	0.15	0.055	0.18	0.023	0.21	0.012	0.23	0.0076	0.25	0.0052
CNN+LSTM	0.076	0.092	0.119	0.03	0.14	0.017	0.16	0.009	0.18	0.005	0.2	0.0031

(c)

	1 trial		2 trials		3 trials		4 trials		5 trials		6 trials	
	far	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**
CNN	0.059	0.098	0.095	0.026	0.12	0.0094	0.14	0.0036	0.158	0.0007	0.174	0.00026
LSTM	0.046	0.09	0.073	0.032	0.089	0.017	0.1	0.01	0.115	0.005	0.128	0.0034
CNN+LSTM	0.059	0.066	0.083	0.026	0.1	0.01	0.11	0.006	0.126	.0036	0.13	0.0026

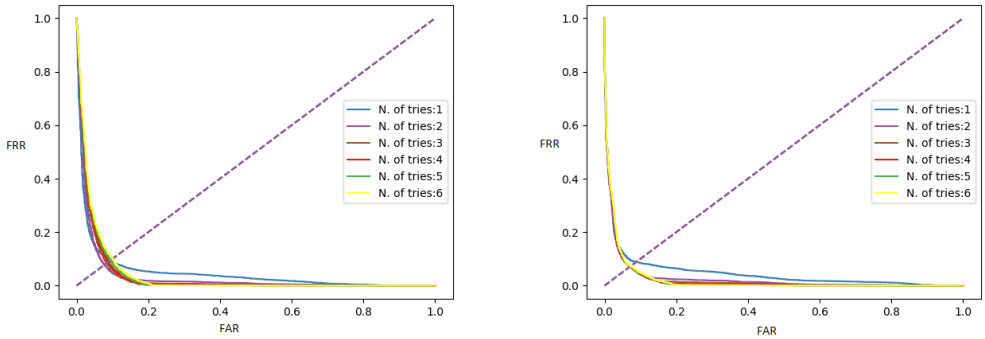
(d)

	1 trial		2 trials		3 trials		4 trials		5 trials		6 trials	
	far	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**
CNN	0.04	0.1	0.065	0.026	0.079	0.01	0.098	0.0034	0.1	0.0013	0.119	0.00026
LSTM	0.035	0.1	0.05	0.031	0.062	0.013	0.072	0.0055	0.078	0.0023	0.084	0.0013
CNN+LSTM	0.038	0.075	0.053	0.03	0.06	0.012	0.067	0.009	0.075	0.0083	0.08	0.008

Table 4.9: FAR and FRR metrics after a number of consecutive trials (1 to 6 trials). Tables a), b), c), d) correspond to gaits of 1, 2, 4 and 8 steps respectively.

Figure 4.8 provides a visual representation of the above results for the case of 2 steps gait and the CNN and the CNN+LSTM model. From the shape of the curves we can argue that the CNN+LSTM model provides better results. In addition, we can see that in both cases the application of multiple trials lowers the FRR values, however FAR seems to increase more for the CNN model than for the CNN+LSTM model, where it appears to have no alterations at all.

In order to observe the behavior of the gait verification models per test user we calculated the FAR and FRR values for each of the 20 test users separately for all trials. Table 4.10 contains the results. We notice that the models can always predict most of the users correctly when 3 or more verification trials are used (really low values for FRR). Nonetheless, there are some users that the models insist to not verify even after 5 or 6 trials. Thus, the number of FRR decreases significantly, however some users cannot be verified by the models (user 6, user 19). A possible reason for this fact might be that during the data collection period more than one people were using the phone or that the users were walking in really different manners during this period.



(a) (b)

Figure 4.8: DET curves for the CNN (a) and the CNN+LSTM (b) models for gait of 2 steps. The figure shows how the curves alter according to the number of verification trials.

