Keystroke Dynamics Analysis for User Authentication Using a Deep Learning Approach

Najwa Altwaijry

Department of Computer Science
College of Computer and Information Sciences
King Saud University
Riyadh, Saudi Arabia

Summary

With the ubiquitous use of the Internet, the importance of secure access to computing resources has grown. Many computer systems authenticate users through a password selected by the user. User typing patterns are usually distinct, allowing users to be differentiated and verified through a suitable verification system, and are considered a behavioral biometric. Authentication based on keystroke dynamics has many advantages, such as ease of data acquisition, continuous non-intrusive monitoring, and ease of integration into existing systems. In this paper, we present a convolutional neural network, which we call CNN-Detect, to detect unauthorized users that attempt to access resources by their typing patterns. We test our model on the publicly available CMU keystroke dynamics dataset, after suitable feature engineering. Our proposed model shows significant improvement over other models in the literature, achieving an average equal error rate (EER) of 0.009, and a zero-miss false acceptance rate (ZM-FAR) of 0.027.

Keywords:
Deep Learning; Convolutional Neural Network; Keystroke Dynamics; Machine Learning, Biometrics.

1. Introduction

Access control is a broad term that covers several different types of mechanisms that enforce access control features on computer systems, networks, and information. Secure access to computing resources is an integral part of security systems, especially considering the ubiquitous use of computers nowadays. For a user to be able to access a resource, he or she must be identified, authenticated, authorized. Identification can be provided with the use of a username, while authentication is the process of verifying the identity of a subject, usually by providing a second piece to the credential set, such as a password. Simple and low-cost user identification and authentication is a desirable aspect of security systems. Although passwords and passphrases are common authentication methods, they are not free from drawbacks. For example, passwords may be forgotten, lost or hacked.

Biometrics refers to any measure used to uniquely identify a user based on biological or physiological traits and is considered one of the most effective and accurate methods of verifying identification. Unfortunately, it is usually much more expensive and complex than other types of verification. A behavioral biometric refers to any pattern of behavior that is specific to the user, such as the rhythm and cadence with which users type on their computer keyboard. Behavioral biometrics are attractive as they allow more secure authentication, while still being less expensive and invasive than physical biometrics.

Keystroke dynamics is a behavioral biometric that analyzes typing rhythms to differentiate among authentic users and impostors, both insiders and external attackers [11]. It is difficult to impersonate typing behavior, making it a useful biometric measure. As it is also unobtrusive, it can augment standard security measures seamlessly [1]. Keystroke dynamics are considered a cost effective and user-friendly user authentication method. On the other hand, the physical and emotional state of an individual affects this biometric [7], potentially causing misclassification of users.

Biometric analysis of typed text can be either structured, on predetermined phrases, or dynamic, on continuously typed text. Many features may be extracted from text, such as key-press latency (keyup-keyup, keydown-keydown, keydown-keyup), pressure, force, total duration, or speed.

More formally, a keystroke dynamics dataset D is comprised of a collection of keystroke sequences ($x_i \,|x_i \in D$), where $x = \{x_1, x_2, \dots, x_n\}$ is a keystroke sequence with n attributes, and x_i represents time duration in milliseconds (ms). x_i is generated from various timing attributes, such as keyup-keyup or keydown-keydown, collected as text input of length ($m|m < n$) that is entered by a user.

Various approaches have been proposed in the literature to classify users based on the features collected, such as statistical methods that calculate the mean, standard deviation, or other distance measures, and machine learning methods, including neural networks, support vector machines (SVM), and decision trees. Other approaches used evolutionary and swarm intelligence algorithms, such as genetic algorithms (GA), ant colony optimization (ACO), and particle swarm optimization (PSO). Unfortunately, it is

Manuscript received December 5, 2020
Manuscript revised December 20, 2020
https://doi.org/10.22937/IJCSNS.2020.20.12.23

not always possible to compare different methods and algorithms, because of the different datasets researchers employ in their studies. For this reason, this work focuses on the CMU keystroke dynamics benchmark dataset [11].

