AUTHENTICATION OF LEGITIMATE USERS OF SMARTPHONES BASED ON APP USAGE SEQUENCES

by

ENRIC PIFERRER TORRES

B.S., Polytechnical University of Catalonia, Barcelona, 2014

A thesis submitted to the Graduate Faculty of the University of Colorado Colorado Springs in partial fulfillment of the requirements for the degree of Master of Engineering in Information Assurance Department of Computer Science

2018
This thesis for the Master of Engineering in Information Assurance degree by

Enric Piferrer Torres

has been approved for the

Department of Computer Science

by

Jugal Kalita, Chair

Jonathan Ventura

Rory Lewis

Abdulaziz Ali Alzubaidi

Date ____________________
Legitimate user authentication based on app usage sequences

Thesis directed by Professor Jugal Kalita

ABSTRACT

Nowadays, it is hard to understand society without smartphones. People may own one or even more of these devices and use them regularly. Consequently, such a trend brings up many security issues, initial authentication being one of the first problems to address when a user tries to access a phone. But what happens once a user has gained initial control and access to the device? Most authentication protocols can prevent intruders from accessing the device at the point of entry, but very few continue this authentication once access has been granted. This research focuses on a continuous authentication based on behavioral biometrics performed once the user has achieved control over the device. Precisely, it considers application usage patterns to evaluate and identify the current user, who may or may not be legitimate. We believe that by analyzing sequences of application usage, we can identify a user and, therefore, prevent non-legitimate access. To do that, we analyze patterns of app usage using Recurrent Neural Networks (RNN), in particular Long Short-Term Memory (LSTM). This methodology allows us to evaluate a considerably long sequence of applications used and identify which was the user and, consequently, whether it is legitimate or not.
DEDICATION

I would like to dedicate this research work to my parents, who have been a very big source of strength and motivation in my life, and specifically during my master’s program and the time spent completing my master’s thesis. Sometimes, children do not value parents as mentors as much as we should, but they did a great job leading me in the right direction, keeping me focused. Lots of love for Jaume and Carmen.
ACKNOWLEDGEMENTS

First of all, I would like to thank the Balsells Foundation and the University of Colorado Colorado Springs for giving me the opportunity to pursue the MEIA through the Balsells Graduate Fellowship. Thank you for helping me achieve this wonderful and fulfilling experience in the United States.

I would like to thank again my parents for always being there, and all my friends in Colorado Springs, who have motivated and encouraged me to keep working hard to reach my goals and finish this research. Especially, I would like to thank Marc Moreno, who has had the patience to listen to my questions and try to share his knowledge in Neural Networks as much as possible. It would not have been achieved without all you guys!

Finally, I would like to thank my advisor, Jugal Kalita, for accepting me as his student, and guide me throughout these months on my master thesis.
# TABLE OF CONTENTS

I. INTRODUCTION .......................................................................................................................... 1

II. BACKGROUND .............................................................................................................................. 7

2.1. Neural Networks (NN) ............................................................................................................. 7

2.2. Recurrent Neural Networks (RNN) .......................................................................................... 11

2.3. Long Short-term Memory (LSTM) ............................................................................................ 13

2.4. Behavioral Biometrics ............................................................................................................ 16

III. ANALYSIS AND RESULTS ........................................................................................................ 20

3.1. Analysis .................................................................................................................................. 20

3.2. The Datasets ............................................................................................................................ 23

3.3. The Model ................................................................................................................................. 25

3.4. Experiments ............................................................................................................................. 27

3.4.1. Multi-class classification ....................................................................................................... 29

3.4.1. Binary classification .............................................................................................................. 33

IV. CONCLUSIONS .......................................................................................................................... 37

V. FUTURE WORK ............................................................................................................................ 40

REFERENCES ................................................................................................................................. 42
LIST OF FIGURES

Figure 1. Most essential apps, survey of 18-34 years olds. ........................................3

Figure 2. A feed-forward neural network with one fully connected hidden layer. 8

Figure 3. General structure of a fully-connected neural network with many outputs..............................................................10

Figure 4. Recurrent neural network over t time steps.............................................11

Figure 5. Recurrent neural network structures for different cases..........................12

Figure 6. The problem of the vanishing gradient with RNNs.................................13

Figure 7. General structure of an LSTM...............................................................14

Figure 8. Performance of a Multi-Class classification, in terms of EER, for different training days and different periods.................................................................18

Figure 9. Performance of a Binary classification, in terms of EER, for different training days and different periods.................................................................18

Figure 10. Sequences re-organized depending on the number of timesteps. The horizontal axis represents the timesteps; the vertical axis represents the number of input sequence.................................................................23

Figure 11. The neural network model.................................................................26
LIST OF TABLES

Table 1. Features for the initial dataset.................................................................5

Table 2. The available datasets. .............................................................................24

Table 3. Multi-class classification showing results in terms of accuracy and EER. ......................................................................................................................29

Table 4. Multi-class classification for short periods of time with the UCCS dataset .........................................................................................................................32

Table 5. Multi-class classification for short periods of time with the UCCS-IF dataset .................................................................................................................33

Table 6. Binary classification results using timesteps = 6 and Input sequence length = 12, in terms of accuracy and EER and with both the UCCS and UCCS-IF datasets ..................................................................................................................33

Table 7. Binary classification for short periods of time with both the UCCS and UCCS-IF datasets. .............................................................................................................35

Table 8. Identifying an intruder with one input sequence against an owner with many input sequences. .........................................................................................36
CHAPTER I

INTRODUCTION

We live surrounded by smartphones. A recent forecast [1] projects 2.53 billion smartphone users worldwide for 2018, United States being the third country with the highest number of users [2]. Unavoidably, the more the number of smartphone users, the more the probability that security issues arise. There are many vulnerabilities in these devices and many attacks target such devices [3], but one of the most pertinent issues that needs to be addressed is authentication. Every time a user wants to use the smartphone, an authentication procedure occurs, whether it is in terms of a PIN code, a pattern, some type of biometric recognition using fingerprint, face or voice, a lock slide or no mechanism at all [4]. But what happens next?

This research focuses on behavioral biometrics, a potentially effective approach to authentication [5]. This method performs continuous authentication once the user has been granted access to the device after a first point-of-entry identification. By continuous authentication, we refer to identification functions that constantly evaluate a user’s behavior, such as screen coordinates touched, keystroke, pressure, gesture, or even the users’ location [4]. By tracking and analyzing a user’s activities using the smartphone, it may be possible to predict the user as owner or intruder. When a user is identified as an intruder, the smartphone should stop its current work as soon as possible and ask for extra authentication before any malicious actions may be performed.
In addition, a core authentication problem is to find the right balance between security and usability [4]. Continuous Authentication, also called Implicit Authentication [6], is a lightweight methodology: it avoids being intrusive until an intruder is suspected. Thus, continuous authentication, if properly administered, has the potential to be effective in providing security without being intrusive and consuming a large amount of resources.

