T.C.

ALTINBAŞ UNIVERSITY

Electrical and Computer Engineering

PATTERN-BASED BEHAVIORAL BIOMETRIC AUTHENTICATION USING ARTIFICIAL NEURAL NETWORKS

Ali Saad Hussein ALNASER

Master Thesis

Supervisor

Prof. Dr. Osman N. UÇAN

Istanbul (2019)

PATTERN-BASED BEHAVIORAL BIOMETRIC AUTHENTICATION USING ARTIFICIAL NEURAL NETWORKS

by

Ali Saad Hussein ALNASER

Electrical and Computer Engineering

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2019

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

____

Asst. Prof. Dr. Osman N. UÇAN

Supervisor

Examining Committee Members

Academic Title Name SURNAME Faculty, University ____________________

Academic Title Name SURNAME Faculty, University ____________________

Academic Title Name SURNAME Faculty, University ____________________

I certify that this thesis satisfies all the requirements as a thesis for the degree of ....

____

Academic Title Name SURNAME

Head of Department

____

Academic Title Name SURNAME

Director

Approval Date of Graduate School of

Science and Engineering: ____/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Ali Saad Hussein ALNASER

DEDICATION

I am dedicating this thesis to my friends and family who mean so much to me. I want to thank everyone who has helped me to achieve and continued to support me throughout. I am very grateful and appreciative of all you. To my parents who have shown me endless love and courage to further my future. To my sister Dania, who has given me strength with love to be the role model towards her future. To my beloved sister Haneen, who is no longer with us but always in our hearts. I will always love you and miss you dearly, hoping to make you proud. To her beautiful son Ayad. Every possibility is achievable with hard work, focus, and support. I am very thankful from my heart to every one of you.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude Prof. Dr. Osman Nuri UCAN for all the knowledge and support he provided during my study for the Master Degree and throughout the work to complete this thesis. You have made my dream come true.

ABSTRACT

PATTERN-BASED BEHAVIORAL BIOMETRIC AUTHENTICATION USING ARTIFICIAL NEURAL NETWORKS

Ali Saad Hussein ALNASER,

M.Sc., Electrical and Computer Engineering, Altınbaş University

Supervisor: Prof. Dr. Osman N. UÇAN

Date: April/2019

Pages: 58

The use of smart devices to collect and store sensitive and private information has been increasing exponentially in recent years, according to the rapid progress in the electronics industry and the digital era. Different protection schemes are being proposed to protect these data from being revealed to unauthorized users that may gain physical access to the device. Some of these schemes have become obsolete, according to their sensitivity toward simple attacks or the development in the attacking strategies adopted by the intruders. Thus, more attention is being applied to the use of biometric features to recognize and authenticate users, according to the difficulty of replicating such features. However, the collection and storage of physiological biometric features can pose threats to the privacy of the users, which is the main reason behind not favoring such authentication schemes. A more privacy-conserving alternative has been widely employed in authentication schemes, which relies on behavioral features instead of the physiological. The collection and storage of behavioral biometric do not pose threats toward the privacy of the user, hence, users can adopt such schemes without worrying about their privacy. However, the robustness of the behavioral biometrics is lower than physiological, which imposes limitations to the performance of such schemes. Thus, in this study, Artificial Neural Networks (ANNs) are employed to handle variations in the behavioral biometrics and produce more robust descriptors per each user. The proposed method is designed to collect behavioral biometrics during the drawing of the graphical secret pattern that is widely used to protect smart devices with touch-sensitive screens. The generated fixed-size descriptor contains 128 values that describe the unique way each user draws a pattern, so that, even when the same pattern is used, different descriptors are generated for different users. Then, by measuring the Euclidean distance between the descriptor stored as a template and the one collected from the attempt, the user can be recognized and authenticated if the distance is lower than a threshold value. A Convolutional Neural Network (CNN), Long- Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) neural networks are evaluated for the proposed system. The results show that the descriptors generated by the GRU neural network have produced the highest performance of the proposed system, with only 6.91% Equal Error Rate (EER).

Keywords: Security; Biometric Authentication; Behavioral Biometrics; Artificial Neural Networks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

RFID	: Radio Frequency Identifier
PIN	: Personal Identification Number
EER	: Equal Error Rate
FAR	: False Acceptance Rate
FRR	: False Rejection Rate
ANN	: Artificial Neural Network
CE	: Cross Entropy
ReLU	: Rectified Linear Unit
CNN	: Convolutional Neural Network
RNN	: Recurrent Neural Network
LST	: Long- Short-Term Memory
GRU	: Gated Recurrent Unit
ROC	: Receiver Operating Characteristic
DTW	: Dynamic Time Warping
DL	: Deep Learning
ML	: Machine Learning
GPU	: Graphical Processing Unit
GUI	: Graphical User Interface

1. INTRODUCTION

The use of smart devices of different types has been growing exponentially in recent years, according to the proposal of the digital era. Several private and sensitive information is being stored and communicated, digitally, using these devices, according to the wide range of applications they are being used to access [1-3]. Accordingly, concerns about the security of such information are being the concern of many researchers, so that, any unauthorized access is denied and the information is protected. However, attacking techniques are also developing rapidly, which allows attackers to gain access to these devices. Thus, more complicated authentication methods are being proposed and employed to protect these devices [4, 5].

The aim of using an authentication method is to distinguish intruders and deny their access to the information on the device, even when they gain physical access to it. In other words, only the legitimate user is allowed to access such information. Thus, an input is required from the user to be used to recognize legibility and make the appropriate decision. Such input can be a predefined secret, textual or graphical, an identifying object, such as tokens and Radio Frequency Identifier (RFID), or using biometric features, which can be physiological or behavioral [6, 7]. However, shoulder surfing attack, which occurs when an intruder spies on the legitimate user during the authentication process to gain the secret used for authentication, is one of the simple attacks that can be executed against secret-based methods. Moreover, the use of easy-to-remember secrets is a tendency among users, hence, such secrets become easy to predict. Thus, secret-based authentication has become obsolete and rarely used to protect any sensitive information or systems [8].

Additionally, authentication systems that rely on objects that users have to recognize them can be easily attacked by simply acquiring that object, so that, the intruder can authenticate as the legitimate user. These vulnerabilities to simple attacks have encouraged the use of biometric authentication to protect systems and information. Depending on the nature of the biological features collected from the user, two types of biometric authentication system exist, physiological and behavioral. When the biometric features represent the physical characteristics of a certain body part of the user, such as the iris or fingerprint, the system is denoted as physiological. Moreover, when these features are collected from the behavior of the user while the execution of a certain task, it is denoted as behavioral [9, 10].

As the features that are used to recognize the user are extracted from the biometrics of that user, the possibility of gaining access to the system by an intruder is significantly lower than in other types of authentication systems, even if the intruder recognizes the type of the features being collected. This difficulty is according to the uniqueness of such features, in both physiological and behavioral biometrics, which makes the reproduction of such features significantly more difficult. Thus, systems and information protected using biometric authentication systems are more secure than those protected using the secret-based or object-based techniques. However, despite the robustness of physiological features, most of the users avoid the use of such features according to concerns about their privacy. Thus, more emphasis on behavioral biometric features has been applied in recent years [4].

Despite the good security of the behavioral biometric authentication systems, behavioral features collected from the users are less robust the physiological ones. A set of features is considered robust when the same values are extracted from the user every time the user authenticates into the system. According to the different situations and environments that the user can be when authenticating into the system, some of the features collected from the behavior of the user may vary, which is not the case in physiological biometric features. Thus, it is important to consider variation in the behavioral biometric features and rely on the most robust features in the collected data [11].

According to the popularity of devices with touch-sensitive screens, several methods have been proposed to achieve behavioral biometric authentication based on features collected from these screens. Nguyen et. al. [12] propose a behavioral biometric authentication system that uses handwritten Personal Identification Numbers (PIN) over the touch-sensitive screens, instead of using numbered buttons to enter these numbers. Authenticating or recognized individuals based on their handwriting has been widely employed in different applications, some of them are not related to smart devices, and has shown good recognition rate. Thus, the method has shown good security measure with 4.48% Equal Error Rate (EER), i.e. 4.48% False Acceptance Rate (FAR) and 4.48% False Rejection Rate

(FRR). However, the usability of this method can be very low according to the long time required to draw the number of the PIN, one by one, instead of tapping the numbered buttons. Moreover, most of the users of smart devices with touch-sensitive screens use graphical pattern-based secrets to authenticate into their devices, 32.5% according to Nathan et. al. [13].

1.1 PROBLEM DEFINITION

Information is being stored on smart devices with touch-sensitive screens, such as smartphones, tablets and computers, are becoming of increasing importance according to the rapid growth of the digital era. Protecting such information is mandatory to deny intruders who may gain physical access to such devices from accessing the digital information stored on it. Several types of authentication system are being used to recognize and verify the legitimate user of the device before allowing access to its services and information. However, with the rapid development of attacking methods and vulnerabilities in earlier methods has encouraged the use of biometric authentication. Concerns about collecting and storing physiological biometric features have been rising from users as such information is considered personal and private. Thus, the use of behavioral biometric features has risen in recent years, despite the lower robustness of behavioral biometric features. Verifying user based on their handwriting has achieved good security measures in [13] but the usability of such a system is low according to the long time required to input the PIN. Graphical pattern-based secrets are widely used by users, which has encouraged the development of behavioral biometric authentication systems that rely on features collected from the touch-sensitive screen of the device during the input of the pattern [14]. However, the security measures of the method show poor performance, with 19% FRR and 21% FAR, which are caused by the variation in the behavioral biometric features collected from the touch-sensitive screen.

1.2 AIM OF THE THESIS

To improve the security measures of the behavioral biometric authentication system based on biometric features collected from the touch-sensitive screen while inputting the graphical pattern-based secret, the aim of this thesis is to use machine learning techniques that can handle variation in the collected biometric features. The proposed method uses artificial neural networks to produce a more robust descriptor that can be used to measure the similarity between the features stored in the template stored for the legitimate user of the device and those collected from the current authentication attempt. Artificial neural networks have shown the ability to handle variations in the inputs and can significantly improve the performance of the system while simplifying the computations to measure the similarity between the template and the attempt.

1.3 THESIS LAYOUT

The structure of the thesis following this chapter is organized as:

- The literature regarding artificial neural networks and behavioral biometric authentication is reviewed in Chapter Two.
- The descriptor generation approach and similarity measurement approach proposed in this study are described in Chapter Three.
- The security of the proposed method is evaluated through a set of experiments that are illustrated in Chapter Four.
- The performance of the method is discussed, to select the approach with the best performance, and compare it to similar methods in the literature in Chapter Five.
- Conclusion and future work are summarized in Chapter Six.

2. LITERATURE REVIEW

The purpose of using an authentication system is to distinguish the legitimate user of the device and allow their access to the services and information stored on the device while denying the access of any other users. Such a system requires a template that defines the secret or features, of the legitimate user, that can be used to verify and validate any future inputs collected from users [15]. Depending on the type of the input collected by the authentication system, the stored template and collected information may require different types of processing, so that, the usability of the system is maintained while improving its security.

2.1 USERS AUTHENTICATION

Basically, users can be authenticated into the system using three main types of information, which are [16]:

- **Something you know:** the input collected from the user in this type of authentication systems represents a predefined secret that, supposedly, only the legitimate user knows. This secret is stored on the device and is required from the user upon authentication, so that, the inputted secret is validated against the stored one to make the appropriate authentication decision, i.e. the user is allowed access if secrets match. Thus, these methods are also known as secret-based methods, where different types of secrets are being used, such as graphical patterns or PINs **[17]**. However, attacks as simple as shoulder surfing, dictionary predictions and brute force can be executed successfully against this type of authentication [18].
- **Something you have:** Instead of using predefined secrets, objects of known characteristics are used in this type of authentication to prove the identity of the user. The use of RFID or ATM cards are examples of such type authentication. However, losing the authentication object can cause the rejection of the legitimate user's access to the system, while acquiring it by an intruder may gain them access to the system [19]. This type of authentication is rarely used

with smart devices, as these devices are widely used and the need to maintain the physical persistence of an object every time the user uses the device limits the usability of methods in this category.