This paper aims to leverage a deep neural network to detect unauthorized users that attempt to access computing resources by entering a correct password. The neural network should learn the specific features of an authentic user's typing patterns, i.e. the user's behavioral biometric, in order to correctly deny access to imposters. This work presents the following contributions to the literature: a feature engineering approach of the CMU dataset as explained in Section 3.2, and a novel convolutional neural network model for behavioral biometrics. Our proposed model outperforms other models in the literature by a significant margin, indicating its applicability as a suitable behavioral biometric verification method.

Table 1: Nomenclature of abbreviations

Acronyms	Definition
ACO	Ant Colony Optimization
ANN	Artificial neural network
CMU	Carnegie Mellon University
CNN	Convolutional neural network
DFS	Deep Feature Synthesis
DNN	Deep Neural Network
EER	Equal Error Rate
FAR	False Acceptance Rate
FRR	False Rejection Rate
GA	Genetic Algorithms
GRU	Gated Recurrent Unit
k-NN	k-Nearest Neighbor
LSTM	Long short-term memory
PSO	Particle Swarm Optimization
ReLU	Rectified Linear Unit
ROC	Receiver operating characteristic
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support vector machines

The rest of this paper is organized as follows. Section 2 presents a review on various works in this field. Section 3 describes the CMU dataset, the data preprocessing, and our model, CNN-Detect. Section 4 presents a detailed evaluation of CNN-Detect on the CMU dataset. We conclude our work in Section 5. Table 1 presents the nomenclature of abbreviations used in this paper.

2. Related Work

This section presents research work on the usage of keystroke dynamics as a biometric method for user identification and authentication, with particular emphasis on works done on the CMU dataset by Killourhy and Maxion [11]. The CMU dataset is a keystroke dynamics benchmark dataset. Killourhy and Maxion implemented 14 detectors on their proposed dataset: Manhattan (scaled), Nearest Neighbor (Mahalanobis), Outlier Count (z-score), SVM (one-class), Mahalanobis, Mahalanobis (normed), Manhattan (filter), Manhattan, Neural Network (auto-assoc), Euclidean, Euclidean (normed), Fuzzy Logic, k-Means, and Neural Network (standard). They trained each detector on their proposed dataset as follows: train detector on 200 passwords by a user and use a further 200 passwords by the same user to generate *user scores*. They also generate 5 *imposter scores* per 50 imposters. They use the *user scores* and *imposter scores* to generate an ROC curve and calculate the equal-error and zero-miss rates for each user. The error means of the 51 users are used to measure the detectors performance. The best equal-error rate was 0.096, obtained by the Manhattan (scaled) detector, and the best zero-miss false-alarm rate was 0.468, obtained by the Nearest Neighbor (Mahalanobis) detector.

Zhong et al. [18] improved the above results on the CMU dataset using a new distance metric that incorporates the Mahalanobis distance and Manhattan distance. First, features are normalized and decorrelated, and then the Manhattan distance is computed. They applied the nearest neighbor classifier on the new features, and also tested the classifier with outliers removed. They achieved an equal-error rate of 0.084 and a zero-miss false-alarm rate of 0.405. In another study, Deng and Zhong [6] performed user authentication using a Gaussian mixture model with universal background model (GMM-UBM), identity vector (i-vector), and deep neural network (DNN). They achieved the best performance using the DNN model [5], with an equal-error rate of 0.035.

Ho and Kang [8] proposed two algorithms: the mini-batch bagging (MINIBAG) method, tested with Euclidean, Manhattan, and Manhattan scaled, as well as with various aggregation operators, and the attribute ranking of one-class naÃ¯ve Bayes (AR-ONENB) algorithm. MINIBAG with Manhattan scaled achieved a best EER of 0.056 using aggregation operators, and AR-ONENB achieved an EER of 0.066. Both reported scores are for an imposter unfamiliar with the password, as done in [11].

Ivannikova et al. [9] introduced a Dependence Clustering (DC) based approach for user authentication using keystroke dynamics, as well as applying a k-NN-based approach. They also designed a cross-validation procedure which artificially generates impostor samples to improve the learning process while allowing fair comparison to previous works. The DC approach achieved

an EER of 0.077, and the *k*-NN approach achieved an EER of 0.078, outperforming previous works in the literature.