The research follows previous work performed by Alzubaidi [7], where he focused on continuous authentication based on application usage patterns. He created a new dataset of Android users and extracted several non-intrusive features from their app usage, such as day, duration or frequency. He achieved excellent results in terms of user recognition using Weka [8], a library of machine learning algorithms for data mining tasks. The three main achievements of his work are the following:

1. Realizing that calculating an impact factor for each app/user, obtained by multiplying the duration and frequency in each case, and ranking the apps, helps improve the results,

2. Being able to test for legitimate user (binary case) and user recognition (multi-class) for both long and short periods of training time, such as weeks, days, hours and even minutes, and

3. Obtaining better results when identifying a user with a subset of applications used within a short span. The methodologies used are Market-basket analysis [9] and Apriori algorithm [10].
This previous work demonstrates that authentication is possible by thoroughly analyzing users’ daily activity. This is possible since different users operate differently with their devices. According to Falaki [11], there is an immense diversity among users, the number of usage instances per user widely varies, and so does their activities. The article is from 2010, which gives extra credibility to the fact that the more the number of applications in the market, the bigger is the application diversity.

Another interesting recent survey [12] found that most users access around 30 different apps monthly, and about 10 per day from within those 30. Finally, a survey by comScore [13] asked smartphone users to choose the three most essential apps in their daily routines. The high percentages seen in Figure 1 proves that most users tend to access the same applications daily.

![Figure 1. Most essential apps, survey of 18-34 years olds.](image)

We are indeed able to predict users accurately based on their device activity, meaning that most users usually perform the same tasks in characteristic
patterns. But do they perform these tasks in the same order? For example, does someone wake up and always open \textit{app1} first, then \textit{app2}, and so on, in a manner that is distinguishable from other users?

The main goal of this thesis is to verify whether it is possible to identify users from their smartphone activity sequence, understanding a sequence as an ordered set of accesses to applications used by each user, i.e., if \textit{user1} accesses \textit{app1}, \textit{app2}, and \textit{app3} in sequence, \textit{app1-app2-app3} would be considered a sequence from \textit{user1}. To identify users by app usage sequence, we are going to use a widely used and effective type of Neural Networks, Recurrent Neural Networks (RNN) [14]. RNNs are able to take a sequence as input and predict a possible output by analyzing and training on prior items in the sequence as a chain, also called window of units. Specifically, our sequence problem will be evaluated using Long Short-Term Memory (LSTM) [15], a special subclass of RNNs. LSTMs work better with longer sequences, and overcome one issue with RNNs, the vanishing gradient problem, where inputs that came in several time steps before are not considered as important as the more recent ones, although the key for solving the sequence output might have come in many inputs earlier. LSTMs contain memory cells that can keep a state based on every single previous input and can therefore take into account inputs from many timesteps ago in making decisions.

The model presented in this paper will consider and evaluate several different sequence lengths in order to verify that the model can work with longer as well as shorter application sequences. Moreover, both multi-class and binary classification scenarios will be explored to predict which user is manipulating the
device and predict whether the user is the owner or an intruder. We expect to
obtain higher accuracy on the binary case, where only two possible outputs are
possible.

One of the limitations of this research is the lack of large, publicly available
datasets to train, which might bring some overfitting. The experiments discussed
in this thesis use a dataset from [7], which contains application usage data from
25 different users within a span of three months. Each sample or application record
contains the features shown below in Table 1. Further experiments may be
performed with other datasets to validate the results.

Table 1. Features for the initial dataset.

<table>
<thead>
<tr>
<th>FEATURE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>APP</td>
<td>The name of the application accessed</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>Application's category</td>
</tr>
<tr>
<td>APP_OPEN</td>
<td>The exact date, including seconds, when the app was opened</td>
</tr>
<tr>
<td>APP_CLOSE</td>
<td>The exact date when app was closed</td>
</tr>
<tr>
<td>DAY</td>
<td>Day the app was accessed</td>
</tr>
<tr>
<td>MONTH</td>
<td>Month the app was accessed</td>
</tr>
<tr>
<td>DAY_OF_WEEK</td>
<td>From Monday to Sunday, when it was accessed</td>
</tr>
<tr>
<td>DURATION</td>
<td>How long the user accessed</td>
</tr>
<tr>
<td>NORMDURATION</td>
<td>Normalized duration</td>
</tr>
<tr>
<td>OPEN PERIOD</td>
<td>The quarter of the hour when the app was opened, having each hour 4 quarters</td>
</tr>
<tr>
<td>CLOSE PERIOD</td>
<td>The quarter of the hour when the app was closed</td>
</tr>
<tr>
<td>FREQ_OPEN</td>
<td>How often there is user activity within 15 minutes, using information theory</td>
</tr>
<tr>
<td>FREQ_CLOSE</td>
<td>How often there is user activity within 15 minutes, using information theory</td>
</tr>
<tr>
<td>WORK</td>
<td>1 if accessed from Monday to Friday, from 8AM to 5PM; 0 otherwise</td>
</tr>
<tr>
<td>SESSION</td>
<td>The opposite of the above work feature</td>
</tr>
<tr>
<td>WEEKEND</td>
<td>1 if accessed Saturday or Sunday; 0 otherwise</td>
</tr>
<tr>
<td>ID</td>
<td>user's ID</td>
</tr>
</tbody>
</table>
The following chapters discuss the research setup and the experimental results. Chapter 2 explains the main relevant concepts by first introducing the Neural Networks, and after that presents RNNs and LSTMs, by giving examples of cases where they have been used in behavioral biometrics and other fields. Chapter 3 presents the model used in this thesis, discusses the different input structures considered for the experiments, and shows the results obtained for both binary and multi-class classification. Chapter 4 concludes the work done in the research, and Chapter 5 proposes future work for continuous authentication.
CHAPTER II

BACKGROUND

This chapter presents a detailed discussion on the main concepts used in the experiments presented later in the thesis. It provides the background and usefulness of these methodologies, with some practical applications where they have been successful.

2.1. Neural Networks (NN)

Neural networks are biologically inspired models of computation [14] designed to learn from datasets comprised of pairs of input/output given in a first phase of training, to finally be able to predict the output in the second phase of testing, where only the input is provided. In a real neural network, we have many neurons connected to each other, and these connections transfer information from one neuron to the other. Similarly, an artificial neural network consists of input data connected to one or more artificial neurons, forming the hidden layer, connected to the output. Figure 2 shows an example of a basic neural network with three inputs fully connected to the artificial neurons from the hidden layer for predicting an output.
Figure 2. A feed-forward neural network with one fully connected hidden layer.