- **Something you are:** This type of authentication relies on identifying individuals using characteristics collected directly from the user, so that, such information cannot be lost by the legitimate user or replicated by an intruder. Methods in this category are also known as biometric authentication methods, where characters used by these methods can be collected from the face, iris or handwriting of the user [19, 20]. Furthermore, these methods can be categorized into physiological and behavioral, depending on the type of biometric features collected from the user.

2.1.1 Graphical Pattern-Based Authentication

This type of secret-based authentication uses a predefined set of dots in a certain layout, shown in Figure 2.1, to define a secret graph that is used to authenticate the user [21]. The secret is defined as the sequence of the dots the user passes through in a single touch, i.e. without removing their finger from the touch-sensitive screen. The sequence starts as soon as the first dot is touched and terminated when the finger of the user is lifted from the screen. This sequence is compared to the sequence stored for the legitimate user and the user is authenticated if they match. This type of authentication is widely used among the smart devices' users according to its ease of use and better security, compared to other secret-based methods [13]. This better security is according to the resistance of pattern-based authentication to several attacks that can be executed against PIN and passphrases, such as dictionary-based guessing, keyloggers and spyware [22].

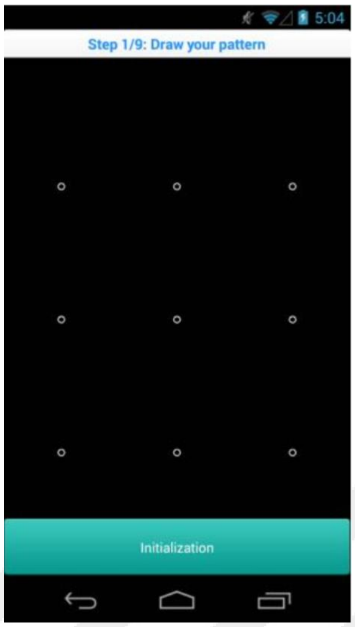


Figure 2.1: Layout of the pattern inputting interface [21].

2.1.2 Behavioral Biometric Authentication

Humans have unique behavior when executing similar tasks, so that, the same task is executed in a unique way by each individual. Behavioral biometrics are features that describe this unique behavior, so that, they can be used to distinguish different individuals. However, the possibility of collecting the exact same behavior from the same individual every time that task is executed is very low, according to the different environments and condition the individual may be in during the execution of that task [23]. Thus, unlike the secret-based methods, collecting data identical to the template stored on the device for the legitimate user cannot be achieved, which disables the use of direct matching techniques to authenticate the user. Moreover, using a wider range of similarity between the collected features and the template can impose limitation in the security of the authentication system, as an intruder that can reproduce features with similarity in the defined range can gain access to the device [24]. To overcome such limitation, machine learning techniques are widely used in behavioral biometric authentication. Such techniques provide more flexibility in measuring the similarity or predicting the authenticity of the user, so that, variation in the collected features and the template can be handled [9, 10].

2.2 ARTIFICIAL NEURAL NETWORKS

Inspired by humans' brains, artificial neurons are the basic units of an Artificial Neural Network (ANN). These neurons are distributed over multiple layers, where the distribution of the neurons in the layer defines its operations and the position of the layer in the network defines its type. Each neuron collects its inputs from the previous layer, i.e. the outputs of the neurons in the previous layer. As shown in Figure 2.2, the output of the neuron is calculated by multiplying each input with a corresponding value, known as the weight, before summed up and passed through an activation function. This nonlinearity provided by the activation function allows the detection of more complex features, as the margins of the values that form the feature can be nonlinear. Moreover, to adjust the positioning of these margins in a more accurate way, an additional value, known as the bias value, is added to the inputs of the neurons, where the value of the bias is adjusted by the neural network depending on the need to position the margins of the detected feature [25, 26].

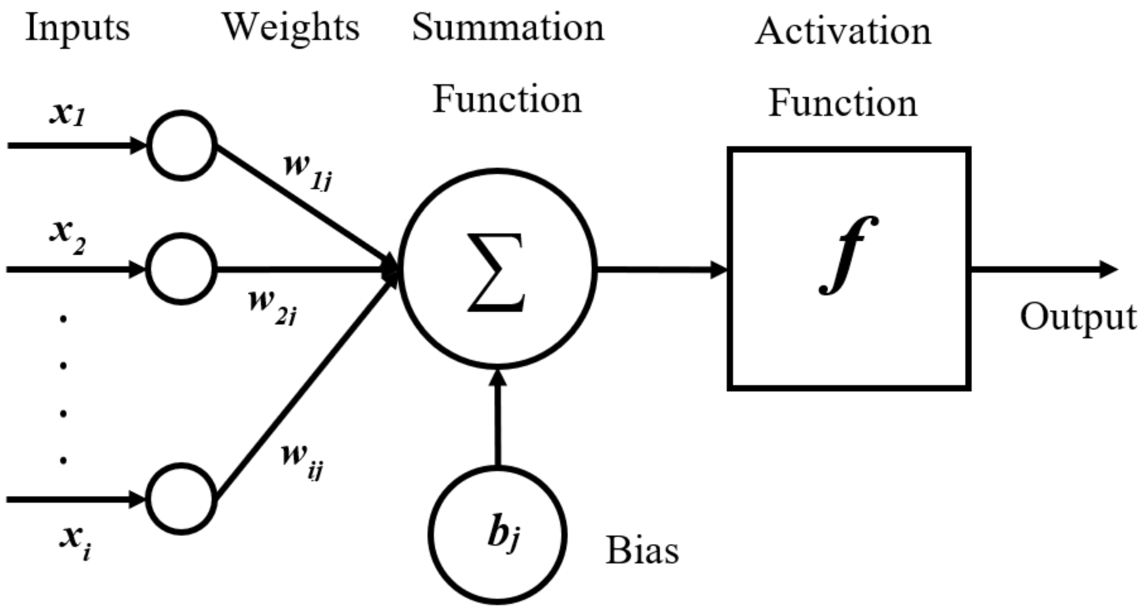


Figure 2.2: Illustration of the computations inside an artificial neuron [25].

Two types of computations are executed in any neural network, forward and backward passes. During training, both passes are required to update the weights and biases values, while when a prediction is required, only the forward pass is executed [27]. In the forward pass, the execution of the computations is initiated from the input layer, where the output of each neuron in a certain layer is calculated and inputted to those in the next layer until the output of the network is calculated. The modification of the weights and biases values is known as the training of the neural network, where the optimal value per each weight and bias is calculated to achieve the required output. Several types of optimizers can be used to update these values depending on the difference between the output of the neural network and the values required to be outputted from it. Mainly, these optimizers are derived from the gradient descent algorithm, which relies on the rate of change of the output per each value to optimize it [28-30]. The new value of each parameter is selected as shown in Equation 2.1.

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \tag{2.1}$$

As the formula shows, the difference between the required output and the prediction of the neural network is multiplied by the derivative of the difference with respect to the parameter being updated. As this value represents the direction to maximize the difference, it is subtracted from the current value in that parameter, so that, the difference is reduced. However, to avoid large updates that may reduce the efficiency of finding the global minimum, a learning rate α is used to damp these changes [31]. To calculate the difference between the predictions and the required values, different types of functions are used, depending on the type of the output and task required from the neural network, as used to summarize the differences into a single value. Cross Entropy (CE) loss function is widely used in neural networks that are implemented for classification problems [32], which has the formula shown in Equation (2.2).

$$CE = -\sum_{i}^{C} t_i \log\left(f(s)_i\right) \tag{2.2}$$

Moreover, the selection of the activation function of the neuron depends on the position of the layer that the neuron belongs to and the performance of the activation function. Despite the existence of several types of activation function, as shown in Figure 2.3, significant improvement of the performance of the neural network is noticed when the Rectified Linear Unit (ReLU) is used as the activation function [33]. However, the task required from the neural network determines the type of the activation function used for the output layer's neurons. For classification tasks, the SoftMax function is used, where the summation of the outputs of the neurons adds up to the value of one and each value reflects the input's probability to be in the class correspondent to that neuron [34, 35]. The formula of the SoftMax function is shown in Equation 2.3.

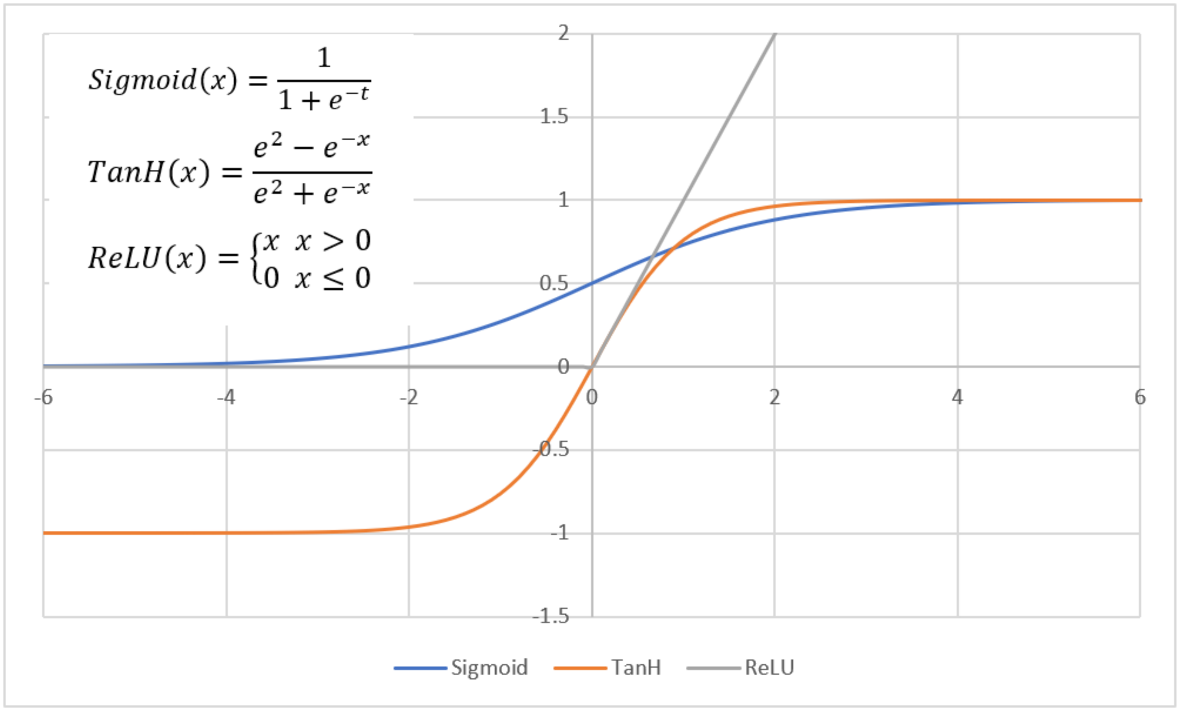


Figure 2.3: Outputs of the Sigmoid, Hyperbolic Tangent and ReLU activation functions.

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \tag{2.3}$$

2.2.1 Convolutional Neural Network (CNN)

Comvultional layers in an artificial neural network consist of multi-dimensional filters that are convoluted over the input of the layer, depending on the dimensions of the input. Each filter contains the weights of a single neuron in the convolutional layer, so that, this neuron gains the ability of detecting multi-dimensional local features. Based on the values of the weights in the filter, the neuron becomes capable of measuring the similarity between the pattern in the filter and the feature it recognizes. As the neurons in deeper layers combine the outputs of those in the previous one, the complexity of the detected feature can be increased by adding more convolutional layers, so that, the feature in a layer represents a combination of those in the previous. Depending on the defined convolution strategy, the output of the neuron has the same number of dimensions of its input but the size of each dimension can be different [36, 37].