Kobojek and Saeed [13] presented two models, a long short-term memory (LSTM) model and a gated recurrent unit (GRU) model trained on the CMU dataset. They tested various architectures for the LSTM model, achieving a best equal-error rate of 0.14 using a 2-cell LSTM model and a zero-miss false-alarm rate of 0.379. Their best zero-miss false-alarm rate was achieved by their 3-cell LSTM, with a value of 0.33, however, the equal error-rate degraded to 0.165. Both LSTMs outperformed the GRU model.

Maheshwary et al. [15] introduced Deep Secure, a neural network-based approach for keystroke dynamics user authentication. Deep Secure is a fully connected neural network comprised of three hidden layers of 100, 400 and 100 dimensions respectively. The higher size of hidden layers introduces sparsity and helps in capturing the inter-feature relations, eliminating the need for manual feature selection or engineering. They employed batch normalization, dropout, and used the Leaky ReLU function with Adam as the optimizer. They achieved an EER of 0.30 on the CMU dataset.

Çeker and Upadhyaya [2] proposed transfer learning to transfer acquired knowledge of keystroke dynamics with the objective of authenticating an enrolled user under different environmental conditions with the least amount of re-training. They used three techniques: Adaptive SVM, Deformable Adaptive SVM and Projective Model Transfer SVM. Transfer learning performed up to 13% better on the CMU dataset compared with the classifier trained from scratch. In another work, Çeker and Upadhyaya [3] applied deep learning, specifically, a convolutional neural network on three different datasets, including the CMU dataset. They also augmented the keystroke dataset using Gaussian data augmentation techniques [14, 17]. Their CNN is composed of a single convolutional layer, employing a ReLU function, followed by a max-pooling layer, then a fully connected layer with dropout. They achieved an EER of 0.023.

This section presented an overview of various works proposed by researchers and tested on the CMU dataset. The EER rates and ZM-FAR rates were mentioned (where applicable). The results demonstrate acceptable EER rates, however, the ZM-FAR rates were unacceptably high, rendering the systems proposed unusable in any real-life scenario. These results show that an improved system, one that can achieve low EER and ZM-FAR rates, is desirable, and an open research area.

3. Methodology

In this section, we outline our methodology for keystroke dynamics user authentication. First, we present the dataset used in our experiments, followed by our data preprocessing techniques. Then, we present our deep neural network approach for detecting unauthorized users.

Table 2: CMU Dataset: Detailed features

Feature	Type	Content
subject	text	Unique ID for each user, e.g. s002
sessionIndex	integer	The session number the user entered the password in, ranges from 1-8
rep	integer	The repetition number of the password in each session, ranges from 1-50
H.period	real	Hold time for the *period* key
DD.period.t	real	KeyDown-Keydown time: *period* key to *t* key
UD.period.t	real	KeyUp-Keydown time: *period* key to *t* key
H.t	real	Hold time for the *t* key
DD.t.i	real	KeyDown-Keydown time: *t* key to *i* key
UD.t.i	real	KeyUp-Keydown time: *t* key to *i* key
H.i	real	Hold time for the *i* key
DD.i.e	real	KeyDown-Keydown time: *i* key to *e* key
UD.i.e	real	KeyUp-Keydown time: *i* key to *e* key
H.e	real	Hold time for the *e* key
DD.e.five	real	KeyDown-Keydown time: *e* key to *5* key
UD.e.five	real	KeyUp-Keydown time: *e* key to *5* key
H.five	real	Hold time for the *5* key
DD.five.Shift.r	real	KeyDown-Keydown time: *5* key to *R* key
UD.five.Shift.r	real	KeyUp-Keydown time: *5* key to *R* key
H.Shift.r	real	Hold time for the *R* key
DD.Shift.r.o	real	KeyDown-Keydown time: *R* key to *o* key
UD.Shift.r.o	real	KeyUp-Keydown time: *R* key to *0* key
H.o	real	Hold time for the *o* key
DD.o.a	real	KeyDown-Keydown time: *o* key to *a* key
UD.o.a	real	KeyUp-Keydown time: *o* key to *a* key
H.a	real	Hold time for the *a* key
DD.a.n	real	KeyDown-Keydown time: *a* key to *n* key
UD.a.n	real	KeyUp-Keydown time: *a* key to *n* key
H.n	real	Hold time for the *n* key
DD.n.l	real	KeyDown-Keydown time: *n* key to *l* key
UD.n.l	real	KeyUp-Keydown time: *n* key to *l* key
H.l	real	Hold time for the *l* key
DD.l.Return	real	KeyDown-Keydown time: *l* key to *return* key
UD.l.Return	real	KeyUp-Keydown time: *l* key to *return* key
H.Return	real	Hold time for the *return* key