A general neural network works as follows. The system learns from the input by adjusting weights on the edges, that are updated after each prediction based on how similar the given output is to the actual output of the output cell (or neuron) in the current network. Each cell receives weights from each of the incoming connections, performs a weighted sum of inputs, and compares this sum to a threshold stored in each of the neurons. If the weighted sum of the inputs is bigger than the threshold, then it is said that the cell “fires”, meaning that the summed information is passed through to the outgoing connection. In case the sum is smaller, the information is not passed on. In the output cell then, the system will make a prediction and compare it with the output that has been given.

Most currently designed networks perform computations using a feedforward mechanism [16], where the information travels in only one direction, from the input to the output, as seen in Figure 2. However, the weights within the network are usually updated in a backward direction based on the amount of error at the output using an algorithm called backpropagation [16]. With this algorithm, when the output given to the network is different from the one predicted by the
network, the system goes back from the output to the input cell across all the connections and updates the weights. In a sense, the network tries to modify how individual features of the data can be weighted to better understand it the next time. This is called supervised learning [17], where the neuron has a “teacher” that gives feedback for every single prediction.

This happens for each input data, and all inputs are used for training repeatedly many times. A single presentation of all data points is called an epoch [17]. Before starting the training process, the weights and thresholds are initialized with random values, which make the network start with low prediction accuracy, which increases as the weights are updated. After a certain number of epochs, the learning stops.

Once the network has been trained, the testing phase starts. This time, the system predicts an output for one or more previously unseen inputs using the final updated weights. Ideally, the training and testing accuracy should be similar. However, often the system does better in training than in testing. This is called overfitting [14] and means that the network learns well how to predict output from the input considering the training dataset but fails to find the statistical process of generalization and does not adapt well to new input data, in this case, the testing data. It usually happens when there is not a lot of data available, which is one of the limitations of this research. There are some techniques, presented later in Chapter 3, to decrease this effect.

Finally, it is important to remark that such a network can work with both binary and multi-class problems. This is crucial for this research, providing that we
want to identify whether the current user is the owner of the smartphone or not in a binary fashion, but also predict the identity of the device user from a set of users. Figure 3 shows the structure of a more complex network with several outputs.

![General structure of a fully-connected neural network with many outputs.](image)

**Figure 3.** General structure of a fully-connected neural network with many outputs.

Nowadays, neural networks are used in a huge variety of fields and have become a very important prediction tool. Medicine is probably the field with the most outstanding and essential discoveries. For example, Neural Networks have been used to accurately identify diseases in MRI images [18]. In the field of security, authentication has become more effective, thanks to behavioral biometrics [19] such as face recognition or fingerprint recognition, preventing malicious intruders from accessing others’ devices. More details on this topic are given in Section 2.4.
2.2. Recurrent Neural Networks (RNN)

Feed-forward neural networks work great for pattern recognition, but do not consider the order of the input data. To solve sequence problems where the order matters, a subtype of the artificial neural networks is used, the Recurrent Neural Network [20].

RNNs have a similar structure, where they receive an input, process it, and finally give out an output. However, this output is also fed back together with the next input. Inputs arrive in a stream, in what are called timesteps. Figure 4 clarifies the idea by showing a simple RNN on the left, and the unrolled network on the right side, for a sequence of $t$ inputs.

![Figure 4. Recurrent neural network over $t$ time steps.](image)

There are variations on the architecture of an RNN system, depending on the problem to solve. Figure 5 shows three different types of organization. This research uses the organization shown in the middle called many-to-one, where many inputs are given, and one output is predicted, representing a timestep of three. In a sense, the network takes the last input received at time $t$ along with the previous two inputs at times $t-1$ and $t-2$, and predicts a value based on that sequence. Structures like one-to-many or many-to-many are useful for other...
cases such as given an image, predict a sequence of words, or given a sequence of words in one language, predict a sequence of words in another language. It seems clear though that in all the cases the order is the key to get precise predictions.

![Diagram of RNN structures for different cases.](image)

**Figure 5.** Recurrent neural network structures for different cases.

RNNs have produced great results for short sequences, but when the sequence becomes long (over ten timesteps) the neurons are not able to retain earlier information and lose it, causing bad predictions [21]. This is called the problem of the *vanishing gradient* and it is shown in Figure 6. The network considers the last piece of input data as the most important one, and input information becomes less important as it goes back in time, due to the large number of multiplications of small numbers that takes place from stage to stage. Therefore, the weights from distant data become too small and the network does not have any way to hold data to influence the predicted data. For the network to
be able to predict the right output from distant information, adjustments need to be made.

**Figure 6.** The problem of the vanishing gradient with RNNs.

### 2.3. Long Short-term Memory (LSTM)

LSTMs are a tweaked version of RNNs to fix the problem of forgetting input data from the distant past. The way it does so is by adding “memory” to each of the neurons or cells. Each cell now has three gates that manage the network.

- **Forget gate:** Its main task is removing information that is no longer required for modeling (or is just less important) from the cell state. It is done by taking as input the new input data and the output of the previous cell and outputting a value between 0 and 1 for each of the information items. This indicates which values to keep and which to discard. When a value for some information gets down to 0, it means the information is forgotten. The decision is made by a sigmoid activation function [22].
• **Input gate**: It is responsible for adding the input data to the cell state. It has two processes. First, deciding which values to update using a *sigmoid* function, and then, creating a vector of the new candidate values to add to the state by using a *tanh* activation function [22].

• **Output gate**: It decides what the cell is going to output. It first decides which piece of information from the cell state to consider using a *sigmoid* function. Then, it takes the new information updated by the input and forget gates, applies a *tanh* function, and multiplies it by the result of the *sigmoid* function.

These procedures are performed for every timestep, and are summarized in Figure 7. At every timestep, the cell applies the relevant operations to the data received, i.e., the input at time \( t \) and the previous cell state or memory. After that, it passes the new updated cell state to the next cell, while also predicting an actual output, showed as \( h_t \) in the figure.

![Figure 7. General structure of an LSTM.](image)
An area that uses this method successfully is image captioning [23], where the input is an image, and the output is a sentence with the corresponding label word in it. Yang [24] recently proposed a new method to correctly caption images. Their proposal contains a Convolutional Neural Network (CNN) [25] for the encoding part, where it recognizes objects and associates it with a spatial location, and a one-to-many LSTM architecture for the decoding part using an attention mechanism that associates sub-regions of an image with words or features. The results are outstanding, producing complete sentences that include the objects and the associated location words.

