BeCAPTCHA-Type: Biometric Keystroke Data Generation for Improved Bot Detection

Daniel DeAlcala[1], Aythami Morales[1], Ruben Tolosana[1], Alejandro Acien[1], Julian Fierrez[1], Santiago Hernandez[1], Miguel A. Ferrer[2], Moises Diaz[2]
[1]Biometrics and Data Pattern Analytics Lab, Universidad Autonoma de Madrid, Spain
[2]University Las Palmas Gran Canaria, Spain

Abstract

This work proposes a data driven learning model for the synthesis of keystroke biometric data. The proposed method is compared with two statistical approaches based on Universal and User-dependent models. These approaches are validated on a bot detection task, using the keystroke synthetic data to improve the training process of keystroke-based bot detection systems. Our experimental framework considers a dataset with 136 *million keystroke events from* 168 *thousand subjects. We have analyzed the performance of the three synthesis approaches through qualitative and quantitative experiments. Different bot detectors are considered based on several supervised classifiers (Support Vector Machine, Random Forest, Gaussian Naive Bayes and a Long Short-Term Memory network) and a learning framework including human and synthetic samples. The experiments demonstrate the realism of the synthetic samples. The classification results suggest that in scenarios with large labeled data, these synthetic samples can be detected with high accuracy. However, if the proposed synthetic data is nor properly modelled using massive data by bot detectors, then that data will be very difficult to detect even for the most sophisticate bot detectors. Furthermore, these results show the great potential of the presented models for improving the training of bot detection technology.*

1. Introduction

The use of Artificial Intelligence (AI) in cyberattacks is an important concern for our society [16]. Along with the massive use of the Internet, the usage of bots to access digital services and platforms has grown, being the detection of these bots an open challenge with a high worldwide economical impact [33]. The rapid development of generative models during the last decade has allowed to synthesize realistic images and videos [32, 38], audio [37], or text data [9]. These technologies can be integrated in new generations of bots with realistic human-like behavior. As an example, the language generation models developed within the last two years have made almost impossible to distinguish between human and bot conversation. In this context, biometric technologies appear as a solution to distinguish between human and synthetic behaviors [22].

Biometric recognition is the ability to authenticate a person with the highest possible reliability based on their physical characteristics or behavioral attributes [19]. This technology can be used to uniquely recognize one user among others (e.g., user identification), to recognize groups of subjects (e.g., soft-biometrics classification), or finally to differentiate real users from non-real users (e.g., bot detection). This work focuses on the topic of bot detection, more precisely in the generation and detection of synthetic keystroke patterns. Keystroke biometrics play an important role in bot detection due to its suitability in digital environments. Keyboards and touchscreens are among the most common human-machine interfaces nowadays, and their use in digital platforms and services is almost universal.

In bot detection, a platform/system must detect bot attacks and differentiate them from legitimate user's interactions. Traditionally, this detection has been carried out with conventional CAPTCHAs, which ask the user to perform some cognitive challenges. Most common conventional CAPTCHAs are: *i)* Recognize characters in a distorted image; *ii)* Identify a specific class in a set of images; and *iii)* Analyze the interaction and web traffic.

Traditional CAPTCHAs are becoming less effective due to advances in Computer Vision and the image classification approaches based on Deep Learning. As a result, other less intrusive and more effective CAPTCHAs are being developed nowadays based on the interaction information between the human/bot and the platform without actively requesting any information [1, 2]. The behavioral biometric characteristics, and specially the so-called web-biometrics [15], play an important role in this interaction modelling. These biometrics characteristics include keystroke dynamics, mouse dynamics, and mobile interaction, among others.

In this work we propose three synthetic keystroke data generation methods with application to bot detection. The main contributions of this work can be summarized as follow:

- Three approaches for the synthesis of keystroke dynamics data based on Universal, User-dependent, and Generative Models. The first two proposed approaches are based on the statistical modelling of the biometric keystroke dynamics features of 100,000 subjects. The third approach is based on a Generative Neural Network. In this work we demonstrate that data-driven learning approaches can be used to generate realistic keystroke dynamics, opening a new way to synthesize and detect keystroke samples.
- A bot detection method based on keystroke dynamics using algorithms trained with human and synthetically generated samples.
- A comprehensive performance analysis including: *i)* amount of data available to train the bot detector; *ii)* type of synthetic data used to model the human behavior; *iii)* input text dependencies.

The rest of the work is organized as follows: Section 2 summarizes the related literature. Section 3 presents the proposed synthesis approaches. Section 4 describes the bot detection method. Section 5 presents the experimental results of the bot/human classification methods trained with the synthetic and human samples. Finally, in Section 6 we present the conclusions and limitations.