	1 trial		2 trials		3 trials		4 trials		5 trials		6 trials	
	far	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**	**far**	**frr**
User 1	0.152	0.0052	0.1947	0.0052	0.2105	0	0.2157	0	0.2263	0	0.2368	0
User 2	0.126	0.0052	0.1684	0	0.1842	0	0.2263	0	0.2473	0	0.2631	0
User 3	0.173	0.0473	0.2263	0	0.2526	0	0.2684	0	0.2789	0	0.3105	0
User 4	0.0052	0.0105	0.0052	0	0.0052	0	0.021	0	0.0421	0	0.0526	0
User 5	0.0315	0.0947	0.0578	0	0.0842	0	0.0842	0	0.0842	0	0.0842	0
User 6	0.1	0.4052	0.1631	0.1578	0.2	0.0736	0.2473	0.0315	0.2789	0.0105	0.3052	0
User 7	0.1894	0.121	0.3263	0.0157	0.3842	0	0.4315	0	0.4631	0	0.5157	0
User 8	0.15789	0.0315	0.2578	0	0.3052	0	0.3473	0	0.3578	0	0.3736	0
User 9	0.021	0.0421	0.0368	0	0.0473	0	0.0526	0	0.0631	0	0.0684	0
User 10	0.1157	0.0157	0.1684	0.0052	0.1947	0.0052	0.221	0	0.2526	0	0.257	0
User 11	0.0473	0.2052	0.0842	0.0473	0.1105	0.0105	0.1263	0	0.1473	0	0.1631	0
User 12	0.0789	0.1052	0.121	0.0105	0.1368	0	0.1578	0	0.1842	0	0.2052	0
User 13	0.021	0.0368	0.0368	0	0.0578	0	0.0578	0	0.0736	0	0.0894	0
User 14	0.0368	0	0.0526	0	0.0578	0	0.0578	0	0.0842	0	0.1	0
User 15	0.0684	0.0421	0.0947	0.0052	0.1315	0	0.1368	0	0.1421	0	0.1526	0
User 16	0.1	0.0526	0.1473	0.0052	0.1631	0	0.1947	0	0.2	0	0.2263	0
User 17	0.0473	0.0526	0.0947	0.0052	0.1105	0	0.1263	0	0.1578	0	0.1789	0
User 18	0.0368	0.2631	0.1	0.0631	0.1473	0.0263	0.1631	0.0052	0.1736	0	0.221	0
User 19	0.1157	0.5157	0.1631	0.2421	0.2105	0.1526	0.2473	0.0842	0.3	0.0421	0.3368	0.0315
User 20	0.0263	0.0578	0.0368	0.0052	0.0684	0	0.1052	0	0.121	0	0.1473	0

Table 4.10: FAR and FRR per test user and per number of trials for CNN model

Another observation is that for many users the values of FAR are high even for 1 verification trial and these values are increasing as expected, as the number of trials increases. However, there are some users who are rarely mistaken for other users (user 4, user 20). A reason for that might be that these users have really peculiar walking styles that the models were able to capture, in contrast with the walking style of most of the rest users. Probably users such as 3, 7 and 1 have similar walking patterns and the models are not able to distinguish among them.

Chapter 5

Discussion

5.1 Summary of findings

From the results obtained by the first and main experiment of the project we conclude that the model with the highest performance depends on the number of gait steps.

In terms of accuracy the CNN+LSTM model for 8 steps of gait had the highest performance (94.08%) in the test set. However, from Figure 4.6, after performing 10-fold cross validation, we observe that for gaits of 1 and 2 steps the best model is the CNN, while for gaits of 8 steps the choice of the model is of less importance.

In terms of FAR and FRR though, we can argue that the values of these metrics are similar for a specific number of steps, regardless of the model we choose. However, they are not low enough and cannot be used commercially. An accurate biometric system, can be used in the real world only if the FAR and FRR values have sufficiently low magnitude (order of -3) [42].