Sometimes it may be interesting and useful to modify the basic architecture of an LSTM network in order to achieve better results or to address computational complexity, as in the case of Sak et al. in [15], where they created their own LSTM architecture for large vocabulary speech recognition. Adding some extra final layers to the general LSTM structure allows them to handle the scalability problem better and find the parameters they need for a proper prediction. They were able to obtain better results than using a general LSTM.

In general, LSTMs perform well, but modifications or extra layers are usually added to achieve better accuracy. In our research, we plan to use a few layers to get more accurate predictions. Although a lot of studies show great results with this methodology, no recent work has focused on smartphone app usage as a sequence to predict legitimate and non-legitimate users. However, LSTMs have been applied to other behavioral biometric applications for authentication, and they are mentioned in the next section.
2.4. Behavioral Biometrics

Security and privacy have become important issues in the era of the Internet and smartphones. This has given rise to the use of biometric technology, providing an extra level of usage safety. Biometrics cover many different topics such as: face recognition, fingerprint, palm-print, iris, gesture, gait, voice, speech or keystroke [26]. These mechanisms form a perfect addition to the traditional and simple identification methods such as PIN code, pattern drawing, or a simple finger swipe.

There are two kinds of biometrics authentication: physiological and behavioral [19]. While physiological biometrics rely more on physical attributes such as face recognition, fingerprint or palm-print, behavioral biometrics identify features of user behavior that remain stable over time during daily activities. Good examples are motion, keystroke or gait.

A positive argument for behavioral biometrics is that they are able to perform lightweight identification tasks without impacting upon devices’ usability. Physiological features are mostly used to extend initial security with additional authentication options that improve their effectiveness. This thesis focuses on what happens after an initial authentication. If an intruder has been able to take control e.g., has gained knowledge of the PIN code, when no further identification control is exercised, he/she may have access to critical and private information such as credit card credentials or may be able to login to the user’s accounts without permission. Recent studies on behavioral biometrics have also focused on continuous authentication after the first point-of-entry to prevent from illegitimate access later.
It is also important to know the evaluation metrics that are commonly used. One obvious metric calculates the accuracy of an authentication mechanism by dividing the number of correctly predicted users by the total number of users tested. Another metric that has been used in behavioral biometrics is the Equal Error Rate (EER), which evaluates the False Acceptance Rate (FAR) and False Rejection Rate (FRR), also called False Positive Rate (FPR) and False Negative Rate (FNR), respectively, and takes the point where they are equal. In other words, FPR corresponds to the percentage of users that are incorrectly accepted as rightful ones, while FNR corresponds to the percentage of owners that are identified as intruders. The goal of any model is to decrease the values of both.

There is some published research using datasets that contain smartphone activity. Fridman et al. [27] collected data from 200 users during a period of 30 days about their texting, application usage, accessed websites and device location. Using a binary classifier, they identified whether it is an intruder or not, and achieved a 5% EER using 60% of the data for the training phase. However, we feel that this data is a bit intrusive, which our research intends to avoid.

Another research effort on user authentication [28] used a dataset provided in [29] to identify legitimate users based on telephone calling, device usage and Bluetooth scanning. The results showed EER of 13.5%, 35.1% and 35.7%, respectively.

Finally, Alzubaidi [7] created his own dataset from the smartphone activities of 25 Android users and performed several experiments to identify a user. He realized that by calculating an impact factor for each user-application pair by
multiplying the app frequency and duration for a specific user, and then ranking the top apps based on the impact factor, he could get better results. He also found the most used application sets for each user and performed experiments for binary/multi-class classification and long/short periods of time. The excellent final results, expressed in terms of EER, are shown in Figure 8 and Figure 9. These results are the starting point for this research to compare further performance of methodologies.

<table>
<thead>
<tr>
<th>Period</th>
<th>All days</th>
<th>UCCS 7 days</th>
<th>1 days</th>
</tr>
</thead>
<tbody>
<tr>
<td>12H</td>
<td>3.9</td>
<td>4.3</td>
<td>6.5</td>
</tr>
<tr>
<td>6H</td>
<td>4.2</td>
<td>4.6</td>
<td>6.9</td>
</tr>
<tr>
<td>1H</td>
<td>4.5</td>
<td>5.4</td>
<td>8.1</td>
</tr>
<tr>
<td>30M</td>
<td>5.66</td>
<td>6.5</td>
<td>10.6</td>
</tr>
<tr>
<td>15M</td>
<td>5.75</td>
<td>6.8</td>
<td>11.3</td>
</tr>
<tr>
<td>5M</td>
<td>6.1</td>
<td>7.7</td>
<td>11.5</td>
</tr>
<tr>
<td>1M</td>
<td>6.5</td>
<td>7.9</td>
<td>12.3</td>
</tr>
</tbody>
</table>

**Figure 8.** Performance of a Multi-Class classification, in terms of EER, for different training days and different periods.

<table>
<thead>
<tr>
<th>Period</th>
<th>All days</th>
<th>UCCS 7 days</th>
<th>1 days</th>
</tr>
</thead>
<tbody>
<tr>
<td>12H</td>
<td>1.13</td>
<td>1.28</td>
<td>2.1</td>
</tr>
<tr>
<td>6H</td>
<td>1.18</td>
<td>1.35</td>
<td>2.57</td>
</tr>
<tr>
<td>1H</td>
<td>1.25</td>
<td>1.72</td>
<td>2.84</td>
</tr>
<tr>
<td>30M</td>
<td>2.5</td>
<td>2.37</td>
<td>4.3</td>
</tr>
<tr>
<td>15M</td>
<td>3.1</td>
<td>4.46</td>
<td>5.57</td>
</tr>
<tr>
<td>5M</td>
<td>3.33</td>
<td>4.79</td>
<td>6.17</td>
</tr>
<tr>
<td>1M</td>
<td>5.65</td>
<td>5.4</td>
<td>7.77</td>
</tr>
</tbody>
</table>

**Figure 9.** Performance of a Binary classification, in terms of EER, for different training days and different periods.
Experiments have recently been performed on behavioral biometrics using LSTM models. Research on key-stroke authentication [30] shows an EER of 13.6% with only two LSTM cells using dataset of 51 typists and a sentence written 400 times.

Face recognition has also been tested [31], with about 50 faces for training and 50 additional faces, from the same user, for testing. A 96% accuracy was achieved when authenticating a user using a binary classifier.

Motion of the user can also be used for authentication. Motion can be tracked by smartphone sensors, i.e., the accelerometer and the gyroscope. Neverova et al. [32] developed an LSTM model with motion data and obtained a 21.13% EER on a binary classification task. Gait recognition has also been used [33] with an 83% accuracy rate.