2. Related Literature

Keystroke dynamics has been widely focused on user recognition (i.e., differentiate one user from others). In 1980, a pioneer study of this biometric trait was made demonstrating that it is possible to differentiate subjects according to their typing patterns [14]. In general, keystroke biometrics are commonly divided into two different approaches [8]: free-text and fixed-text. Fixed-text approaches usually outperform free-text ones in terms of performance due to its lower intra-class variability. Nevertheless, the transparency and no restrictions of free-text approaches represent a clear advantage in most applications.

During the last decades, the performance of keystroke biometric user recognition approaches has improved to reach the actual state of the art. Some classic approaches (before the deep learning revolution) include non-elastic sample alignment (e.g., Dynamic Time Warping [30]), scaled Manhattan distances [26], and statistical models (e.g., Hidden Markov Models [6]). The performance of these approaches varies depending on the characteristics of the database and experimental protocol, but in general, Equal Error Rates (EER) over 5% were consistently reported. During those years, the performance of free-text approaches was far from the performance achieved by fixed-text methods [7, 31]. More recently, the release of new large-scale datasets and the use of Deep Neural Networks have boosted the performance of free-text keystroke biometrics with EERs under 5% [3, 28, 35].

The improvement of keystroke biometric technologies opens the doors to new applications apart from the traditional user recognition. One of these applications is bot detection. Bot detection presents some differences with respect to user authentication. While user authentication approaches are developed to model user-specific characteristics, bot detection approaches model the general population characteristics. The final aim is to extract the characteristics of human's keystrokes dynamics and differentiate them from bots.

Before getting into bot detection, we must talk about the synthesis of keystroke data generated by bots. One of the first studies was presented more than 10 years ago in [34] creating a synthetic database with 20 subjects using first-order Markov chains. An improved keystroke biometric attack generator was presented in [27], using a Linguistic Buffer and Motor Control model. The use of synthetic keystroke samples to study the vulnerability of keystroke biometric systems was also studied in [17, 23–25]. Those studies have proposed methods using higher-order contexts and empirical distributions to generate impersonation attacks (i.e., samples generated to confuse the identity of a specific user). The conclusions from previous studies suggest that it is possible to generate realistic keystroke data.

Regarding keystroke bot detection itself, there is a pioneer work involving the use of function calls analysis [4]. The system proposed in [4] was based on communication protocol analysis (frequency of keyboard logs) rather than keystroke dynamics modelling. **Our work explores novel synthesis approaches based on keystroke dynamics for the development of new bot detection methods.**

A lot of research is currently being done in statistical synthesis of keystrokes, creating synthetic samples used both for attacking systems and training bot detectors [5, 17, 23–25, 34]. In this work we specially focus on papers [5] and [34] for the comparison with our proposed methods as they cover the topic of bot detection whereas the rest are more focused on the creation of synthetic samples and not that much in competitive bot detection. In [5] the authors classified bots using Euclidean distance between human and bot features. In [34] they used a Support Vector Machine (SVM) classifier trained with real and synthetic samples.

As our last connection with related works, in this paper we will exploit recent technology for statistical synthesis of functions (applicable in general beyond keystroking and other biometrics) using Generative Neural Networks [18, 21, 36], and based on that we will develop a novel Gen-