The number of values per each dimension that are skipped by the filter per each convolution is known as the strides. Larger strides values reduce the size of the output from the neuron but can cause the loss of detecting features in the skipped values. Thus, pooling layers are used to reduce the dimensionality of the output without producing significant information loss [38]. Pooling layers also use multi-dimensional filters to summarize the values in that filter into a single value. The output values are calculated based on the type of pooling executed in the layer, where Max-Pooling is one of the widely used pooling layers. As shown in Figure 2.4, the Max-Pooling filter outputs the maximum value in the filter to summarize all the values in that filter [38, 39].

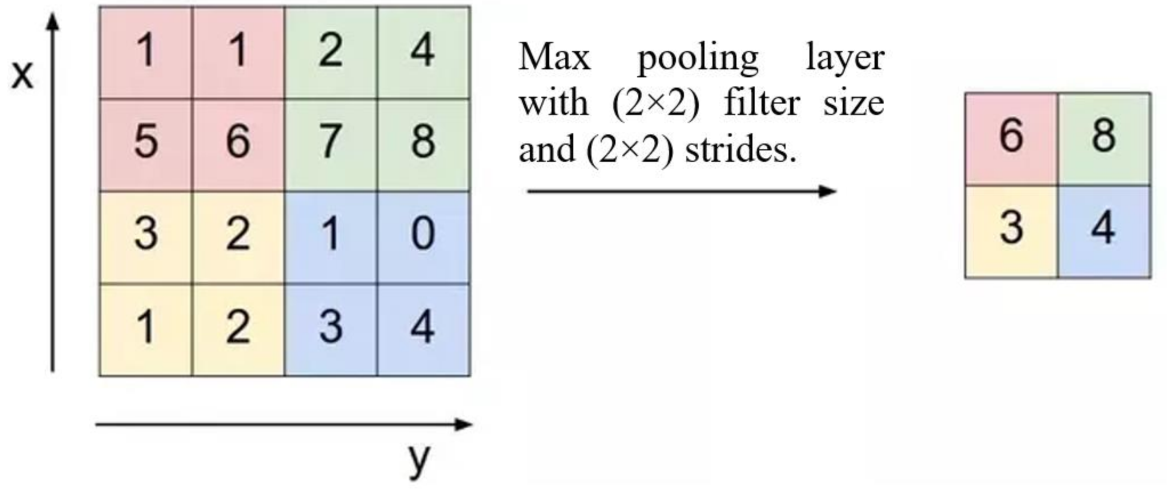


Figure 2.4: Output of Max-Pooling filter.

2.2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are proposed to handle data that have features relevant to time, mostly time-series. The values are provided in a timely manner, where the position of the value in the series depends on the time it occurs, with respect to the other values in the same series. Thus, this type of neural networks handles two-dimensional inputs, where the size of one of the dimensions equals to the number of the features and the size of the second one is the number of values in each series. As shown in Figure 2.5, suppose a weight value *f* is used to adjust the value of the output from the tuple previous to the current tuple positioned at *t*. During the computations of the output of the neuron at *t*, the output *h* from *t-1* is included after being weighted using *f*. The output at this *t* tuple is also weighted using *f* and included with the inputs *x* of the next tuple at *t+1*. This process is repeated until all the tuples in the input set are processed [40, 41].

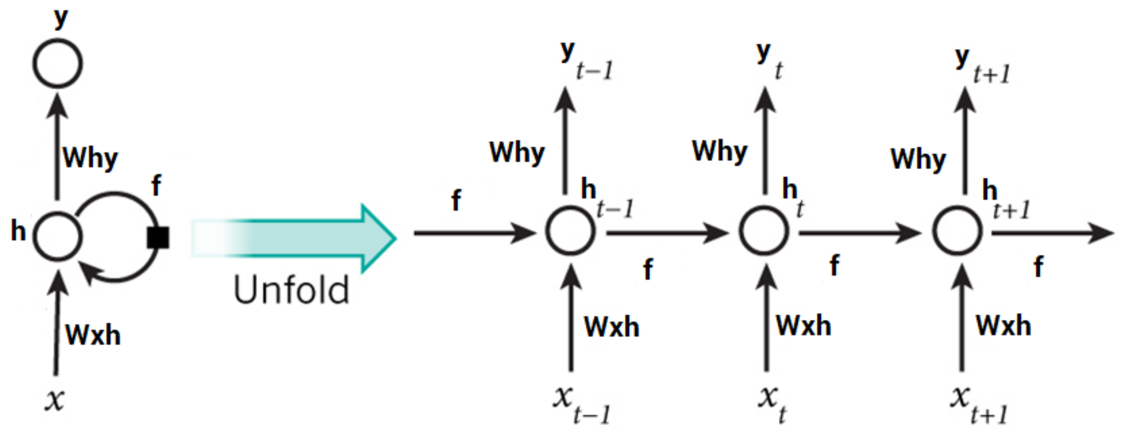


Figure 2.5: Computations in an RNN neuron.

However, this basic RNN topology suffers from the exploding or vanishing gradient problem, where the weights values become extremely high or low to compute the output of the network. This problem has imposed the need to modify such topology, where the Long-Short-Term Memory (LSTM) neural network has proposed the required solution [42]. As shown in Figure 2.6, LSTM uses a set of gates to control the flow of data in the neuron, where each gate is controlled by a separate neural network. The neural network net_c is the input network that receives the values from the external domain and calculates the outputs depending on its weights. Another network net_{in} receives a copy of these inputs in order to control the gate that defines the flow of the output from net_c, through the input gate value y^{in}. The effect of the previous output is adjusted using the forget gate values y^{ϕ}, which is controlled using net_{ϕ}. This output S_c is squashed using an activation function before being adjusted using the values y^{out} acquired from the output gate, which is controlled using net_{out} that calculates the values of the gate using the outputs collected from the previous time instance. As each gate is controlled using a different neural network, the weights of each neural network are updated during the training of the networks, so that, the appropriate decision is made based on the input values of the current time instance and the outputs collected from the previous ones [43].

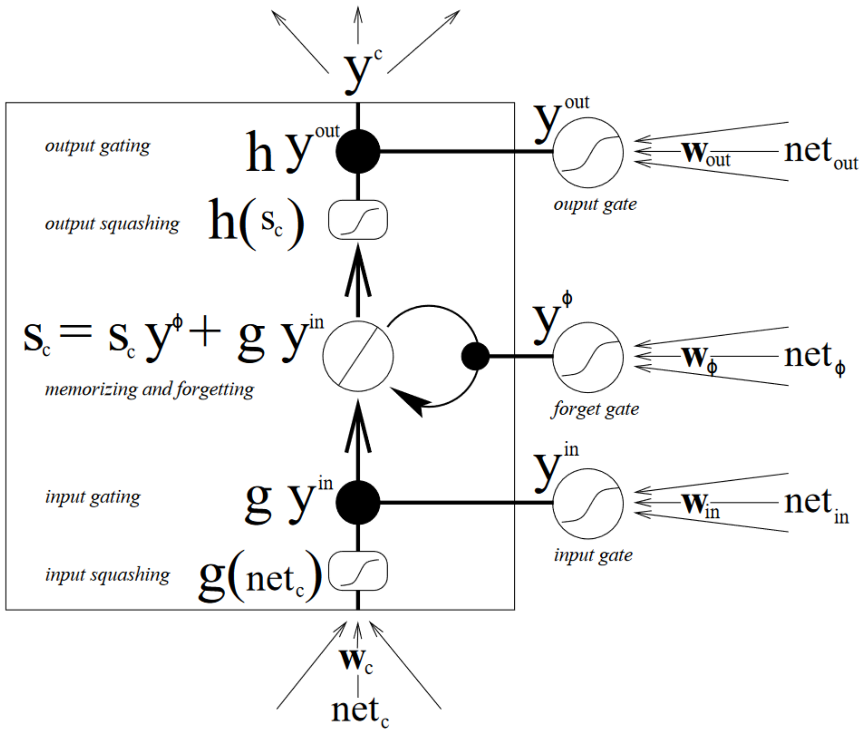
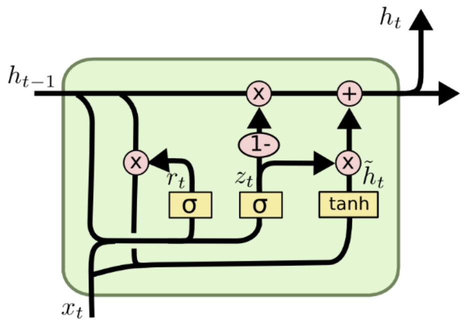


Figure 2.6: Illustration of the data flow in an LSTM neural network [43].

To reduce the complexity of the LSTM, Gated Recurrent Unit (GRU) has been proposed to avoid the exploding and vanishing gradient problem using lower computations. A GRU contains two gates to control the flow of the values through the neuron, which are the reset and update gates, as shown in Figure 2.7 [44]. The reset gate controls the effect of the values outputted from the previous timestep, depending on the importance of those values in the computation for the current input. The update gate controls the effect of the current input on the output of the unit, so that, the output can consider both the current and previous values depending on the decision made at these gates. Such topology achieves the same methodology of the LSTM using fewer computations, as it uses fewer gates. However, the qualities of the predictions for both methods are very similar and both methods must be evaluated in order to select the appropriate method for the required application [45-47].



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 2.7: Gated Recurrent Unit.

2.3 PERFORMANCE EVALUATION OF THE AUTHENTICATION SYSTEMS

As illustrated earlier, authentication systems rely on measuring the similarity between the template stored for the legitimate user and the descriptor or data collected from the current attempt. To convert such similarity to a binary decision, i.e. allow or deny access to the device, the similarity is compared to a threshold value, so that, only users with similarity lower than the threshold value are allowed access to the device. However, some intruders may be successful in producing such similarity measurement, lower than the threshold value, wherein such case they can gain access to the device, which is known as false acceptance. The ratio between the false acceptance and the total number of attacks executed against the authentication system is calculated as the security measure of the authentication system, which is denoted as the False Acceptance Rate (FAR) [48, 49]. Moreover, some of the similarity measures calculated for attempts collected from the legitimate user may not be less than, or equal to, the threshold value, in which the access to the device is denied. Such cases are known as false rejections, as the user has legitimate right to access the device, and the False Rejection Rate (FRR) is calculated to represent the usability of the authentication system [50, 51].

Authentication systems with high FAR are insecure, as most of the intrusion attempts may success when executed against the system, and such system is inapplicable even if low FRR persists. Moreover, systems with high FRR are of low usability even if they have low FAR, as legitimate users are not gaining the rightful access to the device. Additionally, the use of multiple values to compare authentication systems imposes difficulties toward reasonable comparison, as one system may outperform the other in one measure and the other system outperforms it in the other. Thus, the threshold value that produces equal FAR and FRR is selected to represent the Equal Error Rate (ERR) and this performance measure is normally used when available, as in some authentication systems, the threshold value is constant and cannot be adjusted to produce EER [52-54]. Figure 2.8 shows the variation in the FAR and FRR versus the threshold value and position of the EER value, using the Receiver Operating Characteristic (ROC) curve.

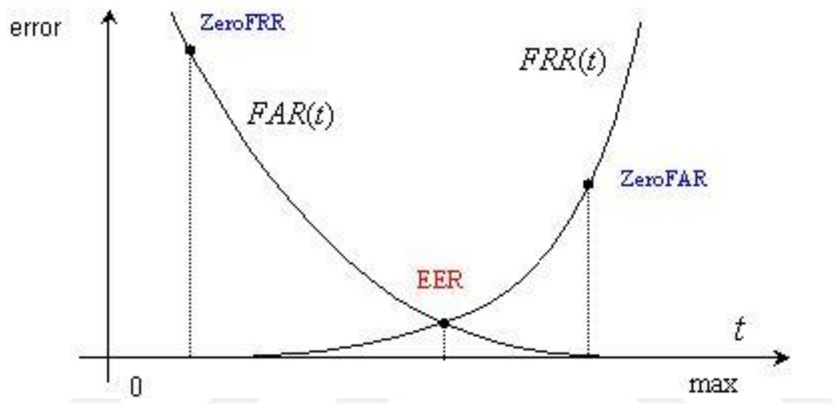


Figure 2.8: ROC curve and Equal Error Rate.