3.1 Dataset

The dataset used in this study is the Carnegie Mellon University (CMU) Keystroke Dynamics dataset [11]. Data were collected from 51 people at CMU, each of whom typed 400 repetitions of the same password (.tie5Roanl). The passwords were typed in 8 sessions, where in each session, the user entered 50 repetitions of the password. Various timing features were extracted from the data, and the authors consider the “Enter” key to be part of the password. The authors extracted keydown-keydown (keystroke latency) times, keyup-keydown (flight) times, and hold (dwell) times for all keys in the password. The total number of keys in the password (including “Enter”) is 11, for a total

of 31 timing features. Timing features are in seconds and are represented as floating point numbers. Many of the timing features are correlated: keydown-keydown time is the sum of a hold time and a keyup-keydown time. In Table 2, a detailed list of the dataset features is presented.

3.2 Data Preprocessing

Preprocessing data usually involves cleaning the data, allowing it to be passed into the next stage, feature extraction, where relevant information that aids in classification is extracted. Feature extraction is a difficult process and is usually time consuming. The CMU dataset contains 31 real timing features, as well as two integer features, and the label. First, using the real timing features, the total time to enter each password is calculated. This produces a total of 34 features. Next, we use these features to create 1156 new features using the Deep feature synthesis (DFS) algorithm [10]. The DFS algorithm automates and optimizes the feature engineering process. In DFS, features are derived from relationships among data points in the dataset by using similar mathematical operators, called primitives. In addition, feature creation is a multi-level process, where features may be created utilizing previously created features. For our purposes, we create features via addition and multiplication primitives. These features are then standardized by subtracting the mean and scaling to unit variance. Feature scaling is an important step while preparing data for any machine learning algorithm, as features may have largely differing ranges. The "subject" feature is treated as the label, where the text is encoded into ordinal integers.

The dataset contains 20400 rows for 51 users. We follow the process described by Killourhy et al. [11], where we split the dataset into 51 datasets, each containing the data for one user, i.e., 400 rows. This represents the negative class in our dataset (not an attacker). Next, we select the first 5 rows from the remaining 50 users as the positive class (attacker). The 51 datasets now contain 650 rows each, and are imbalanced, with 400 negative class observations, and 250 positive class observations. We could have selected 8 rows from the remaining 50 users, but that would deviate our dataset from the process by Killourhy et al. [11]. Instead, we balance our data using the Synthetic Minority Oversampling Technique (SMOTE) [4]. SMOTE is an oversampling technique used to increase the observations in the minority class. It generates synthetic observations by detecting nearest neighbors in the feature space of the target, then selects similar samples and randomly changes a column at a time in the feature space of the neighboring samples. Following this step, we have 51 datasets, of size 800×1156.

The preprocessed data for all 51 datasets is split into two parts: 75% training set, and 25% testing set. The training set is further split into 80% training set and 20% validation set. The final step in the data preprocessing process is reshaping the input into a format suitable for our model, which is presented in the next section.

3.3 Convolutional Neural Network Model: CNN-Detect

In this section, we discuss our deep neural network and the methods implemented to improve the recognition performance on the CMU dataset. We use a convolutional neural network (CNN) as our model, and we train it on the 51 datasets outlined in Section 3.2. We call our model CNN-Detect.

Input Layer:

The input layer in CNN-Detect is of size 34×34. One input row in the dataset is of size 1156. We transform each row into a square matrix of size 34×34, to fit our input layer. This matrix can be thought of as the input image into our convolutional neural network.