We should, also mention, as seen from the confusion matrices in Figure 4.5, that the suggested approaches are not sensitive enough and cannot capture small differences in the walking patterns of some users. Thus, there are users that are confused for others. It is obvious that such a system would be problematic in the real world. The application of multiple trials did not seem to improve the performance and neither did the usage of low pass filtered data, even though a deeper investigation is required.

Comparison to the baseline paper

In recent years, gait biometric gained a lot of interest and scientists tried to tackle it in different ways, either with video-based data or inertial sensor data.

This work is based on the previous work of Zou et.al [5] in 2018. In our approach we used the same number of layers and nodes, as in the research of Zou et.al, however we tried a siamese architecture.

Our proposed architecture produced better results in the case of the CNN model, 90.74% accuracy instead of 87.72% for the same dataset. However, the siamese LSTM network (86.66% accuracy) was outperformed by the LSTM proposed by Zou (91.70%), while the same holds for the last model as well. Our CNN+LSTM approach provided results of 90.94% test accuracy while the 2018 paper produced results of 93.75% accuracy. This might be due to fact that we did not perform a proper parameter search, suitable for the architecture we suggested.

In addition, in our approach we tried different durations of gait, which improved the accuracy of the models. In our approach with 8 steps of gait we managed to reach an accuracy 94.08%, without going into deep investigations for the parameters.

5.2 Work's impact to the field of biometric verification

During the last 25 years at least, researchers focus on gait analysis and try to build systems that can perform gait identification and verification.

Gait biometric has recently started to be investigated using deep learning and the results are promising. The current project suggested deep neural networks models to tackle the task, which have improved the performance of many other biometric systems. The suggested approach, tries out 3 models, with different architectures and parameter sizes and also investigates the effect of time on the verification task.

The gait biometric community can be benefit by the results, as they can provide an initial idea of what is the more suitable approach, in terms of steps and models, according to the task that needs to be performed. For some applications e.g. health purposes, longer recording period of gait might be more a suitable setting, while for others, such as user authentication while entering a building, shorter periods are much more convenient and care-free for the users. According to the results of the current project when longer recording period are employed, the choice of the model in use is not important. However, for shorter periods CNN outperformed the rest.

These information can be of help to people that investigate the field of gait biometric, especially when applied in different fields.

5.3 Reflections on project's limitations

During the current project we did not perform a proper parameter search, due to time constrains. Thus, we might have not found an optimal parameter settings for the suggested architecture. The training would take hours, even when using GPUs, which was not possible in the context of this project. However, a more suitable parameter setting might have improved the perform ace and produce more accurate results.

Furthermore, data collection was performed by users that were free to use their phones, without being regulated by anyone. This practice has the advantage that the trained systems would be able to verify users while they are walking casually, however we can not be sure, that during the data collection period the devices were only used by the appropriate users. Such a mistake would certainly affect the performance of the models.

Another limitation is the fact that we only had one dataset in our disposal. Due this fact, we were not able to test the models and their generalization ability in depth and we had to restrict in performing 10-fold cross validation instead.

5.4 Reflections of Ethics, Sustainability and Societal aspects

Gait authentication can be employed in authentication systems, surveillance systems and smart environments [43]. It can also appear useful in some Human Activity Recognition (HAR) applications, where it might be important to know the identity of the person performing a specific task, such as health care applications [44]. As the world we live in is becoming more and more connected, the employment of biometric systems and specifically gait biometric will also grow in the near future. Thus, it would be interested to discuss the work presented in this report from an ethical, sustainabily and social perspective.

The ethical concerns around this work are mainly related to the data collection procedures and privacy matters. As it was explained in chapter 3.1, the data used for this project were obtained from a previous research, from people that gave their consent and willingly participated in it. However, in cases where gait authentication is involved with medical applications, some measures should be also taken to provide security. Although this work is not about security concerning the data storage, it is of great importance to isolate them and filter who can access them.