3. METHODOLOGY

The proposed method aims to produce a fixed-size descriptor in which the values summarize the behavioral biometrics collected for an attempt. This descriptor is generated from the information collected from the user during the drawing of the pattern secret, using the sensors available in the touch-sensitive screen. In addition to the traditional information that can be collected from any touch-sensitive screen, which are the coordinates of the pixel being touched, modern screens provide the pressure being applied and the size of the region covered by the touching finger. As the timestamp of each pixel touching is known, the speed and direction of the movement can be recognized and used as behavioral biometrics. However, such biometric information can change for the same individual, which should be considered in the generated descriptor, so that, the values are similar for the same individual regardless of the speed the user draws the pattern with. This can be achieved by considering the relative speed, i.e. monitoring the positions that the user normally has different speeds.

3.1 CHARACTERISTICS OF THE REQUIRED DESCRIPTOR

In order to define the neural network that is going to be implemented for the descriptor generation, it is important to define the characteristics of that descriptor. The main characteristics required for the descriptor are:

- The produced descriptor must be of a fixed size to ease the similarity measurement procedure. Comparing descriptors with different sizes requires more complex computations, such as the use of Dynamic Time Warping (DTW) method. The number of values in the generated descriptor is set to 128, similar to descriptors generated by several techniques for authentication purposes.
- The values in the descriptor must be different when different users draw the same pattern, so that, the descriptor can be used to distinguish different users even when the same pattern is used to protect their smart devices.

- The values in the descriptor must remain similar per user per pattern, regardless of the variation in their behavior when inputting the pattern. Such variation may occur according to the different situations and environments that the user may be in during the unlocking process of their devices.
- The descriptor’s values must be different when the same individual uses different patterns, so that, drawing the old pattern does not produce a match, even if the pattern is rejected by the default pattern authentication process.

3.2 NEURAL NETWORK IMPLEMENTATION FOR THE DESCRIPTOR GENERATION

The output from the neural network is computed based on the values in the input, using the forward pass. Each neuron is responsible for detecting a feature, or pattern, from the outputs of the neurons from the previous layer, so that, more complex features can be recognized. Accordingly, the values outputted from neurons in a certain layer represent features in the input, which can be used to produce the required output. Thus, these features must include the important information to be considered in the decision made by the neural network and neglect any information that has no effect.

To produce the required descriptors, a dataset is collected from a range of different users, where each user is required to draw a set of predefined patterns. Supposing U users have drawn P patterns, each for N times, a total of $U \times P$ unique attempts are produced in the collected dataset. This dataset is then used to train neural networks, with the topology shown in Figure 3.1. The neural network is trained to recognize the pattern and the user who has drawn it, so that, the values generated in the layer shaded in gray hold information of both of these variables. Thus, when the descriptors are similar, it is possible to recognize that the user is legitimate, while unsimilar descriptors indicate an intrusion attempt. Three types of special layers are evaluated in this study, which are the Convolutional GRU and LSTM neural networks.

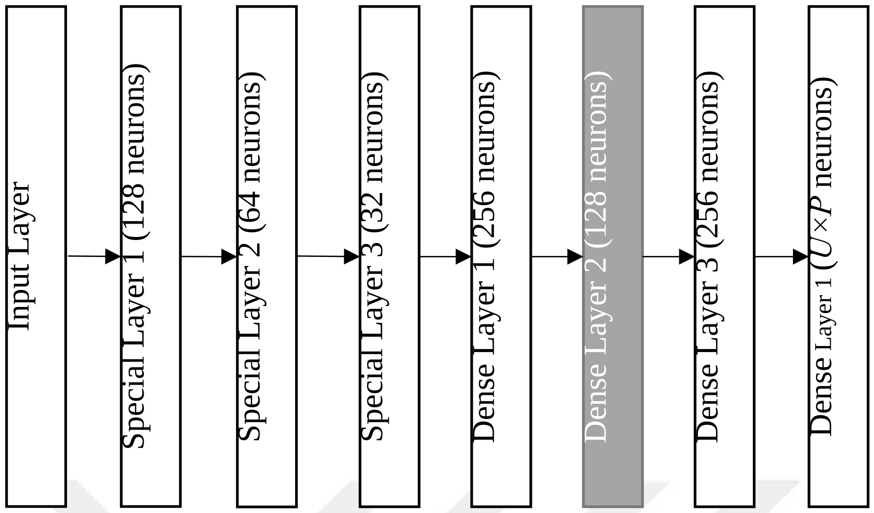

Input Layer
Special Layer 1 (128 neurons)
Special Layer 2 (64 neurons)
Special Layer 3 (32 neurons)
Dense Layer 1 (256 neurons)
Dense Layer 2 (128 neurons)
Dense Layer 3 (256 neurons)
Dense Layer 1 ($U \times P$ neurons)

Figure 3.1: Topology of the trained neural network.

3.3 DESCRIPTORS GENERATION AND MATCHING

During the training of the implemented neural network, to produce the required outputs at the output layer, the values in Dense Layer 2 must be invariant per user per pattern and have different values when any of these inputs change. Thus, these values can be used to summarize the behavioral biometrics collected from the pattern drawing attempt. Accordingly, the last two layers are removed from the neural network, which produces the neural network shown in Figure 3.2, where Dense Layer 2 is the output layer. Be feeding the data collected from the touch-sensitive screen the neural network produces the required fixed-size descriptor, which can be used to match the attempts.

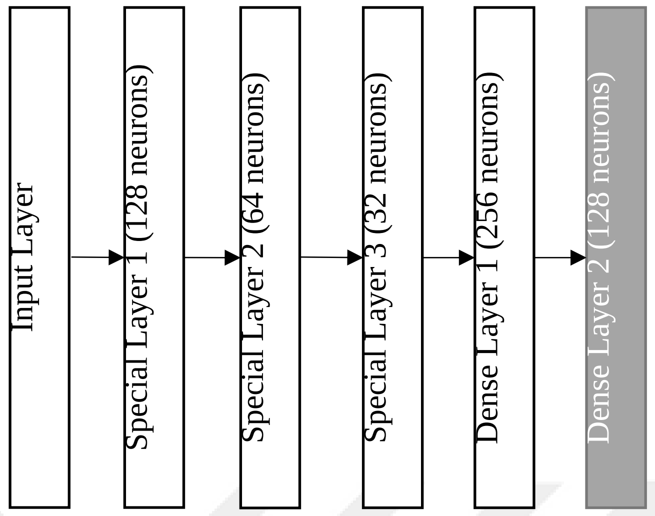

Input Layer
Special Layer 1 (128 neurons)
Special Layer 2 (64 neurons)
Special Layer 3 (32 neurons)
Dense Layer 1 (256 neurons)
Dense Layer 2 (128 neurons)

Figure 3.2: Descriptor generation neural network.

As the descriptor is of a fixed size, a simple distance measurement can be used to calculate the difference between any two descriptors, unlike the direct use of the collected data, which required more complex computations. For the proposed method, the Euclidean distance, shown in Equation 3.1, is used to measure the distance between any two descriptors, a and b, with i values in each. This distance can then be compared to a threshold value, selected based on the experimental results, to make a binary authenticity decision.

$$d = \sqrt{\sum_i (a_i - b_i)^2} \tag{3.1}$$

3.4 DATA PREPROCESSING

The data collected from the touch-sensitive screen, *x* and *y* coordinates of the touched pixel, the pressure applied on the screen and the size of the region covered by the finger, are appended to the timestamp of each tuple and used to generate the required descriptor. However, as neural networks required fixed-size inputs, regardless of the number of dimensions, it is important to preprocess these data before forwarding them to the network. Moreover, the preprocessing procedure depends on the type of the neural network, where the data formation required by the CNN is different from that required by the RNN. Thus, two preprocessing procedures are proposed, one per each type of neural network.

3.4.1 Preprocessing the Data for the CNN

A three-dimensional array is created to present the values collected from the user to the CNN. The first two dimensions are equal to the size of the touch-sensitive screen, i.e. the height and width of the screen. The third dimension is equal to three, one for the time, one for the pressure and one for the size. Each time value *t* is normalized to the interval required by the user to complete the pattern, based on the start time *st* and end time *et*, as shown in Equation 3.2.

$$t \leftarrow \frac{t - st}{et - st} \tag{3.2}$$

Then, each time, pressure and size values are positioned in the array depending on the position of the pixel being touched, so that, the distribution of the values per each layer reflects the drawn pattern. Moreover, to reduce the size of the input of the CNN, hence the complexity of the computation, the size of the produced array is reduced to one-third of the original size of the screen. The algorithm shown in Figure 3.3 summarizes the procedure conducted to convert the data for the CNN network.

Input: Raw data from the touch-sensitive screen. **Output:** A three-dimensional array for the CNN.	
Step 1:	T ← Read the time values from the data. P ← Read the pressure values from the data. S ← Read the size values from the data. C ← Read the coordinates of the touched pixels. O ← [, , ,] //Empty three-dimensional array for the output values.
Step 2:	For each timestamp t in T: $t \leftarrow \frac{t-st}{et-st}$

Step 3:	For i = 0 to length(T):
	    x ← int(C[i, 0]/3)
	    y ← int(C[i, 1]/3)
	    O[0, x, y] ← T_i
	    O[1, x, y] ← P_i

	O[2, x, y] ← S_i
Step 4:	Return O

Figure 3.3: Data Preprocessing Algorithm for the CNN.

3.4.2 Preprocessing the Data for the RNN

Unlike CNN, the RNN can only handle two-dimensional inputs, where one of these dimensions is the number of features in the descriptor and the other is the number of tuples in each series. Although the positioning of the values reflects the sequence they are collected in, the exact time difference between any two sequential values cannot be recognized based on the position. Thus, the timestamp is maintained in the inputs and is also normalized similarly to the CNN. Moreover, the length of the data is set to be equal to the total number of tuples in the longest data collected from the patterns. However, as the attempts cannot be guaranteed to have this length, the data of shorter patterns is padded to match that length. The padding uses a value of -1, as it does not occur in the actual data and can be easily recognized as padding by the neural network, and the actual data collected from the user is positioned after the padding, so that, the actual data is considered in the final output of the RNN. Moreover, the coordinates of the touched pixel are also fed to the neural network, producing five features per each tuple. The algorithm shown in Figure 3.4 summarizes the data preprocessing procedure for the RNN.

Input: Raw data from the touch-sensitive screen, Length of the required array. **Output:** A two-dimensional array for the RNN.	

Step 1:	T ← Read the time values from the data.
	P ← Read the pressure values from the data.
	S ← Read the size values from the data.
	C ← Read the coordinates of the touched pixels.
	L ← Read the required length of the data.
	O ← [L,5] //Empty two-dimensional array for the output values.
	O = -1 //Fill the array with the value -1.
	S = L - Length(T) //The position of the first tuple in the output array.

Step 2:	For each timestamp t in T:

	$t \leftarrow \frac{t-st}{et-st}$

Step 3:	For i = 0 to length(T):
	    O[S+I, 0:2] = C_i
	    O[S+I, 3] = T_i
	    O[S+I, 4] = S_i
	    O[S+I, 5] = P_i

Step 4:	Return O

Figure 3.4: Data Preprocessing Algorithm for the RNN.

4. EXPERIMENTAL RESULTS

To train the neural network implemented to generate the proposed descriptor, data is collected from volunteers and labeled according to the pattern number and the volunteer's name. Then, this data is preprocessed and inputted to each neural network in order to train it and evaluate the quality of the produced descriptors. The neural networks are implemented, trained and evaluated using Python [55] programming language with the aid of the Keras [56] Deep Learning (DL) and Scikit-Learn [57] Machine Learning (ML) libraries. The training and evaluation experiments are conducted using a computer with Intel® Core™ i7-8700K Processor at a frequency of 3.7GHz, 16GB of memory and 11GB of Graphical Processing Unit (GPU) memory in Nvidia GTX1080Ti display card.