Hidden Layers:

A CNN has a number of hidden layers, such as convolutional layers, pooling layers, and fully connected layers. We note our small dataset size, 600 training samples (120 of which are used for validation) and 200 testing samples. This implies that a small network size should be sufficient to learn all the needed parameters for our classification purposes. Our model is composed of three convolutional layers, where each layer performs a batch normalization operation after a Leaky ReLU activation function. This architecture was inspired by a number of CNN architectures in the literature for similarly sized images. Our input image is 34×34 pixels and so a small filter size of 2×2 is appropriate in this case, as is a small stride, we have stride $= 1$. The number of features maps in the first, second, and third convolutional layers is 8, 16, and 32, respectively. Increasing the number of features maps as the network deepens strengthens the representational power of the network. In addition, zero padding is applied in all convolutional layers. Zero padding is used to overcome the problems of image shrinkage and information loss around the perimeter of the image as the filter is passed over the image.

Each activation map i is calculated as shown in Equation 1, where l is the current layer, $B_i^{(l)}$ is a bias matrix, $k^{(l-1)}$ is the number of kernels used in the previous layer, W is the current layer kernel matrix, and $Y^{(l-1)}$ is the output of the previous layer. Our nonlinearity is the Leaky ReLU function, defined as shown in Equation 2, where we set $\alpha = 0.2$.

$$Y_i^{(l)} = B_i^{(l)} + \sum_{j=1}^{k^{(l-1)}} W_i^{(l)} Y_j^{(l-1)} \quad (1)$$

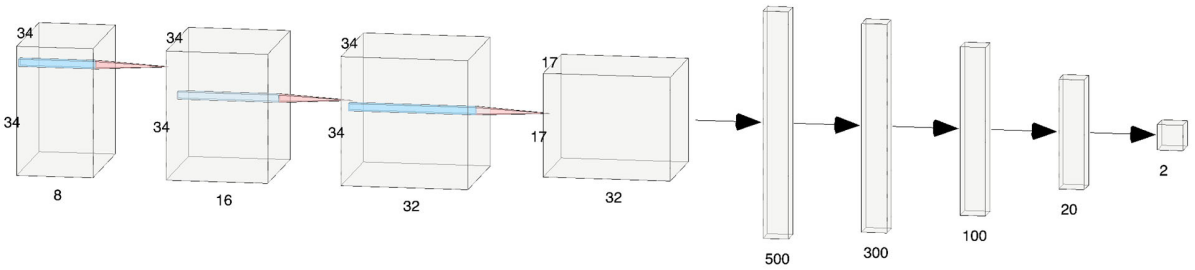

Figure 1: Convolutional Neural Network for Keyword Dynamics Recognition

$$Leaky - ReLU(x) = \begin{cases} \alpha x & x < 0 \\ x & x \geq 0 \end{cases} \quad (2)$$

The convolutional layers are followed by a max-pooling layer with a $2 * 2 * 1$ window size, and a stride of 2, resulting in a $16 * 16 * 32$ layer. Max pooling is useful in reducing overfitting, by providing a more abstract view of the data, as well as reducing computational requirements. The tensor is then flattened into a 9248 neuron layer, followed by four other fully connected layers, with sizes 500 300, 100 and 20, respectively. Fully connected layers aid in the final classification by combining all the signals into one cohesive framework. All fully-connected layers use a 30% dropout rate to reduce overfitting [16], set experimentally. In addition, each fully connected layer implements Ridge Regression, also known as L2 regularization, which adds the squared magnitude of coefficients as a penalty term to the loss function, defined in Equation 3, where $\lambda = 0.0008$.

$$Cost_i = Y_i^{(l)} + \lambda \sum_{j=1}^{k^{(l-1)}} W_i^{(l)} \quad (3)$$

Output Layer:

The output layer is a 2 class Sigmoid layer: one class for the authorized user, and the other for the unauthorized user. The Sigmoid layer uses Equation 4 as the activation function.

$$Sigmoid = \frac{1}{1 + e^{-x}} \quad (4)$$

Optimization:

In our model, we tested two optimizers: Stochastic Gradient Descent and Adam [12], and selected Adam as it was found to work better. In our model, we set the learning rate to $lr = 0.001$, set experimentally. The loss for CNN-Detect is calculated using the binary–cross entropy loss (Log loss), as in Equation 5, where N is the number of samples, y is the label and $p(y)$ is the predicted probability of the label. A summary of our model is shown in Figure 1.