As for the sustainability issues, the main concern focuses on the performance of the gait authentication system. The gait modality allows the advanced functionality of continuous authentication, in contrast to other biometrics like

fingerprint or face recognition, where authentication can only happen once at the login time [44]. That means, that the system will always run in the background gathering data and performing the authentication. Hence, the system should require as little computational resources as possible. In addition, since the gait can be affected by some factors like aging, injury, gaining or losing weight etc [43], the system should stay updated and learn from the always monitoring input data, so that the accuracy does not drop.

When it comes to the social aspects, the main benefit of the project is to study and help the development of a gait authentication system, that will be used widely and improve our everyday lives. Citizens will not need to remember a large number of passwords, therefore they will be more relaxed and feel more secured, as the only identifier they will need is their own walking pattern.

Chapter 6

Conclusions and future work

6.1 Conclusion

In order to tackle the gait verification problem we built 3 neural networks CNN, LSTM, CNN+LSTM and we trained them using 6-dimensional inertial sensor data from smartphones. The main purpose of the project was to investigate what is the minimum period of time for gait monitoring in order to verify users' identity. The results of the first experiment showed that the model with the highest performance depends on the number of gait steps. However, the effect size is rather low especially when 4 or 8 steps of gait were used for training. For longer gait periods the results showed that the models were able to authenticate most of the users.

Even though the values of the FAR and FRR metrics were low (magnitude order of -2 in most cases), they were not sufficiently low for biometric verification systems, where appropriate FAR values should be closer to zero (at least a magnitude order of -3) [42]. The use of low-pass filtered data did not seem to improve the performance, even though a deeper investigation is needed in this direction.

Finally, the application of multiple verification trials managed to slightly improve the performance of the models, but the FAR values increased a lot. The models were not able to capture small differences among users with similar gait patterns even after 3 or 4 trials.

6.2 Future work

During this project period, different approaches were tried, however due to time limitation some things were left out, as mentioned in Discussion chapter.

Initially we should mention that the second and third experiment were additional, and were not investigated thoroughly. Thus, further research and

statistical analysis is needed.

Following, a proper approach in finding the best parameters for the models would be to perform grid search and select the combination of hyper-parameters that provided the best results. Thus, a further and more systematic investigation of the hyper-parameters (number of layers, dropout probability, learning rate, number of epochs) is suggested.

Another important limitation of this project was the existence of only one available dataset, that did not really help us assess how well the models generalize. Thus, building a new dataset to train and test the models can also be a suggestion. In addition, it is always a good practice to create our own data, as we can be more assertive about their quality and thus have a deeper understanding of the behavior of the models.

Bibliography

[1] J. Unar, W. C. Seng, and A. Abbasi, "A review of biometric technology along with trends and prospects", en, *Pattern Recognition*, vol. 47, no. 8, pp. 2673–2688, Aug. 2014, ISSN: 00313203.

[2] J. Fletcher, P. Howard, D. Mody, A. Vyas, and H. Eng, "A Study of Biometric Security Technology Acceptance and Primary Authentication", en, p. 9,

[3] A. K. Sharma, A. Raghuwanshi, and V. K. Sharma, "Biometric System- A Review", en, vol. 6, p. 4, 2015.

[4] S. Liu and M. Silverman, "A practical guide to biometric security technology", en, *IT Professional*, vol. 3, no. 1, pp. 27–32, Feb. 2001, ISSN: 15209202.

[5] Q. Zou, Y. Wang, Y. Zhao, Q. Wang, C. Shen, and Q. Li, "Deep Learning Based Gait Recognition Using Smartphones in the Wild", en, Nov. 2018.

[6] "A Survey on Gait Recognition", en, *ACM Computing Surveys*, vol. 51, no. 5, C. Wan, L. Wang, and V. V. Phoha, Eds., pp. 1–35, Aug. 2018, ISSN: 03600300.