4.1 DATA COLLECTION

An application is implemented using Android Studio Integrated Development Environment to replicate the pattern layout and collect data from the users. Five patterns are defined in the application, so that, each user is required to provide attempts per each pattern. Each volunteer is allowed to attempt the pattern until feels comfortable to draw it as their own pattern, without logging the data. Then, data logging is started and each volunteer is required to provide 50 attempts per each pattern. A counter is placed in the Graphical User Interface (GUI), so that, the user knows the number of successful attempts logged so far, and the application is set to terminate the logging and disable the layout when 50 successful attempts are collected. Figure 4.2 shows the GUI of the implemented application.

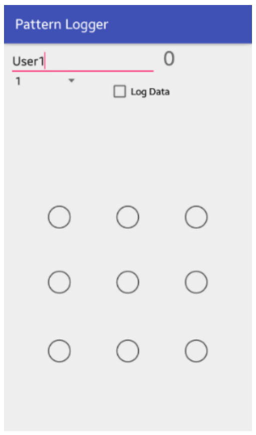


Figure 4.1: GUI of the implemented data logging Android application.

The patterns that are used to collect the data vary in the number of dots the user is required to connect and the complexity of the pattern, based on the positions of the connected dots. Such a variety of patterns are used to produce an unbiased and accurate evaluation, as users may choose different patterns with different levels of complexity. Figure 4.2 shows the pattern defined for the data collection phase.

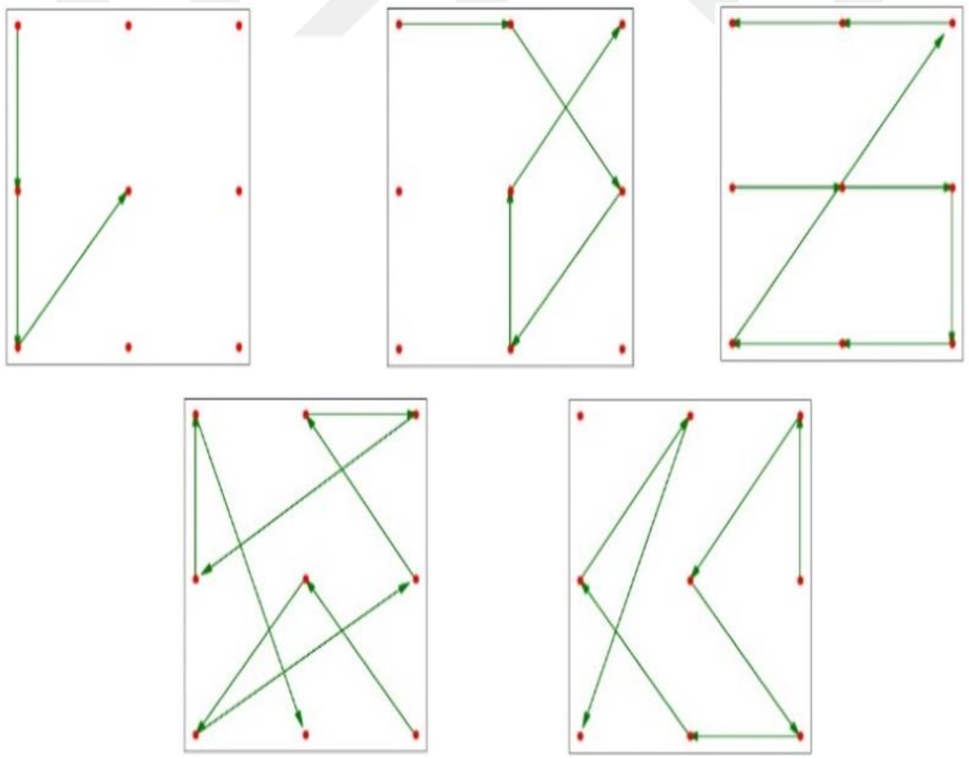


Figure 4.2: The patterns used for data logging from volunteers.

Random attempts from 26 of the volunteers are excluded from the training and used to evaluate the proposed method, where attempts from other users using the same pattern are used as intrusion attempts for the legitimate user being investigated. These sets are stored and used for all the experiments, so that, no biased evaluation is conducted as all the evaluated methods use the exact same training and testing data.

4.2 CNN'S PERFORMANCE EVALUATION

The convolutional neural network shown in Figure 4.3 is implemented and trained using the data collected from the volunteers. This neural network is trained for 100 epochs before the removal of the last two layers to produce the descriptors generation neural network. During training, the output layer is set to have 155 neurons, each neuron represents a certain user inputting a certain pattern. As the size of the screen is equal to 450×600 and the array generated for the CNN by the preprocessing procedure is one third in dimensions, the input layer is set to accept an array with the dimensions of 150×200×3.

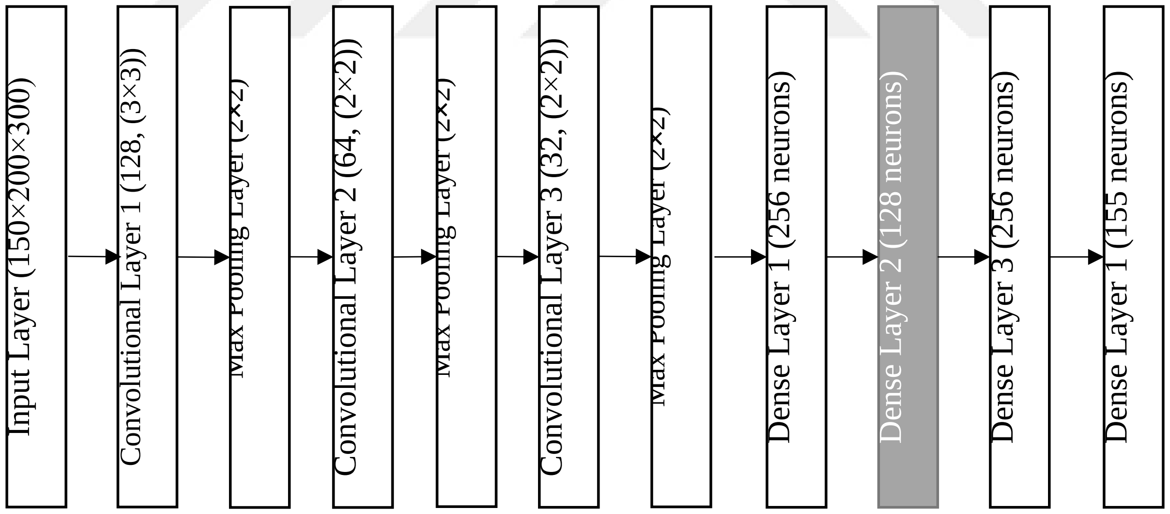

Input Layer (150×200×300)
Convolutional Layer 1 (128, (3×3))
Max Pooling Layer (2×2)
Convolutional Layer 2 (64, (2×2))
Max Pooling Layer (2×2)
Convolutional Layer 3 (32, (2×2))
Max Pooling Layer (2×2)
Dense Layer 1 (256 neurons)
Dense Layer 2 (128 neurons)
Dense Layer 3 (256 neurons)
Dense Layer 1 (155 neurons)

Figure 4.3: Topology of the trained CNN.

When the training epochs are finished, the convolutional neural network has achieved 98.40% accuracy, where the accuracy and loss versus the epochs are shown in Figure 4.4.

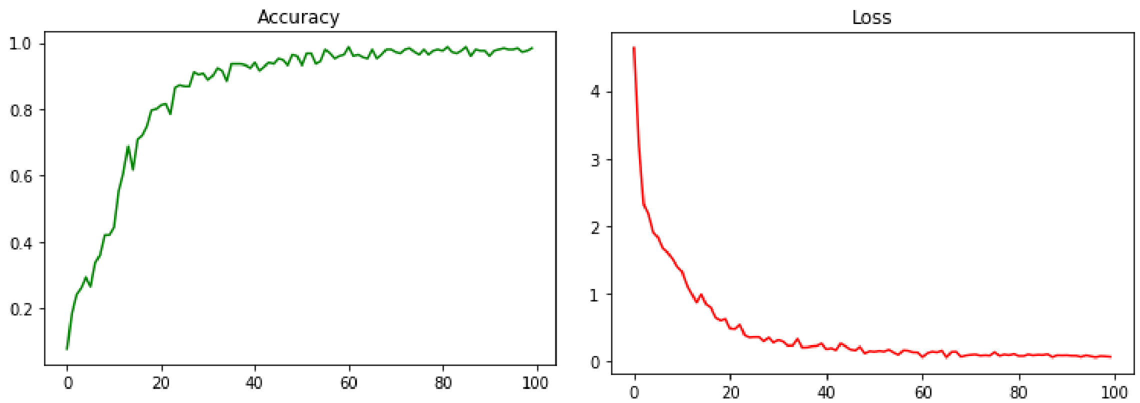


Figure 4.4: Accuracy and loss versus training epochs for the CNN.

Next, the descriptor generation neural network is extracted from the trained CNN and used to generate descriptors for the testing data collected from the volunteers. A template is generated for each user based on the median of the descriptors from the first five attempts. The Euclidean distance is then used to calculate the distance between each template and the remaining attempts of that user in the corresponding pattern. The distances are also measured using descriptors of users' attempts for the same patters, to simulate the intrusion attempts. Additionally, the templates are also compared to descriptors from other patterns, to ensure the existence of differences in the produced descriptors. The measured distances for these three situations are shown in Figure 4.5.

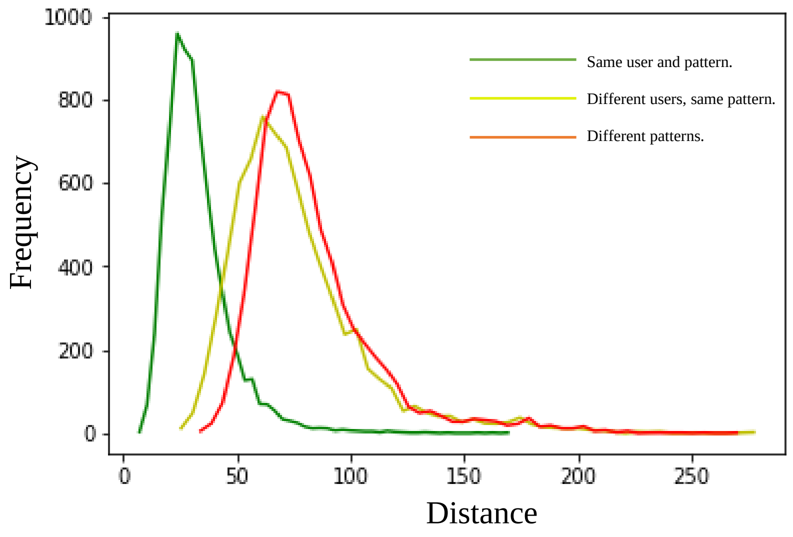


Figure 4.5: Distribution of distances measured from the CNN descriptors.

As shown in the figure, the distance between the template and attempts from other users using the same pattern are more similar than the attempts from other patterns. Thus, the performance measures of the system to detect the difference among these distances is used for the evaluation, as the distinguishing such attacks is more challenging to the proposed method. The ROC curve shown in Figure 4.6 shows that the EER of the proposed method based on the CNN's descriptors is 10.91% and is achieved at a threshold value of 50.48.