As seen above, many different methodologies have been used in biometrics to authenticate users to increase smartphone security. After recent successes on data sequence recognition, the use of RNNs has become more prevalent because LSTMs provide great performance with longer timesteps.
CHAPTER III
ANALYSIS AND RESULTS

3.1. Analysis

This thesis uses data from smartphone users’ activity in sequences (after point-of-entry authentication) and predicts the rightful user. These predictions are performed using a sequence methodology, LSTM.

To build the model, we use Keras [34], a high-level neural networks API in Python, on top of TensorFlow [35]. This software provides all the tools and functionalities to build an efficient model to perform our experiments.

When building a sequence model based on LSTMs, the main structure usually includes the input data as a sequence input to one or many LSTM hidden layers. The results obtained out of these layers are used as the input of a Multilayer Perceptron [16]. This hidden layer is responsible for outputting the final class prediction.

The first experiments were performed with very few neurons, one LSTM layer and the whole UCCS dataset. By incrementing the number of neurons, we can see the model increases the overall performance and outputs better predictions. It finally reaches a point where the performance stabilizes. In order to improve the results of the model, we try using stacked LSTMs. By stacking two LSTMs together, the model is able to improve performance at the expense of vastly increasing training time. As we try to find the best overall performance, it is not
convenient to increment the training time. For this reason, stacking more than two LSTM layers is not considered.

After evaluating neurons and layers, there are some other aspects that need to be considered. One of them is the number of epochs. By increasing it, the model is able to learn more accurately the input information. However, it reaches a point where increasing this number does not improve the learning performance and predictions, in fact, the performance may decline.

As the number of epochs and complexity of the network increases, there tends to be a gap between the training and testing performance. This is called overfitting and it is a very common problem. One way to fix this is by using Dropout activation functions [36]. These functions randomly drop some of the units of the network, along with their connections, during the training phase. Another possibility is to stop the training phase once the performance starts to get worse. This method is called Early Stopping [39] and monitors the overall performance of the training phase. Setting up an attribute called patience, the network stops running when a certain number of consecutive worse results are obtained.

Last, but not least, the key to success of the model is the batch size. To make the network remember states and values from earlier timesteps, all the data from a user sequence needs to be included in the batch. There are two design possibilities evaluated in this thesis:

1. Consider one sample of the input sequence for each timestep, meaning that we have as many timesteps as samples in the input sequence (or as input sequence length), or
2. Set a fix number of timesteps, and reorganize the data based on the sequence size. Each input sequence is then split into several input sequences, which are as long as the number of timesteps, adding one new sample for each new input sequence, and removing one old sample.

To further explain the two design possibilities, consider Figure 10. The horizontal axis represents the number of timesteps, while the vertical axis represents the number of input sequences. The yellow cell represents the first option with an input sequence of 30 samples (from sample 1 to sample 30), taking one sample each timestep. The green cells represent the second option. If we take the green cell with the number 12, it represents an input sequence of 12 samples (from sample 1 to sample 12) split into several sequences within a fixed timestep. In this case, we have 7 input sequences within 6 timesteps. To re-organize the data, we can see that each new input sequence has 6 samples. The first one contains samples 1 to 6, but each of the next sequences contain one new sample that replaces the first or oldest sample.

It can be observed that there are many possibilities that could be evaluated, but only a few of them, represented in green, are considered.
Figure 10. Sequences re-organized depending on the number of timesteps. The horizontal axis represents the timesteps; the vertical axis represents the number of input sequence.

### 3.2. The Datasets

The experiments performed with our model use a dataset previously collected by Alzubaidi [7] in 2015, which we call the UCCS dataset. This dataset contains smartphone users' activity, sorted by date, from 25 Android users within three months. There is another available dataset from an MIT research [38], although its data is older and contains less features. This dataset is used to validate our model results, comparing it with the UCCS dataset. Table 2 shows more detailed information about both datasets.
Features *open date* and *close date* are the key to our research since they provide the users’ sequence of accesses. These features are used to sort the samples and create the input sequences, but they are not included in the training or testing dataset. The feature *category* is not used for the experiments either, although it might be considered for future experiments. The main experiments contain a total of 12 features for each app access or sample.

The dataset contains very unbalanced data, since some users have as few as 120 samples, and others with as many as 22599 samples. To reduce the impact of this large difference, the model adds weights to each possible output.

**Table 2.** The available datasets.

<table>
<thead>
<tr>
<th>DATASET</th>
<th>UCCS</th>
<th>MIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAMPLES</td>
<td>90395</td>
<td>186240</td>
</tr>
<tr>
<td>NUMBER OF DIFFERENT APPS</td>
<td>302</td>
<td>126</td>
</tr>
<tr>
<td>NUMBER OF USERS</td>
<td>25</td>
<td>44</td>
</tr>
<tr>
<td>FEATURES</td>
<td>15: app, category, open date, close date, day, month, day of the week, duration, open period, close period, frequency open, frequency close, session, work, weekend</td>
<td>10: app, category, open date, close date, location, day, month, year, duration</td>
</tr>
<tr>
<td>YEAR</td>
<td>2015</td>
<td>2009</td>
</tr>
</tbody>
</table>

Several datasets are created out of the UCCS dataset to perform experiments with different data designs, as mentioned in the previous section. To do that, each user within the dataset contains a number of samples multiple of the input sequence length chosen. This research considers four different input sequence lengths: 12, 18, 24 and 30.
Finally, a new dataset is also generated where we apply the Impact Factor technique seen in [7]. The Impact Factor takes the duration $d$ of an app accessed by a user, the frequency $f$ of accesses of that app by that user, and computes the value of $f*d$ for each pair user,app existing in the dataset. This dataset is referenced as UCCS-IF in the following pages. The apps are sorted based on these results for each user, and only the top apps for each user are used, removing the rest of apps. We want to see if we are able to better identify users by only taking the top k apps for each user as the input sequence of apps.

3.3. The Model

The model design is shown in Figure 11. The input sequence of app accesses is given to two stacked LSTM layers of 256 neurons each. The LSTM prediction output is passed through to the MLP, which contains a Fully-connected layer, with a softmax activation function [37]. This function gives a prediction value for each of the possible users. The highest prediction value is the most probable user.

This model, as mentioned in Section 2, uses a many-to-one mapping modeling from the input to the output, but a many-to-many from the first LSTM layer to the second one. This means the first layer returns the whole sequence of predictions, i.e., the output prediction for each timestep, but the second layer only outputs the final LSTM prediction. The final prediction values come from the MLP.

There are two initial problems when using the UCCS dataset: potential overfitting and unbalanced data. Our model deals well with overfitting thanks to
adding one dropout layer right after each LSTM layer. These dropout layers discard some units randomly to prevent the network from adapting to the training data. In addition to that, the network is generally only trained for 100 epochs. We realized that, in most of the cases, the results did not improve significantly when training over that number of epochs and, in some cases, it even decreases the overall accuracy. Our system adds an Early Stopping method [39] that monitors the training loss of each epoch and stops running the program once it gets N number of consecutive worsen results setting up the patience parameter down to 10.