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\big(p(y_i)\big) + (1 - y_i) \cdot log(1 - p(y_i \quad (5)$$

4. Experimental Results

In this section, we present the evaluation metrics and the evaluation results for CNN-Detect on the CMU dataset.

4.1 Evaluation Metrics

The receiver operating characteristic (ROC) curve is generally used to evaluate biometric systems. The ROC measures the tradeoff between the false acceptance rate (FAR) and the false rejection rate (FRR). FRR is also known as Type I Error and is the rate at which the system rejects a sample provided by a genuine user, and so a small FRR indicates a small number of genuine users rejected. FAR, also known as Type II Error, is the rate of incorrectly accepting an erroneous or false sample. In other words, the system accepts impostors who should be rejected. FAR errors are the most dangerous, thus, they are the most important to avoid in a biometrics system. Low FAR values indicate that imposters are rarely successful at accessing resources. The overall goal is to obtain low numbers for each type of error.

To describe system performance overall and to compare different biometric systems, the equal error rate (EER) is used. The EER is sometimes referred to as the crossover error rate (CER) and is defined as the point at which both FAR and FRR are equal. A lower EER indicates better system performance. The zero-miss false alarm rate (ZM-FAR) is a threshold that measures system performance when the miss rate is zero. In other words, when the system threshold for accepting users is high enough to block all

imposter attempts, it provides the rate of authentic users who are denied access. A low ZM-FAR rate is very desirable, as it measures the system performance while preventing unauthorized users from accessing the system. Figure 2 shows the relationship between FAR, FRR and EER.

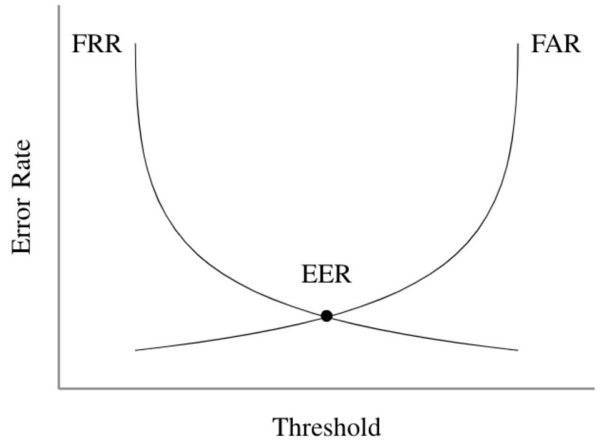

Figure 2: The Relationship between FAR, FRR, and EER

4.2 Performance Evaluation

Our experiments are designed to evaluate the ability of CNN-Detect at detecting unauthorized user access. We use the 51 datasets that we prepared in Section 3.2 from the CMU dataset. We compare our results with other classification algorithms in the literature, presented previously in Section 2. All models were trained on 75% of each dataset (with 20% used for validation), and results are reported on the test set, which is 25% of the dataset. The training set has 480 records, the validation set has 120 records, and the testing set has 200 records. We create 51 models, one for each user, and report the EER and ZM-FAR for each user in Table 3. For 16 users (s002, s007, s011, s012, s016, s017, s018, s020, s027, s028, s033, s035, s041, s043, s047, s053), CNN-detect was able to correctly classify all users as authentic or imposters. The ZM-FAR rates are reported in the third and sixth columns, and are very low, indicating excellent performance by CNN-Detect at blocking unauthorized access.

We present detailed CNN-Detect confusion matrix results for each user in Table 4. The false negative counts in the table show the number of users incorrectly classified as authentic, while they are actually imposters or attackers. In other words, it shows the number of users who were granted access incorrectly. The low counts show that CNN-Detect is able to differentiate users well. The highest counts can be seen for users s025 and s031, whose patterns were not sufficiently unique for the network to classify correctly.