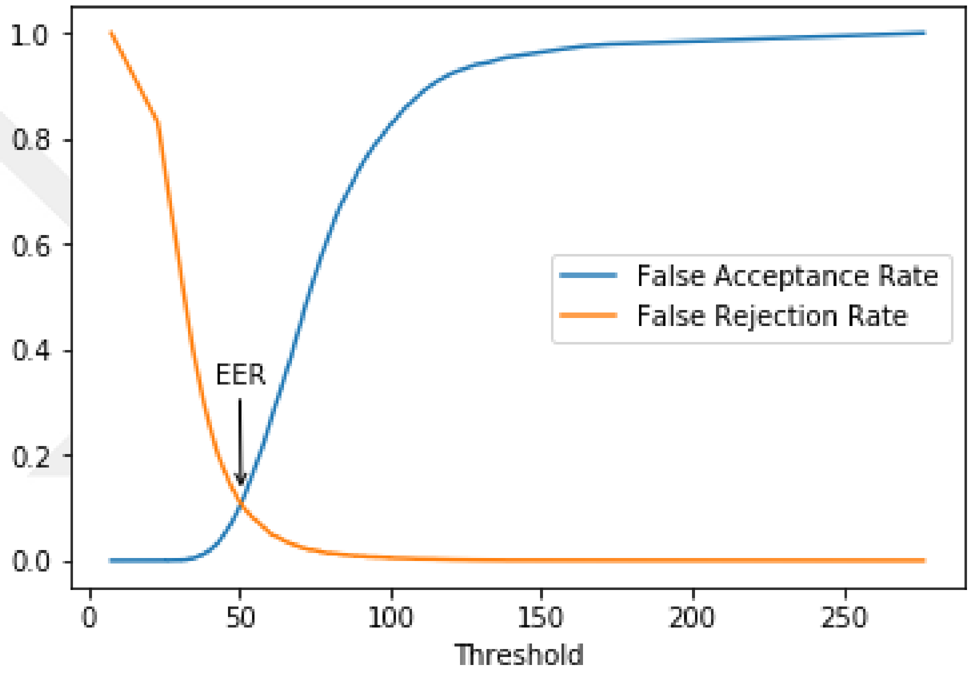


Figure 4.6: ROC curve of the predictions made depending on the descriptors of the CNN.

4.3 LSTM'S PERFORMANCE EVALUATION

The LSTM neural network shown in Figure 4.7 is implemented and trained using the data collected from the volunteers. This neural network is trained for 100 epochs before the removal of the last two layers to produce the descriptors generation neural network. During training, the output layer is set to have 155 neurons, each neuron represents a certain user inputting a specific pattern. As the longest data collected from the patterns has 650 instances, the input layer is set to handle 650×5 features.

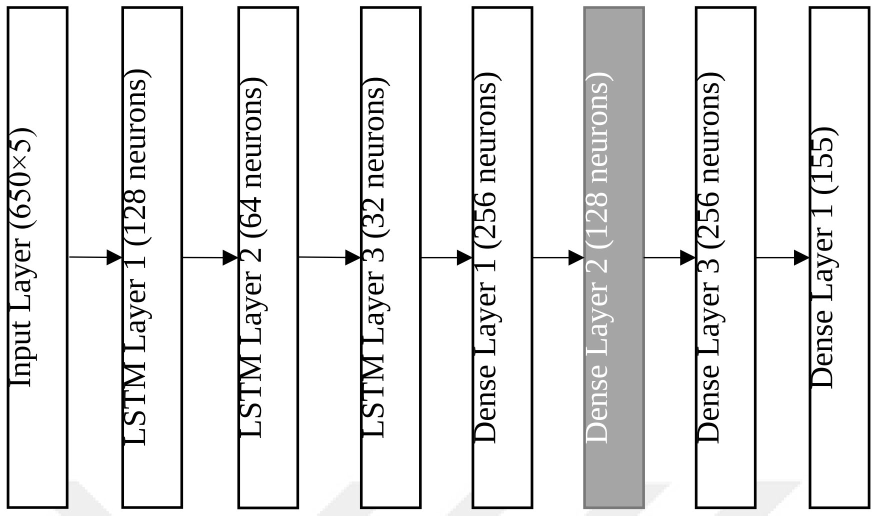

Input Layer (650×5)
LSTM Layer 1 (128 neurons)
LSTM Layer 2 (64 neurons)
LSTM Layer 3 (32 neurons)
Dense Layer 1 (256 neurons)
Dense Layer 2 (128 neurons)
Dense Layer 3 (256 neurons)
Dense Layer 1 (155)

Figure 4.7: Topology of the trained LSTM neural network.

Upon the termination of the training, the accuracy of the predictions provided by the neural network, which represent the user and pattern information, is 92.34%, which is lower than the accuracy achieved by the CNN. However, this accuracy does not necessarily indicate that the generated descriptor is of less quality than that generated by the CNN. Figure 4.8 shows the accuracy and loss versus the training epochs during the training of the LSTM neural network.

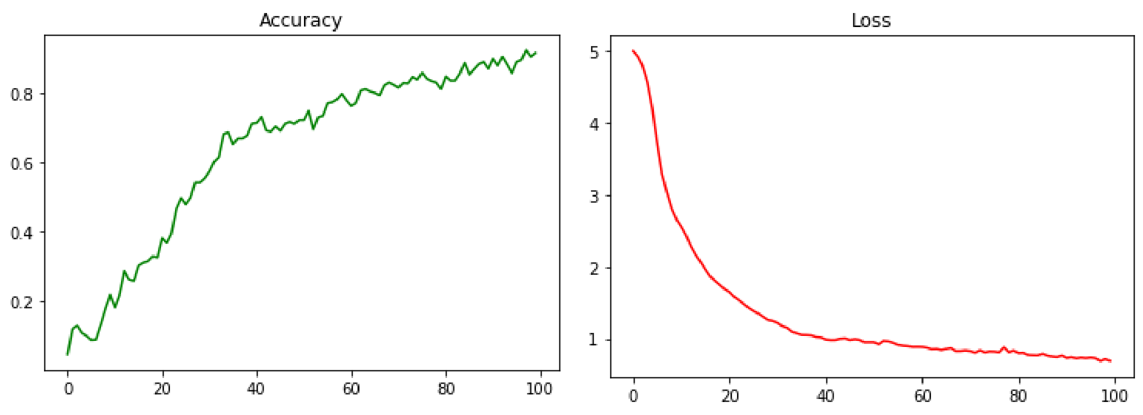


Figure 4.8: Accuracy and loss versus training epochs for the LSTM neural network.

Next, the descriptor generation neural network is extracted from the trained LSTM neural network and then used to generate descriptors for the testing data collected from the volunteers. A template is generated for each user based on the median of the descriptors from the first five attempts. The Euclidean distance is then used to calculate the distance between each template and the remaining attempts of that user in the corresponding pattern. The distances are also measured using descriptors of users' attempts for the same patters, to simulate the intrusion attempts. Additionally, the templates are also compared to descriptors from other patterns, to ensure the existence of difference in the produced descriptors. The measured distances for these three situations are shown in Figure 4.9.

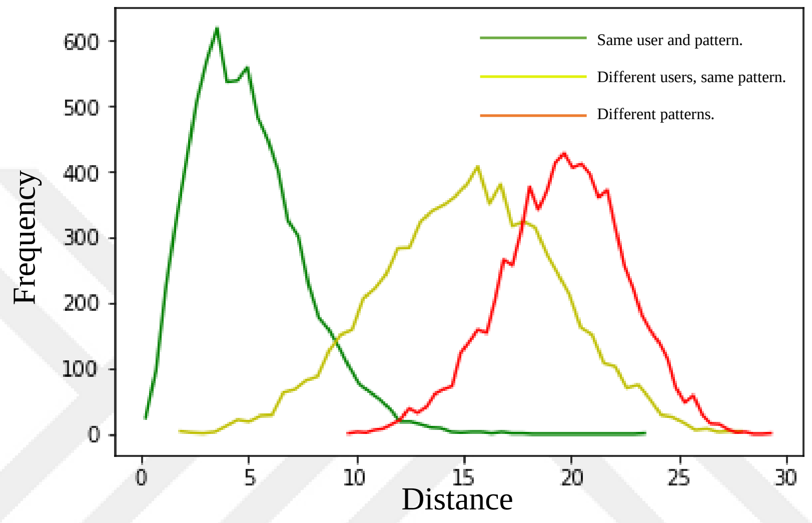


Figure 4.9: Distribution of distances measured from the LSTM descriptors.

Despite the lower distances calculated for the descriptors generated by the LSTM neural network, these distances have better distribution, regarding the relative distances between each type of distances, compared to those calculated using the CNN's descriptors. Moreover, the difference between the distances of the legitimate attempts and those using different template has increased significantly and has been able to achieve a very narrow intersection area. As the most challenging distinguishing is between the attempts of the legitimate user and those of the same pattern but from other users, the EER between these scenarios is calculated, using the ROC curve shown in Figure 4.10, which shows that the LSTM-based descriptors have been able to achieve 7.34% EER at 9.24 threshold value.

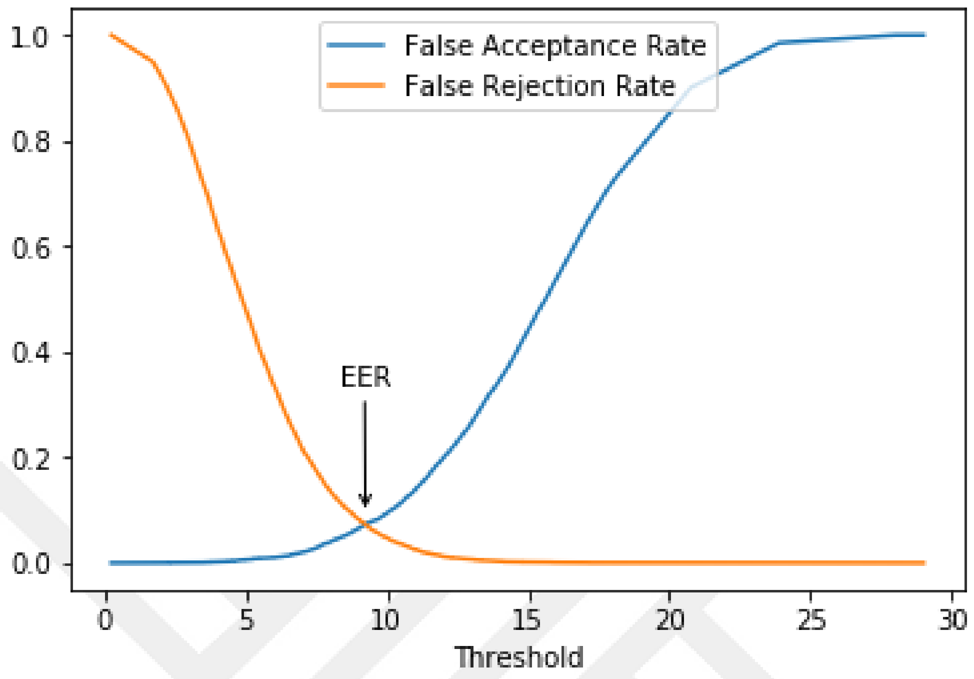


Figure 4.10: ROC curve of the predictions made depending on the descriptors of the LSTM neural network.

4.4 GRU'S PERFORMANCE EVALUATION

The GRU is used to implement the neural network shown in Figure 4.11 and trained using the data collected from the volunteers. This neural network is trained for 100 epochs before the removal of the last two layers to produce the descriptors generation neural network. During training, the output layer is set to have 155 neurons, each neuron represents a certain user inputting a specific pattern. Similar to the LSTM neural network, the longest data collected from the patterns has 650 instances, the input layer is set to handle 650×5 features.