[7] S. Sprager and M. Juric, "Inertial Sensor-Based Gait Recognition: A Review", en, *Sensors*, vol. 15, no. 9, pp. 22 089–22 127, Sep. 2015, ISSN: 1424-8220.

[8] Li-Peng Wang, R. Wolf, Yu Wang, K. Deng, Lichun Zou, R. Davis, and S. Trolier-McKinstry, "Design, fabrication, and measurement of high-sensitivity piezoelectric microelectromechanical systems accelerometers", en, *Journal of Microelectromechanical Systems*, vol. 12, no. 4, pp. 433–439, Aug. 2003, ISSN: 1057-7157.

[9] I. Lee, G. H. Yoon, J. Park, S. Seok, K. Chun, and K.-I. Lee, "Development and analysis of the vertical capacitive accelerometer", en, *Sensors and Actuators A: Physical*, vol. 119, no. 1, pp. 8–18, Mar. 2005, ISSN: 09244247.

[10] M. Liu, "A Study of Mobile Sensing Using Smartphones", en, *International Journal of Distributed Sensor Networks*, vol. 9, no. 3, p. 272 916, Mar. 2013, ISSN: 1550-1477, 1550-1477.

[11] M. Monteiro, C. Cabeza, and A. C. Marti, "Acceleration measurements using smartphone sensors: Dealing with the equivalence principle", en, *Revista Brasileira de Ensino de Física*, vol. 37, no. 1, p. 1303, Mar. 2015, ISSN: 1806-9126.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, ser. Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2016, ISBN: 978-0-262-03561-3.

[13] F. E. Curtis and K. Scheinberg, "Optimization Methods for Supervised Machine Learning: From Linear Models to Deep Learning", en, Jun. 2017.

[14] C. M. Bishop, *Pattern recognition and machine learning*, ser. Information science and statistics. New York: Springer, 2006, ISBN: 978-0-387-31073-2.

[15] B. Polyak, "Some methods of speeding up the convergence of iteration methods", en, *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, Jan. 1964, ISSN: 00415553.

[16] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization", en, p. 39,

[17] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", *arXiv:1412.6980 [cs]*, Dec. 2014.

[18] Y. Hirose, K. Yamashita, and S. Hijiya, "Back-Propagation Algorithm Which Varies the Number of Hidden Units", en, p. 6,

[19] A. Besbes, *Understanding deep Convolutional Neural Networks with a practical use-case in Tensorflow and Keras*, 2017. [Online]. Available: https://ahmedbesbes.com/understanding-deep-convolutional-neural-networks-with-a-practical-use-case-in-tensorflow-and-keras.html.

[20] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, en, ser. Studies in Computational Intelligence. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 385, ISBN: 978-3-642-24796-5 978-3-642-24797-2.

[21] M. S. (Mady), *Chapter 10: DeepNLP - Recurrent Neural Networks with Math.* en, Jan. 2018. [Online]. Available: https://medium.com/deep-math-machine-learning-ai/chapter-10-deepnlp-recurrent-neural-networks-with-math-c4a6846a50a2.

[22] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training Recurrent Neural Networks”, *arXiv:1211.5063 [cs]*, Nov. 2012.

[23] *Understanding LSTM Networks – colah’s blog*.

[24] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory”, eng, *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997, ISSN: 0899-7667.

[25] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, “Signature Verification using a "siamese"time Delay Neural Network”, en, p. 8,

[26] M. Harandi, S. R. Kumar, and R. Nock, “Siamese Networks: A Thing or Two to Know”, en, p. 29,

[27] V. B. Semwal, M. Raj, and G. Nandi, “Biometric gait identification based on a multilayer perceptron”, en, *Robotics and Autonomous Systems*, vol. 65, pp. 65–75, Mar. 2015, ISSN: 09218890.