Table 3: Detailed CNN-Detect results for each user

User	EER	ZM-FAR	User	EER	ZM-FAR
s002	0.000	0.000	s032	0.031	0.000
s003	0.010	0.000	s033	0.000	0.000
s004	0.031	0.000	s034	0.031	0.000
s005	0.020	0.000	s035	0.000	0.000
s007	0.000	0.000	s036	0.010	0.000
s008	0.010	0.000	s037	0.020	0.000
s010	0.010	0.000	s038	0.010	0.031
s011	0.000	0.000	s039	0.010	0.010
s012	0.000	0.000	s040	0.010	0.000
s013	0.020	0.031	s041	0.000	0.000
s015	0.010	0.000	s042	0.020	0.020
s016	0.000	0.000	s043	0.000	0.000
s017	0.000	0.000	s044	0.010	0.000
s018	0.000	0.000	s046	0.010	0.000
s019	0.000	0.031	s047	0.000	0.000
s020	0.000	0.000	s048	0.031	0.000
s021	0.000	0.051	s049	0.010	0.000
s022	0.000	0.031	s050	0.020	0.082
s024	0.010	0.000	s051	0.020	0.000
s025	0.010	0.041	s052	0.010	0.000
s026	0.010	0.000	s053	0.000	0.000
s027	0.000	0.000	s054	0.020	0.000
s028	0.000	0.000	s055	0.010	0.000
s029	0.031	1.000	s056	0.010	0.000
s030	0.000	0.010	s057	0.000	0.010
s031	0.000	0.020			

Table 4: Detailed CNN-Detect confusion matrix results for each user

User	TP	FP	FN	TN	User	TP	FP	FN	TN
s002	102	0	0	98	s032	102	3	0	95
s003	102	1	0	97	s033	101	0	1	98
s004	102	3	0	95	s034	102	3	0	95
s005	102	2	0	96	s035	102	0	0	98
s007	102	0	0	98	s036	102	1	0	97
s008	102	1	0	97	s037	102	2	0	96
s010	102	1	0	97	s038	100	1	2	97
s011	102	0	0	98	s039	100	1	2	97
s012	102	0	0	98	s040	102	1	0	97
s013	101	2	1	96	s041	100	0	2	98
s015	102	1	0	97	s042	101	2	1	96
s016	102	0	0	98	s043	102	0	0	98
s017	102	0	0	98	s044	102	1	0	97
s018	102	0	0	98	s046	102	1	0	97
s019	100	0	2	98	s047	101	0	1	98
s020	102	0	0	98	s048	102	3	0	95
s021	101	0	1	98	s049	102	1	0	97
s022	101	0	1	98	s050	100	2	2	96
s024	102	1	0	97	s051	102	2	0	96
s025	99	1	3	97	s052	102	1	0	97
s026	102	1	0	97	s053	102	0	0	98
s027	102	0	0	98	s054	102	2	0	96
s028	102	0	0	98	s055	102	1	0	97
s029	101	3	1	95	s056	102	1	0	97
s030	101	0	1	98	s057	101	0	1	98
s031	99	0	3	98					

We compare CNN-Detect to other models in the literature in Table 5. As can be seen in the second column, EER rate, CNN-Detect is better than all other models. CNN-Detect outperforms all models in the literature by a large margin when it comes to its ZM-FAR rate, indicating its applicability as a component in a user authentication system as a keyboard dynamics secondary authentication measure, with the first being the password entered itself.

Table 5: Average equal-error rates and average zero-miss false-alarm rates

Detector	EER	ZM-FAR
Manhattan (scaled) [11]	0.096	0.601
Nearest Neighbor (Mahalanobis) [11]	0.100	0.468
Nearest Neighbor with outlier removal [18]	0.084	0.405
DNN [5]	0.035	N/A
LSTM (2 cells) [13]	0.136	0.379
Deep Secure [15]	0.030	N/A
CNN [3]	0.023	N/A
MINIBAG [8]	0.056	N/A
AR-ONENB [8]	0.066	0.700
DC [9]	0.077	0.358
k-NN [9]	0.078	0.377
CNN-Detect	**0.009**	**0.027**

5. Conclusion

In this paper, we have studied the characteristics of keystroke dynamics as a behavioral biometric for user authentication using traditional PC keyboards on the CMU dataset. We have introduced a successful feature engineering approach for the CMU dataset that enabled our proposed classifier, CNN-Detect, to achieve excellent results at classifying imposters and authentic users.