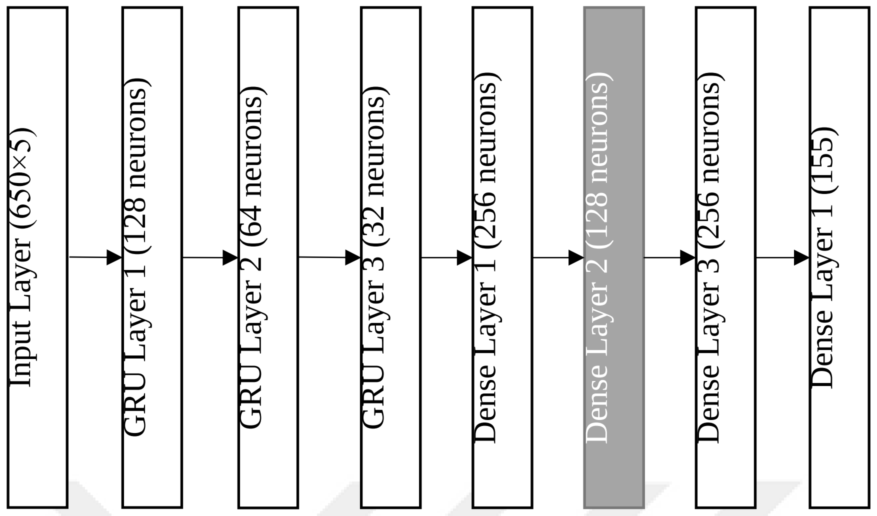

Input Layer (650×5)
GRU Layer 1 (128 neurons)
GRU Layer 2 (64 neurons)
GRU Layer 3 (32 neurons)
Dense Layer 1 (256 neurons)
Dense Layer 2 (128 neurons)
Dense Layer 3 (256 neurons)
Dense Layer 1 (155)

Figure 4.11: Topology of the trained LSTM neural network.

Upon the termination of the training, the accuracy of the predictions provided by the neural network, which represent the user and pattern information, is 95.04%, which is also lower than the accuracy achieved by the CNN but higher than that achieved by the LSTM neural network. Figure 4.12 shows the accuracy and loss versus the training epochs during the training of the LSTM neural network.

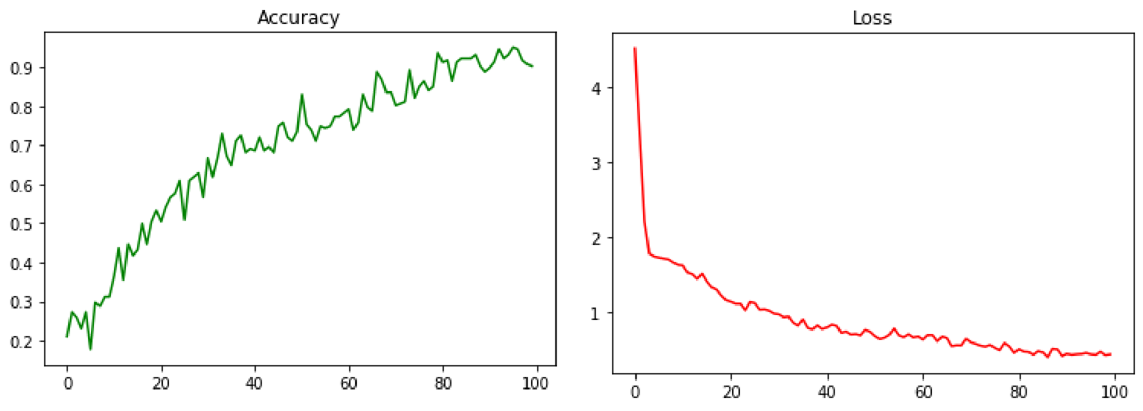


Figure 4.12: Accuracy and loss versus training epochs for the LSTM neural network.

Next, the descriptor generation neural network is extracted from the trained GRU neural network and then used to generate descriptors for the testing data collected from the volunteers. A template is generated for each user based on the median of the descriptors from the first five attempts. The Euclidean distance is then used to calculate the distance between each template and the remaining attempts of that user in the corresponding pattern.

The distances are also measured using descriptors of users' attempts for the same patters, to simulate the intrusion attempts. Additionally, the templates are also compared to descriptors from other patterns, to ensure the existence of difference in the produced descriptors. The measured distances for these three situations are shown in Figure 4.13.

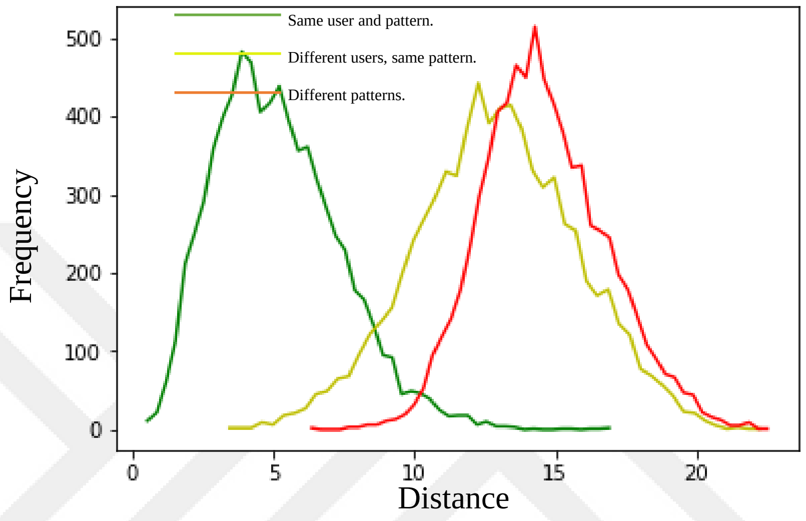


Figure 4.13: Distribution of distances measured from the GRU descriptors.

The distribution of the distances shows a narrower intersection between the distances among the attempts of the same user in the same pattern and the attempts of other users using the same pattern. Moreover, the use of the descriptors generated by the GRU neural network has maintained the difference with the attempts for other patterns. The EER of the proposed system using the descriptors of the GRU neural network is 6.91% achieved at a threshold distance value of 8.81, as shown in the ROC curve in Figure 4.14.

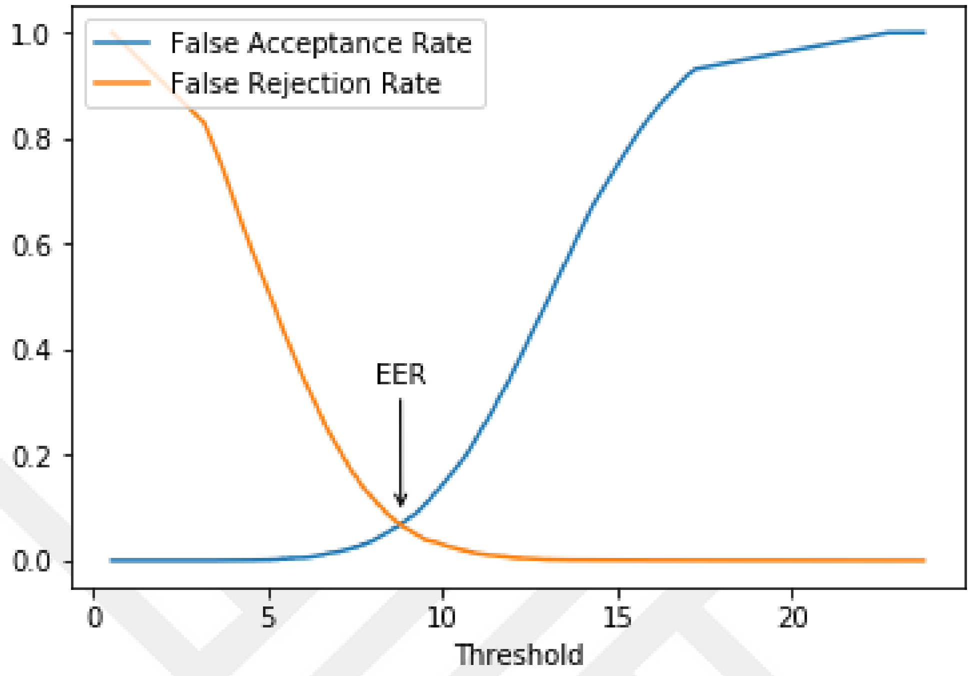


Figure 4.14: ROC curve of the predictions made depending on the descriptors of the LSTM neural network.

5. DISCUSSION

The performance of the proposed system using the descriptors generated by the CNN, LSTM and GRU neural network is summarized in Table 5.1. The best performance of the proposed system has been accomplished based on the descriptors generated by the GRU neural network, which has only 6.91% EER. This low error rate indicates that only 6.91% of the legitimate user attempts are denied by the proposed method, while 93.09% of the intrusion attempts are detected and denied access. Moreover, as Figure 5.1 shows, the lower training accuracy of the GRU neural network, compared to the CNN, proves the hypothesis proposed in this study, which states that the training accuracy does not necessarily reflect the quality of the descriptors generated by the neural network. The LSTM neural network has significantly lower training accuracy, compared to the CNN, but has also achieved significantly lower EER. Moreover, the training accuracy of the GRU neural network is between those of the LSTM and CNN but the descriptors generated using the GRU neural network have the lowest EER. The values in the descriptors generated by the CNN network can be overfitted on the data used for the training, which is the reason behind the high training accuracy and low EER, while the values in the descriptors generated by the LSTM neural network are of low quality, which is the reason behind the low training accuracy and relatively high EER, compared to the GRU neural network. The GRU neural network has achieved the balanced performance during both the training and the descriptors generation.

Table 5.1: Summary of the performance for the evaluated neural networks.

Neural Network	**Training Accuracy**	**EER**	**Threshold**
CNN	98.40%	10.91%	50.48
LSTM	92.34%	7.34%	9.24
GRU	95.04%	**6.91%**	8.81

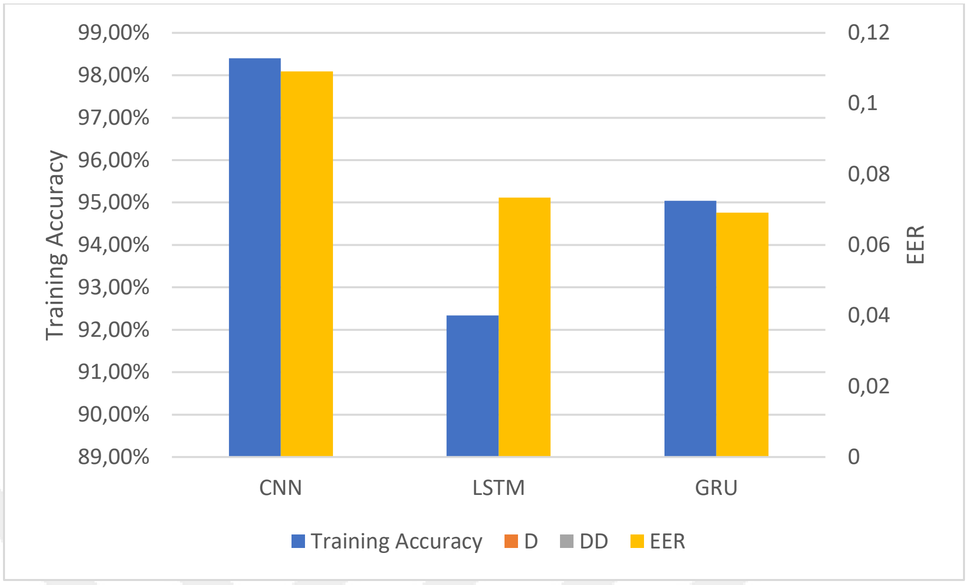


Figure 5.1: Training accuracies and EERs of the evaluated neural networks.

Additionally, Table 5.2 shows that the proposed method has better performance than the method proposed by De Luca et al. [14], using the descriptors generated by any type of neural networks. Moreover, the use of the descriptors generated by the GRU neural network has significantly better performance measures, with 96.09%, compared to De Luca's method, which has only 77% accuracy. The use of the GRU neural network's descriptors has also produced a more secure authentication system, with only 6.91% FAR, compared to 21% in the method proposed by De Luca et al. The usability of the proposed system is also improved with only 6.91% FRR using the descriptors from the GRU neural network, compared to De Luca's method, which has 19% FRR.