![Figure 11. The neural network model.](image)

Unbalanced data is managed using weights in the training dataset. The user with more samples in that training dataset has a weight of 1.0, which is the lowest
value. In contrast, the user with less samples is given the highest weight. Specifically, if the user with the most number of samples in the training dataset has \( N \) samples, and the user with the least number of samples has \( M \) samples, the second user’s weight is \( N/M \). This makes the network consider users with low presence in the dataset more importantly when an input from this user is received.

The last layer added is an activation function that eliminates negative values, the ReLu [37], right after each Dropout layer. In general, this layer tends to improve the network performance and does not slow down the training phase.

The model is tested with different input sequence lengths and different number of timesteps to find which is the data design that fits best with our datasets. Each input sequence contains many samples, and each sample contains information from a user app access. Twelve features, shown in Table 2, form this information when testing the UCCS dataset, while only 6 when testing the MIT dataset.

Finally, it is worth saying that Keras allows one to modify the values of several parameters of the LSTMs, such as the activation functions used, the bias applied or some internal dropout. While many tests have been performed with different configurations of these parameters, the default values seem to perform better with our data. Therefore, they are the values used to test the model.

### 3.4. Experiments

This thesis contains experiments for two different use cases. The first use case applies when we are trying to identify the smartphone user as the owner or
intruder. It is called the binary classification case. The second use case applies when we want to identify the user from within many users. It is called the multi-class classification case.

The dataset has been pre-processed and four additional datasets have been created for four different input sequence lengths: 12, 18, 24 and 30. In other words, each user has N samples, and N is multiple of the sequence length. This allows us to perform many different experiments varying the number of timesteps for each of the sizes, and identify the overall performance of our model.

The results shown in this section, unless stated, are validated using k fold cross-validation [40], with k=4. This allows us to train and test the model with different input sequences and in a different order. The dataset is split into 4 subsets, all the same size, 3 of them used in the training phase and one used in the testing phase. This process is repeated 4 times, modifying each time the piece of dataset taken for the testing. Consequently, we use 75% of the data for training the model and 25% of the data for testing it. The model is trained for 100 epochs in each k fold.

Two metrics are used to evaluate the model performance: the accuracy and the EER. The accuracy is already provided by Keras. To calculate the EER, we take the prediction values for all the classes (in our case, users). Each prediction ranges from 0 to 1. Each prediction is then evaluated over a threshold, which ranges from 0 to 1 as well, and takes the values 0, 0.1, 0.2, ..., 1. When the prediction for a class is correct and higher than the threshold, it is considered a true positive. When the prediction is correct but lower than the threshold, the
prediction is considered a false negative. Finally, when the prediction is wrong and over the threshold, it is marked as false positive. By calculating the false positive rate and false negative rate and finding the intersection point, we can find the EER of each class. The final EER value is the average of all the EERs of each class. In contrast with the accuracy, the lowest EER value shows the best performance.

The results of the experiments are shown for both the UCCS and UCCS-IF datasets.

3.4.1. Multi-class classification

This use case tests the prediction performance of our model when we try to identify the smartphone user from within many possible users. Many different structures have been considered when redefining the input sequences, trying to find the one that better adapts to our research case and to the batch size.

Table 3. Multi-class classification showing results in terms of accuracy and EER.

<table>
<thead>
<tr>
<th>Timesteps</th>
<th>Sequence</th>
<th>Accuracy %</th>
<th>EER %</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12</td>
<td>73.83</td>
<td>11.88</td>
</tr>
<tr>
<td>18</td>
<td>18</td>
<td>75.38</td>
<td>13.32</td>
</tr>
<tr>
<td>24</td>
<td>24</td>
<td>75.21</td>
<td>12.11</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>73.33</td>
<td>12.57</td>
</tr>
<tr>
<td><strong>6</strong></td>
<td><strong>12</strong></td>
<td><strong>77.23</strong></td>
<td><strong>8.39</strong></td>
</tr>
<tr>
<td>12</td>
<td>18</td>
<td>76.76</td>
<td>9.95</td>
</tr>
<tr>
<td>18</td>
<td>24</td>
<td>77.01</td>
<td>11.66</td>
</tr>
<tr>
<td>24</td>
<td>30</td>
<td>76.88</td>
<td>10.36</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>75.37</td>
<td>9.02</td>
</tr>
<tr>
<td>6</td>
<td>24</td>
<td>74.12</td>
<td>9.24</td>
</tr>
<tr>
<td>6</td>
<td>30</td>
<td>71.71</td>
<td>10.08</td>
</tr>
<tr>
<td>12</td>
<td>24</td>
<td>75.48</td>
<td>9.92</td>
</tr>
<tr>
<td>12</td>
<td>30</td>
<td>77.10</td>
<td>9.33</td>
</tr>
<tr>
<td>18</td>
<td>30</td>
<td>77.01</td>
<td>10.75</td>
</tr>
<tr>
<td>(MIT) 6</td>
<td>12</td>
<td>58.07</td>
<td>17.55</td>
</tr>
</tbody>
</table>
Table 3 shows the results of the multiple experiments performed with different data structures. The structure that seems to work better with our dataset is having sequences of length 12 re-arranged in 6 timesteps, obtaining up to a 77.23% accuracy with an 8.39% EER. Taking this same best structure, we performed experiments using the Impact Factor technique, and taking the top 10 apps for each user in the input sequences. The results show an improved 80.03% accuracy with a 7.68% EER.

Considering the previous results in terms of accuracy, we can see that having the same value for input sequence length and timesteps obtain results between 73 and 75%. However, if the number of timesteps remain small and the input sequence length becomes bigger, the accuracy prediction tends to decrease, meaning that having many small input sequences does not work well with our model. It is important to add that training time increases too when there are more input sequences, making it an undesirable data design. If we consider the results in terms of EER performance, the worse structure seems to be having the same input length and the same number of timesteps, increasing by 3 to 5% the EER.

The MIT dataset is also tested with the same best structure, 6 timesteps with input sequences of length 12. The results in both accuracy and EER are worse. It makes sense, provided that there are twice as many samples, half of features and less than half of apps. This makes it harder for the network to predict the rightful user.

If we compare the results with [7], we can see that the best EER obtained in our model is almost 1% better than the best EER Alzubaidi achieved with the
UCCS dataset, which is 8.44% with the XGBoost classifier. He did reach a better classification EER with the MIT dataset, with 5.4% compared to our 17.55%.