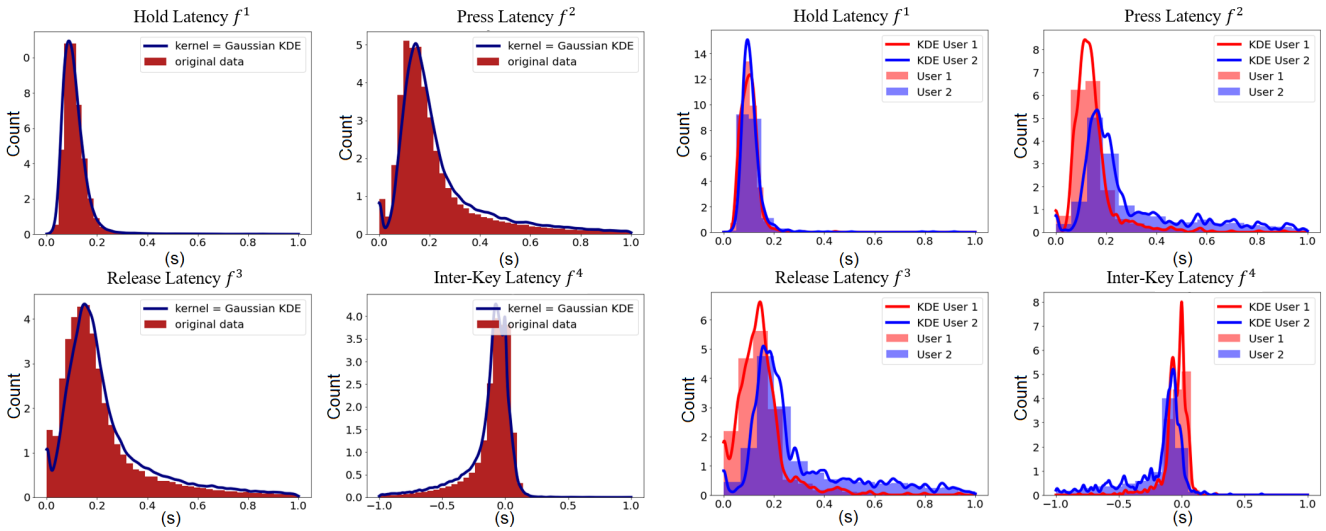

Figure 1. 4 left images: Original human data (bars) and Kernel Density Estimator fitting model (continuous line) for the whole Dhakal dataset. 4 right images: the same for two independent subjects.

erative Neural Network to improve the detection of synthetic keystroke data.

2.1. Keystroke Dynamics Dataset

The Dhakal Dataset [10] is considered in this study to develop the models able to synthesize large-scale keystroke biometric data. There are 168,000 subjects and 136 million keystrokes in the database. Regarding the acquisition procedure, each subject had to learn a sentence and then write it as fast as possible (semi-fixed text scenario) using their own keyboard. Each subject has 15 sentences with a minimum of 3 words and a maximum of 70 characters.

Following the traditional keystroke dynamics modelling, the dataset is processed to extract 4 time features derived from the two main typing events (key press and key release) and the ASCII code for each key pressed [3]:

1. Hold Latency (f_j^1): Time between the key j is pressed and released.

2. Press Latency (f_j^2): Time between two consecutive keys are pressed, j and $j+1$.

3. Release Latency (f_j^3): Time between two consecutive keys are released, j and $j+1$.

4. Inter-Key Latency (f_j^4): Time between a key j is released and the next key $j+1$ is pressed.

5. Key Code (f_j^5): ASCII code normalized between 0 and 1 for each key j.

Figure 1 (4 left subfigures) presents the distribution of the 4 initial (time) features for the complete dataset. As can be seen, the Hold Latency feature is close to a normal distribution. The rest of the features presents tails related to the characteristics of the typist and the key pressed (some combinations of keys usually present larger timing than others). Note that the Inter-Key Latency presents negative times, i.e., the next key is pressed before the currently one is released. This effect is called rollover-typing and it is common in keystroke recognition systems.

3. Keystroke Synthesis: General Outlook

This section presents our three keystroke dynamic data synthesis methods. There are two different approaches: the first one is based on the statistical modelling of the feature distribution of the keystroke time series, while the second is based on a data-driven learning approach with a Generative Neural Network. As described in the previous section, the keystroke dynamic features model the biometric patterns during a typing task as differences of times (i.e., time gaps between key press and key release events). We propose to model the probability distribution of the 5 biometrics features i defined before based on a training sequence of consecutive keystrokes $\mathbf{f}^i = [f_0^i, ..., f_N^i]$, where N is the total number of samples used for training the model. To train the following methods, the N samples used are from 100,000 subjects out of the 168,000 in the database.

3.1. Statistical Generative Models

For the statistical approach we use the Kernel Density Estimator algorithm (KDE) [20]. KDE is a nonparametric algorithm that estimates univariate or multivariate densities. KDE allows to compute the density of keystroke biometrics features as a set of functions $\mathrm{F} = \{\mathrm{F}^1, \mathrm{F}^2, \mathrm{F}^3, \mathrm{F}^4\}$, here the