CNN-Detect was trained using both positive samples from the genuine user and samples from background users, labelled as imposters, resulting in enhanced discriminative power. 51 CNN-Detect models were trained on the 51 datasets created from the original CMU dataset, and the individual and average scores of the models reported. Specifically, the equal error rate (EER) and zero-miss false alarm rate (ZM-FAR) were reported. Our model outperformed all other models in the literature by a large margin and was able to achieve that in both measures. The low ZM-FAR rate was very low, meaning CNN-Detect is suitable as part of a pipe-lined system for user authentication. CNN-Detect can be periodically retrained (using transfer learning) to capture variations of users' typed text over time. For future research, CNN-Detect may be integrated and deployed in a user authentication system. In addition, the applicability of CNN-Detect to authentication of free typed text should be studied.

References

[1] Salil P Banerjee and Damon L Woodard. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research*, 7(1):116–139, 2012.

[2] Hayreddin Çeker and Shambhu Upadhyaya. Adaptive techniques for intra-user variability in keystroke dynamics. In *2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–6. IEEE, 2016.

[3] Hayreddin Çeker and Shambhu Upadhyaya. Sensitivity analysis in keystroke dynamics using convolutional neural networks. In *2017 IEEE Workshop on Information Forensics and Security (WIFS)*, pages 1–6. IEEE, 2017.

[4] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[5] Yunbin Deng and Yu Zhong. Keystroke dynamics user authentication based on gaussian mixture model and deep belief nets. *International Scholarly Research Notices*, 2013, 2013.

[6] Yunbin Deng and Yu Zhong. Keystroke dynamics user authentication using advanced machine learning methods. *Recent Advances in User Authentication Using Keystroke Dynamics Biometrics, GCSR*, 2:23–40, 2015.

[7] Clayton Epp, Michael Lippold, and Regan L Mandryk. Identifying emotional states using keystroke dynamics. In *Proceedings of the sigchi conference on human factors in computing systems*, pages 715–724, 2011.

[8] Jiacang Ho and Dae-Ki Kang. Mini-batch bagging and attribute ranking for accurate user authentication in keystroke dynamics. *Pattern Recognition*, 70:139–151, 2017.

[9] Elena Ivannikova, Gil David, and Timo Hämäläinen. Anomaly detection approach to keystroke dynamics based user authentication. In *2017 IEEE Symposium on Computers and Communications (ISCC)*, pages 885–889. IEEE, 2017.

[10] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015,*

Paris, France, October 19-21, 2015, pages 1–10. IEEE, 2015.

[11] Kevin S Killourhy and Roy A Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*, pages 125–134. IEEE, 2009.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Paweł Kobojek and Khalid Saeed. Application of recurrent neural networks for user verification based on keystroke dynamics. *Journal of telecommunications and information technology*, (3):80–90, 2016.

[14] John Leggett, Glen Williams, Mark Usnick, and Mike Longnecker. Dynamic identity verification via keystroke characteristics. *International Journal of Man-Machine Studies*, 35(6):859–870, 1991.

[15] Saket Maheshwary, Soumyajit Ganguly, and Vikram Pudi. Deep secure: A fast and simple neural network based approach for user authentication and identification via keystroke dynamics. In *IWAISe: First International Workshop on Artificial Intelligence in Security*, page 59, 2017.

[16] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[17] Deian Stefan, Xiaokui Shu, and Danfeng Daphne Yao. Robustness of keystroke-dynamics based biometrics against synthetic forgeries. *computers & security*, 31(1):109–121, 2012.

[18] Yu Zhong, Yunbin Deng, and Anil K Jain. Keystroke dynamics for user authentication. In *2012 IEEE computer society conference on computer vision and pattern recognition workshops*, pages 117–123. IEEE, 2012.

Najwa Altwaijry is an Assistant Professor of Computer Science at King Saud University. She received her PhD degree in 2014 from the College of Computer Sciences at King Saud University. Her research interests include machine learning, swarm intelligence, evolutionary computation, cyber security and bioinformatics.