Another experiment is performed with multi-class classification by taking data within short periods of time, also called spans. Precisely, the experiments consider 4 users’ data in spans of 1, 2 and 4 hours, respectively. The data taken from each user ranges from 9AM to 10AM, from 9AM to 11AM, and from 9AM to 1PM, respectively. This experiment tries to evaluate how well the model works when identifying users within a short period of time. The reason for this experiment is simple: we want to prevent non- legitimate users from gaining access and do it as fast as possible. With this experiment, we verify if we can identify a user with small amount of data.

In contrast with the other experiment, this one has the problem of variable input size. Some users might access their devices more often than others, or in different ranges of time, meaning that some may have many app accesses in one span while others may not have any or just very few. This is managed with a methodology called padding. First, it finds the maximum length of the input sequences, and then zero-pads the shorter sequences. The LSTM is able to realize when the input data is not relevant, and makes predictions only out of the valid input samples.

Those spans that have less than two samples are not considered because no sequence can be given as input. The length of the input sequences grows as we increase the span but, consequently, the difference between the minimum and maximum sequence length increases too. In all three span cases, there are
sequences of two samples, although the maximum length of the input sequences escalates. To manage that, the model modifies the amount of dropout to obtain a better result. This experiment uses the same values for the input sequence length and the number of timesteps.

Table 4 and Table 5 show the results for the UCCS and the UCCS-IF dataset with the Impact Factor computed. We can see that the bigger the span, the better the performance in terms of EER, but not in terms of accuracy. Comparing these with the results obtained in the previous experiment, the accuracy has declined slightly, and EER performance has increased. Applying the Impact factor technique does not seem to increase the overall performance with small amounts of data.

Table 4. Multi-class classification for short periods of time with the UCCS dataset.

<table>
<thead>
<tr>
<th>SPAN</th>
<th>MAXLEN</th>
<th>SAMPLES</th>
<th>SEQUENCES</th>
<th>ACCURACY</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 HOUR</td>
<td>19</td>
<td>456</td>
<td>24</td>
<td>75.00</td>
<td>14.38</td>
</tr>
<tr>
<td>2 HOURS</td>
<td>30</td>
<td>1050</td>
<td>35</td>
<td>68.75</td>
<td>22.08</td>
</tr>
<tr>
<td>4 HOURS</td>
<td>61</td>
<td>3111</td>
<td>51</td>
<td>70.83</td>
<td>11.64</td>
</tr>
</tbody>
</table>

If we compare the results with those obtained in [7], we see that he achieved better results. The best performance was achieved identifying users in 12 hours with a 3.9% EER. The EER achieved with a 1 hour span is 4.5%, lower than the 14.38% obtained in this experiment.
Table 5. Multi-class classification for short periods of time with the UCCS-IF dataset.

<table>
<thead>
<tr>
<th>SPAN</th>
<th>MAXLEN</th>
<th>SAMPLES</th>
<th>SEQUENCES</th>
<th>ACCURACY</th>
<th>EER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 HOUR</td>
<td>19</td>
<td>456</td>
<td>24</td>
<td>75.00</td>
<td>14.87</td>
</tr>
<tr>
<td>2 HOURS</td>
<td>30</td>
<td>1050</td>
<td>35</td>
<td>71.88</td>
<td>16.77</td>
</tr>
<tr>
<td>4 HOURS</td>
<td>55</td>
<td>2805</td>
<td>51</td>
<td>70.83</td>
<td>13.75</td>
</tr>
</tbody>
</table>

3.4.1. Binary classification

The binary experiments try to predict whether the user is the rightful owner or an intruder. Experiments has been performed in a one-against-all fashion for 6 randomly chosen users out of the 25 total users from the UCCS dataset. The results are shown in Table 6. The UCCS dataset has been used with the best data structure from the first multi-class classification experiment, which is 12 as input sequence length and 6 as the number of timesteps. The classification is tested with both the UCCS and UCCS-IF.

Table 6. Binary classification results using timesteps = 6 and Input sequence length = 12, in terms of accuracy and EER and with both the UCCS and UCCS-IF datasets.

<table>
<thead>
<tr>
<th>TARGET</th>
<th>ACCURACY %</th>
<th>EER %</th>
<th>ACCURACY % WITH IF</th>
<th>EER% WITH IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER 0</td>
<td>96.96</td>
<td>7.26</td>
<td>94.87</td>
<td>5.47</td>
</tr>
<tr>
<td>USER 1</td>
<td>98.83</td>
<td>1.10</td>
<td>97.56</td>
<td>5.48</td>
</tr>
<tr>
<td>USER 2</td>
<td>98.51</td>
<td>3.51</td>
<td>99.07</td>
<td>2.78</td>
</tr>
<tr>
<td>USER 3</td>
<td>99.13</td>
<td>0.90</td>
<td>98.67</td>
<td>0.82</td>
</tr>
<tr>
<td>USER 4</td>
<td>96.89</td>
<td>7.4</td>
<td>96.17</td>
<td>6.17</td>
</tr>
<tr>
<td>USER 5</td>
<td>96.79</td>
<td>5.18</td>
<td>96.79</td>
<td>5.42</td>
</tr>
<tr>
<td>AVG</td>
<td>97.85</td>
<td>4.23</td>
<td>97.18</td>
<td>4.36</td>
</tr>
<tr>
<td>(MIT) USER 10</td>
<td>93.69</td>
<td>4.52</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The results show that the accuracy increases from the 77.23% in multi-class classification to an average of 97.85%, more than 20% higher. The EER performance is also improved from the 7.68% in the multi-class classification to the 4.23% for binary. In this experiment, applying the Impact Factor does not improve the results, although they remain very similar. One user from the MIT is also tested to see the performance of the model with another dataset. The EER performance is comparable, but the accuracy is slightly worse.

One interesting fact visible in Table 6 is that the results vary considerably between users, showing an EER ranging from 0.9% to 7.4%. It makes sense that not all the users’ activity can be identified with the same ease. Some users might have some particular behavior while some others might have it similar, making them harder to identify. User 4’s activity seems like a hard one to classify, while user 3’s seems distinctive, making it easier for classification.

Comparing these results with Alzubaidi’s, we see that he obtained better results, with a 0.91% EER using a Rotation Forest classifier, compared to the 4.23% EER achieved with our model.

As done with multi-class classification, we want to know how good is our model with binary classification within short periods of time. The following experiment takes the same spans as in the multi-class, i.e., one, two and four hours, and applies the padding technique to create input sequences of the same length for 4 users for short periods of time.

Table 7 shows the results obtained for both UCCS and UCCS-IF datasets. While the accuracy shows a better overall performance compared to the multi-
class classification one, with about 10% increase, the EER seems to get slightly worse results. Applying the Impact Factor technique does not help to the final results. On the other hand, it reasonably gets worse results than the binary classification using the UCCS dataset.