[28] N. Y. Hammerla, S. Halloran, and T. Plotz, “Deep, Convolutional, and Recurrent Models for Human Activity Recognition Using Wearables”, en, p. 8,

[29] G. Giorgi, F. Martinelli, A. Saracino, and M. Sheikhalishahi, “Try Walking in My Shoes, if You Can: Accurate Gait Recognition Through Deep Learning”, en, in *Computer Safety, Reliability, and Security*, S. Tonetta, E. Schoitsch, and F. Bitsch, Eds., vol. 10489, Cham: Springer International Publishing, 2017, pp. 384–395, ISBN: 978-3-319-66283-1 978-3-319-66284-8.

[30] O. Dehzangi, M. Taherisadr, and R. ChangalVala, “IMU-Based Gait Recognition Using Convolutional Neural Networks and Multi-Sensor Fusion”, en, *Sensors*, vol. 17, no. 12, p. 2735, Nov. 2017, ISSN: 1424-8220.

[31] T. B. Singha, R. K. Nath, and A. V. Narsimhadhan, “Person Recognition using Smartphones’ Accelerometer Data”, en, *arXiv:1711.04689 [cs, eess]*, Nov. 2017.

[32] *Machine learning techniques for gait biometric recognition*. New York, NY: Springer Berlin Heidelberg, 2016, ISBN: 978-3-319-29086-7.

[33] L. Wu, Y. Wang, J. Gao, and X. Li, “Where-and-When to Look: Deep Siamese Attention Networks for Video-based Person Re-identification”, en, *arXiv:1808.01911 [cs]*, Aug. 2018.

[34] D. Chung, K. Tahboub, and E. J. Delp, “A Two Stream Siamese Convolutional Neural Network for Person Re-identification”, en, in *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice: IEEE, Oct. 2017, pp. 1992–2000, ISBN: 978-1-5386-1032-9.

[35] C. Zhang, W. Liu, H. Ma, and H. Fu, "Siamese neural network based gait recognition for human identification", en, in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai: IEEE, Mar. 2016, pp. 2832–2836, ISBN: 978-1-4799-9988-0.

[36] N. Takemura, Y. Makihara, D. Muramatsu, T. Echigo, and Y. Yagi, "On Input/Output Architectures for Convolutional Neural Network-Based Cross-View Gait Recognition", *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2017, ISSN: 1051-8215, 1558-2205.

[37] *PyTorch*, en. [Online]. Available: https://www.pytorch.org.

[38] *Cloud Computing Services*, en.

[39] D. Han, V. Renaudin, and M. Ortiz, "Smartphone based gait analysis using STFT and wavelet transform for indoor navigation", en, in *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Busan, South Korea: IEEE, Oct. 2014, pp. 157–166, ISBN: 978-1-4673-8054-6.

[40] "DET Curves", en, in *Encyclopedia of Biometrics*, S. Z. Li and A. Jain, Eds., Boston, MA: Springer US, 2009, pp. 223–223, ISBN: 978-0-387-73003-5.

[41] E. Conrad, S. Misenar, and J. Feldman, "Domain 1-Chapter 2", en, in *CISSP Study Guide*, Elsevier, 2012, pp. 9–62, ISBN: 978-1-59749-961-3.

[42] P. Biometrics, "White-Paper, Understanding biometric performance evaluation", 1000th ser., vol. 133, p. 4160, 2014.

[43] A. Fathipour Dehkordi, "Reviewing an Authentication Technique Based on Gait Recognition", en, *Journal of Electrical and Electronic Engineering*, vol. 3, no. 2, p. 55, 2015, ISSN: 2329-1613.

[44] R. San-Segundo, R. Cordoba, J. Ferreiros, and L. F. D'Haro-Enríquez, "Frequency features and GMM-UBM approach for gait-based person identification using smartphone inertial signals", en, *Pattern Recognition Letters*, vol. 73, pp. 60–67, Apr. 2016, ISSN: 01678655.

TRITA EECS-EX-2020:64