Table 5.2: Performance comparison to the state-of-the-art method in the literature.

		This Study		
	De Luca et al. [14]	**CNN**	**LSTM**	**GRU**
True Positive	398	437	454	456
False Negative	92	53	36	34
True Negative	852	965	1004	1009
False Positive	231	118	79	74
Accuracy	77%	89.09%	92.66%	93.09%
FAR	21%	10.91%	7.34%	6.91%
FRR	19%	10.91%	7.34%	6.91%

6. CONCLUSION

The rapid development of the digital era has increased the storage of digital data on different types of devices. The revolution in the electronics industry has also increased the number of smart devices with touch-sensitive screens, such as tablets and smartphones, to accomplish different tasks. The exponentially growing amount of digital information being stored on the devices has risen concerns about the security of such information, especially with sensitive and personal nature of such information. Thus, protecting such data from any unauthorized access, even if physical access to the device is gained, has become of more interest in recent years.

The use of biometric authentication has shown good solution toward protecting this information, as the replication of biometric features is a difficult task to be achieved by an attacker. However, concerns about the privacy of the collected information have been rising by the users of such devices, according to the sensitivity of physiological biometrics. Thus, most of the users have chosen to avoid the use of physiological biometric authentication. As a substitute, behavioral biometrics are being used to distinguish and authenticate users into smart devices. However, according to the lower robustness of behavioral biometrics, compared to physiological, the existing method lack the accuracy required to maintain high security and usability measures.

In this study, a new behavioral biometric authentication system is proposed, based on artificial neural networks and distance measurement. The proposed method collects behavioral biometrics from the touch-sensitive screen during the drawing of the graphical secret, known as pattern, which is the most widely used type of authentication. Different types of artificial neural networks are used to generate a more robust and distinctive descriptor for the data collected during the drawing of the pattern. Then, the descriptor stored as the template of the user is matched with the one collected from the current attempt to make the appropriate authenticity decision.

First, each neural network is trained to recognize each user per each pattern. Then, the values collected from the neuron of a hidden layer, with 128 neurons, are selected as the descriptor of the input collected from the attempt. Three types of artificial neural networks are evaluated for this purpose, which are the CNN, LSTM and GRU neural networks. The evaluation results show that the descriptors generated using the GRU neural network have the highest quality, as the proposed system has been able to achieve low EER of only 6.91%, compared to 10.91% and 7.34% for the CNN and LSTM, respectively. The results also show that the proposed system has been able to outperform the existing state-of-the-art method, which has been able to achieve only 77% accuracy, compared to 93.09% achieved by the proposed system.

In future work, the performance of the proposed method with other secret-based method is going to be evaluated, such as Personal Identification Number (PIN), so that, the proposed method can allow more flexibility to the user in selecting the authentication method of their choice, rather than limiting it to pattern-based methods.

REFERENCES

[1] M. E. Chua, M. A. Saunders, P. R. Bowlin, J. M. Ming, R. I. Lopes, W. A. Farhat, *et al.*, "Impact of smartphone digital photography, email, and media communication on emergency room visits post-hypospadias repair," *Canadian Urological Association Journal*, vol. 11, p. E134, 2017.

[2] A. Rahim, S. Z. Safin, L. K. Kheng, N. Abas, and S. M. Ali, "Factors influencing purchasing intention of Smartphone among university students," *Procedia Economics and Finance,* vol. 37, pp. 245-253, 2016.

[3] L. Vasudevan, K. Zeller, and A. Labrique, "Mobile Health," in *Digital Health*, ed: Springer, 2018, pp. 15-25.

[4] M. T. Ahvanooey, Q. Li, M. Rabbani, and A. R. Rajput, "A Survey on Smartphones Security: Software Vulnerabilities, Malware, and Attacks," *INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS,* vol. 8, pp. 30-45, 2017.

[5] J. Velho, D. Marques, T. Guerreiro, and L. Carriço, "Physical Intrusion Detection and Prevention for Android Smartphones," in *INFORUM*, 2015.

[6] C. Bhagavatula, B. Ur, K. Iacovino, S. M. Kywe, L. F. Cranor, and M. Savvides, "Biometric authentication on iphone and android: Usability, perceptions, and influences on adoption," *Proc. USEC,* pp. 1-2, 2015.

[7] A. Alzubaidi and J. Kalita, "Authentication of smartphone users using behavioral biometrics," *IEEE Communications Surveys & Tutorials,* vol. 18, pp. 1998-2026, 2016.

[8] S. Acharya, A. Polawar, and P. Pawar, "Two factor authentication using smartphone generated one time password," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. 11, pp. 85-90, 2013.

[9] F. Monrose and A. D. Rubin, "Keystroke dynamics as a biometric for authentication," *Future Generation computer systems,* vol. 16, pp. 351-359, 2000.

[10] H. Saevanee and P. Bhatarakosol, "User authentication using combination of behavioral biometrics over the touchpad acting like touch screen of mobile device," in *Computer and Electrical Engineering, 2008. ICCEE 2008. International Conference on*, 2008, pp. 82-86.

[11] T. J. Neal and D. L. Woodard, "Surveying biometric authentication for mobile device security," *Journal of Pattern Recognition Research,* vol. 1, pp. 74-110, 2016.

[12] T. Van Nguyen, N. Sae-Bae, and N. Memon, "DRAW-A-PIN: Authentication using finger-drawn PIN on touch devices," *Computers & Security,* vol. 66, pp. 115-128, 2017.

[13] N. Malkin, M. Harbach, A. De Luca, and S. Egelman, "THE ANATOMY OF SMARTPHONE UNLOCKING: Why and How Android Users Around the World Lock their Phones," *GetMobile: Mobile Computing and Communications,* vol. 20, pp. 42-46, 2017.

[14] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann, "Touch me once and i know it's you!: implicit authentication based on touch screen patterns," in *proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012, pp. 987-996.

[15] O. N. Uçan, O. Bayat, and M. B. Çoşkun, "Development and evaluation of the authentication systems by using phase-only correlation palm print identificaton methods," in *2017 International Conference on Engineering and Technology (ICET)*, 2017, pp. 1-4.

[16] W. Meng, D. S. Wong, S. Furnell, and J. Zhou, "Surveying the development of biometric user authentication on mobile phones," *IEEE Communications Surveys & Tutorials,* vol. 17, pp. 1268-1293, 2015.

[17] A. Mahfouz, I. Muslukhov, and K. Beznosov, "Android users in the wild: Their authentication and usage behavior," *Pervasive and Mobile Computing,* vol. 32, pp. 50-61, 2016.

[18] L. Sobrado and J.-C. Birget, "Graphical passwords," *The Rutgers Scholar, an electronic Bulletin for undergraduate research,* vol. 4, pp. 12-18, 2002.

[19] R. Spolaor, Q. Li, M. Monaro, M. Conti, L. Gamberini, and G. Sartori, "Biometric Authentication Methods on Smartphones: A Survey," *PsychNology Journal,* vol. 14, 2016.

[20] M. Harbach, A. De Luca, and S. Egelman, "The anatomy of smartphone unlocking: A field study of android lock screens," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, 2016, pp. 4806-4817.

[21] G. Cho, J. H. Huh, J. Cho, S. Oh, Y. Song, and H. Kim, "Syspal: System-guided pattern locks for android," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 338-356.

[22] A. H. Lashkari, S. Farmand, D. Zakaria, O. Bin, and D. Saleh, "Shoulder surfing attack in graphical password authentication," *arXiv preprint arXiv:0912.0951,* 2009.

[23] A. EJJARI, P. GÖRGEL, and A. SERTBAŞ, "COMPUTER AIDED PALM VEIN AUTHENTICATION SYSTEM BASED ON FEATURE MATCHING," *Electronic Turkish Studies,* vol. 12, 2017.

[24] R. A. Isaac, A. Kathera, K. Venkatachalam, M. T. Raj, and G. Gokulnath, "Spoofing Detection for Fingerprint, Palm-Vein and Facial Recognition Using Deep Representation."

[25] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Advances in Neural Information Processing Systems*, 2015, pp. 1117-1125.

[26] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in neural information processing systems*, 2014, pp. 2924-2932.

[27] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *International Conference on Machine Learning*, 2015, pp. 2113-2122.

[28] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience,* vol. 10, p. 508, 2016.

[29] K. B. Nahato, K. N. Harichandran, and K. Arputharaj, "Knowledge mining from clinical datasets using rough sets and backpropagation neural network," *Computational and mathematical methods in medicine,* vol. 2015, 2015.

[30] P. Gorgel, N. Kilic, B. Ucan, A. Kala, and O. N. Ucan, "A Backpropagation Neural Network Approach For Ottoman Character Recognition," *Intelligent Automation & Soft Computing,* vol. 15, pp. 451-462, 2009.

[31] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks,* vol. 61, pp. 85-117, 2015.

[32] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[33] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Neural Networks (IJCNN), 2015 International Joint Conference on*, 2015, pp. 1-8.

[34] A. Severyn and A. Moschitti, "Unitn: Training deep convolutional neural network for twitter sentiment classification," in *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, 2015, pp. 464-469.

[35] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "Cnn-rnn: A unified framework for multi-label image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2285-2294.

[36] B. Kayalibay, G. Jensen, and P. van der Smagt, "CNN-based segmentation of medical imaging data," *arXiv preprint arXiv:1701.03056,* 2017.

[37] Y. Lu, S.-C. Zhu, and Y. N. Wu, "Learning frame models using cnn filters," *arXiv preprint arXiv:1509.08379,* 2015.

[38] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of CNN activations," *arXiv preprint arXiv:1511.05879,* 2015.

[39] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015, pp. 234-241.

[40] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329,* 2014.

[41] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104-3112.

[42] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[43] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," 1999.

[44] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[45] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2016, pp. 324-328.

[46] B. Athiwaratkun and J. W. Stokes, "Malware classification with LSTM and GRU language models and a character-level CNN," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2482-2486.

[47] R. Dey and F. M. Salemt, "Gate-variants of gated recurrent unit (GRU) neural networks," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 1597-1600.

[48] G. Kambourakis, D. Damopoulos, D. Papamartzivanos, and E. Pavlidakis, "Introducing touchstroke: keystroke-based authentication system for smartphones," *Security and Communication Networks*, vol. 9, pp. 542-554, 2016.

[49] S. Li, A. Ashok, Y. Zhang, C. Xu, J. Lindqvist, and M. Gruteser, "Whose move is it anyway? Authenticating smart wearable devices using unique head movement patterns," in *2016 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2016, pp. 1-9.

[50] S. J. Kang, S. Y. Lee, H. I. Cho, and H. Park, "ECG authentication system design based on signal analysis in mobile and wearable devices," *IEEE Signal Processing Letters*, vol. 23, pp. 805-808, 2016.

[51] T. Kawamata, T. Ishii, and T. Akakura, "Face authentication for e-Learning using time series information," in *2016 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 2016, pp. 116-121.

[52] L. Fridman, S. Weber, R. Greenstadt, and M. Kam, "Active authentication on mobile devices via stylometry, application usage, web browsing, and GPS location," *IEEE Systems Journal*, vol. 11, pp. 513-521, 2017.

[53] A. Salem and M. S. Obaidat, "A novel security scheme for behavioral authentication systems based on keystroke dynamics," *Security and Privacy*, p. e64, 2019.

[54] A. S. H. ALNASER, O. N. UÇAN, and O. BAYAT, "Passive Detection of Islanding Events in Microgrids Using Machine Learning," *AURUM Journal of Engineering Systems and Architecture*, vol. Submitted, 2019.

[55] M. F. Sanner, "Python: a programming language for software integration and development," *J Mol Graph Model,* vol. 17, pp. 57-61, 1999.

[56] F. Chollet, "Keras: The python deep learning library," *Astrophysics Source Code Library,* 2018.

[57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel*, et al.*, "Scikit-learn: Machine learning in Python," *Journal of machine learning research,* vol. 12, pp. 2825-2830, 2011.