Comparing these results with Alzubaidi’s, we see that he also achieved better results, with a 1.25% EER using a Rotation Forest classifier, compared to the 13.80% EER reached with our model.

Table 7. Binary classification for short periods of time with both the UCCS and UCCS-IF datasets.

<table>
<thead>
<tr>
<th>SPAN</th>
<th>ACCURACY %</th>
<th>EER %</th>
<th>ACCURACY % WITH IF</th>
<th>EER % WITH IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 HOUR</td>
<td>88.54</td>
<td>13.80</td>
<td>84.38</td>
<td>19.85</td>
</tr>
<tr>
<td>2 HOURS</td>
<td>83.60</td>
<td>22.25</td>
<td>81.25</td>
<td>24.22</td>
</tr>
<tr>
<td>4 HOURS</td>
<td>84.90</td>
<td>15.73</td>
<td>79.69</td>
<td>21.14</td>
</tr>
</tbody>
</table>

A final experiment is performed with binary classification, where one of the two users contains all the input sequences but one. This situation is useful since we do not have a lot of information from an intrusive action when it happens. For this experiment, we take about 1000 samples from one user, simulating the legitimate user, and split the samples into 75% training and 25% testing. To simulate a non-legitimate user, we take 2 samples from another user, one used for training and one used for testing. The test is performed for four different input sequence length: 12, 18, 24 and 30. The information of each test and the results are shown in Table 8. We can see that the intruder is identified in 3 out of the 4 length cases.
Table 8. Identifying an intruder with one input sequence against an owner with many input sequences.

<table>
<thead>
<tr>
<th>INPUT SEQUENCE LENGTH</th>
<th>OWNER SAMPLES</th>
<th>OWNER SEQUENCES</th>
<th>INTRUDER SAMPLES</th>
<th>IDENTIFIED</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>996</td>
<td>83</td>
<td>12</td>
<td>yes</td>
</tr>
<tr>
<td>18</td>
<td>990</td>
<td>55</td>
<td>18</td>
<td>yes</td>
</tr>
<tr>
<td>24</td>
<td>984</td>
<td>41</td>
<td>24</td>
<td>yes</td>
</tr>
<tr>
<td>30</td>
<td>990</td>
<td>33</td>
<td>30</td>
<td>no</td>
</tr>
</tbody>
</table>

This last experiment shows that good predictions might be output with not much training data. The owner contains data for about 48 hours in the training dataset, and 6 hours for the testing. The intruder contains data for about 20, 30, 35 and 60 minutes, respectively, for each input sequence length.
The overall performance of the LSTM model for sequence recognition proposed in this thesis obtained great results for both binary and multi-class classification. The average binary prediction accuracy obtained is 97.85% and the EER is 4.23%. Multi-class classification does a bit worse, with a maximum accuracy of 77.23%, and an EER of 7.68%, which is better than the 8.44% EER obtained by Alzubaidi in [7]. Although millions are the potential users that may get access to the device, most of the continuous authentication researches try to identify users within a small number of classes. This would make sense, since many users could be classified as a type of user, reducing the number of different users. However, it would be interesting to see how the model works with a similar dataset but many more samples and users. On the other hand, the more the users, the more the training time needed and the bigger the complexity of the network.

Nonetheless, one of the biggest concerns in continuous authentication is to be able to identify the intruder within two or more users and prevent him/her from keep controlling the device. This is equivalent to the binary classification, and the model did a very good job with that.

The model also shows good performance when identifying users within a short period of time, with an 88.54% accuracy and 13.80% EER for binary classification in a 1 hour span.
The experiments also proved that the model is able to adapt to distinct data structures, although it seems to work better for smaller layouts.

Comparing the model performance with some other researches mentioned in Section 2, we see that it indeed upgraded existing behavioral biometrics results. We can observe that it achieved better results than Neverova’s work [32], where she identified users with a binary classifier using LSTM with motion data, obtaining a 21.13% EER. Fridman [27] used a similar approach and tried to identify users from their activity on texting, app usage, websites accessed and location, achieving a 5% EER, slightly higher than our 4.23% EER. Li [28] tried to classify users using neural networks and their activity on calls, device usage and Bluetooth performance, achieving a 13.5% EER using a binary classifier. Correa [31] also used LSTM to identify users using face recognition, and achieved a 96% accuracy with a binary classification.

However, this research does not provide a way to solve problems with outliers. For example, when a user is on holidays, his/her smartphone activity might be different from the regular activity, making it hard to control. Some users also give access to many family members to the device, and each of them use it differently. In these situations, our model would identify (or try to) the activity as non-legitimate, and the device should act consequently. To manage these situations out of the daily routine, further decisions must be made when classifying users’ activity, but it is out of the scope of this thesis.

Finally, it seems clear that further research should be done in this topic because it has proven to identify users very accurately. This methodology would
provide extra security to device users, who are every day surrounded by more technological threats.

The contributions of this thesis are the following:

1. Provide the first research that identifies users from their device app activity information, specifically, the sequence of access of these apps, with a surging sequence methodology, the LSTMs, with great success.

2. Apply a previously successful technique, the Impact Factor, to the datasets, and display similar and, in the case of multi-class classification, better results than in those experiments, with a 7.68% EER multi-class classification with the UCCS dataset.
CHAPTER V

FUTURE WORK

One of the main directions to explore to improve the results is trying new network architectures. The model used in this thesis is simple but efficient. However, more layers or a different organization of them could improve the obtained results. Some LSTM researchers use the Attention mechanism [41] to pay selective attention to the inputs and relate them with the outputs. Another possibility would come from testing and exploring the different parameters from LSTMs with Keras, such as feeding the input sequence from first to last sample, and from last to first sample.

One possible drawback of making the network more complex is incrementing training time. If more layers are added to the network, better overall results might be obtained at the expense of longer training time. It is not convenient to have a system that takes longer to identify the intruder. This could make the intruder to gain access to the device for extended time before the model realizes a non-legitimate user is using the device.

While trying to keep the training time small, it would also be necessary to test the network with larger datasets that contain more users and more samples. This would simulate a closer situation to the reality, where millions of users could get control over the device.

Feature selection on the dataset might be necessary to improve the results too. In the experiments performed, twelve available features are used, but it is not
clear all of them are necessary to better identify users. With less but more attaching information, training time would be reduced, while better results might be achieved.

To conclude, the experiments have been performed with the whole UCCS dataset but also applying the Impact Factor. However, Alzubaidi accomplished great results implementing this technique but applying it together with some other methodologies. The experiments should try to mimic those procedures, and apply them to the model to see if it indeed enhances previous results.
REFERENCES