KDE approximates the probability that the feature i takes the value x as:

$$\mathrm{F}^i(x) = \frac{1}{N}\sum_{j=1}^{N} K(x - f_j^i; \sigma) \quad (1)$$

where K is the kernel function (Gaussian in our experiments) and σ is the bandwidth ($\sigma = 1.0$ in our experiments). We use this method to model the probability distributions F of the keystroke biometric features in the Dhakal Dataset (see Figure 1 continuous lines). The synthesis of keystroke dynamic samples is then divided into: 1) generation of a sequence of K keys representing the typed text: $\mathbf{k} = [k_0, ..., k_K]$; 2) generation of the corresponding keystroke biometric features $\mathbf{f}^i$ as random samples from the the learned models F (random sampling serves to introduce human-like variability between samples); and 3) the calculation of a sequence of timestamps: $\mathbf{t}' = [t'_0, ..., t'_{2\times K}]$ associated to the key press and key release events. The timestamp vector $\mathbf{t}'$ can be easily obtained from the time features $\mathbf{f}^1$ and $\mathbf{f}^4$. The following equations show the calculation of timestamps for the first two keys:

$$t'_0 = 0, t'_1 = t'_0 + f_1^1 \rightarrow \text{(key 1)} \quad (2)$$

$$t'_2 = t'_1 + f_1^4, t'_3 = t'_2 + f_2^1 \rightarrow \text{(key 2)} \quad (3)$$

We propose two synthesis approaches depending on the information used to model the feature distributions: Universal Model or User-dependent Model.

3.1.1 Statistical Approach 1: Universal Model

The Universal Model is based on the estimation of a unique set of KDE functions F representing the behavior of all subjects in the Dhakal dataset. As a result, only 4 KDEs are necessary to model the human typing behavior distributions. Figure 1 shows the set of trained functions (continuous lines in the four images on the left). In general, this approach could approximate in a good way the human features as a group. However, it could also generate unnatural samples due to the combination of timing across different subjects. It is important to highlight that this universal synthesis approach is not able to model: the intra-user dependencies (i.e., each user has certain biometric features and a correlation between them [13]) or the key-dependent features (i.e., each key has a typing pattern depending on itself and also to a certain extent on the previous and following keys). First, the keystrokes from real human samples are parameterized according to the proposed time features. Second, the probability function F^i of each time feature is independently modeled according to a KDE function (see Eq. 1). The four trained KDEs are then used to generate new keystroke dynamic features $\mathbf{f}'$ from which the synthetic keystroke timestamps $\mathbf{t}'$ are obtained (see Eqs. 2 and 3).

3.1.2 Statistical Approach 2: User-dependent Model

The fundamental principle of keystroke biometric recognition systems is that typing patterns vary from one user to another. The User-dependent generation method tries to incorporate these intra-user characteristics [29] into the synthesis process. However, the data available for each user is usually very limited [12], therefore, depending on the amount of data available to model each user [11], User-dependent Models could be less accurate than the previous Universal Model. The user-dependent synthesis approach is aimed to model the statistics of each feature i for every subject u. Even so, with this model we are still not able to model the key-dependent features. First, keystroke samples from the Dhakal dataset data are divided by subjects. Second, the keystroke timestamps $\mathbf{t}^u$ from user u are parameterized to obtain the four time feature sequences $\mathbf{f}^{u,i}$. Then, the probability distribution of each time feature from each training user ($\mathrm{F}^{u,i}$) is independently modeled according to a KDE function (i.e., four $\mathrm{F}^{u,i}$ per user). This process is repeated for U different human subjects in the database. Finally, the $4 \times U$ models are used to generate synthetic feature vectors and their corresponding synthetic keystroke timestamps.

3.2. Generative Neural Network Model

We propose a novel Generative Neural Network (GNN) for the synthesis of keystroke time series. The synthesis of time series with specific statistical distributions using Generative Neural Networks is an open challenge in the literature [18, 21, 36]. If we want the GNN to learn a specific distribution then we can not use a standard loss function because standard losses are not optimized to learn distributions [28]. For example, if we use the regression function Mean Square Error (MSE) for each different key-code, the network will learn to give always the same value (deterministic output), the one that appears more in the distribution to minimize losses (e.g., in a Gaussian distribution the output will be the mean).

The aim of our GNN model is to learn the required parameters to synthesize realistic keystroke biometrics features with realistic intra-class and inter-class variability. The input of the model is a key-code and the GNN generates different human-like times for this specific key-code. We train a specific GNN model for each time feature ($\mathbf{f}^i$) thus obtaining a set of 4 functions $\mathrm{G} = \{\mathrm{G}^0, \mathrm{G}^1, \mathrm{G}^2, \mathrm{G}^3\}$ of the following form:

$$\mathrm{G}^i(k) = \mathrm{GNN}(k; W^i) \quad (4)$$

where k is the key code and GNN is a neural network with its weights (W^i). During the training process the key-codes are introduced as input, and the network learns the time distributions for each $\mathbf{f}^i$ from the real data. During the inference process, the code is introduced as input and the net-

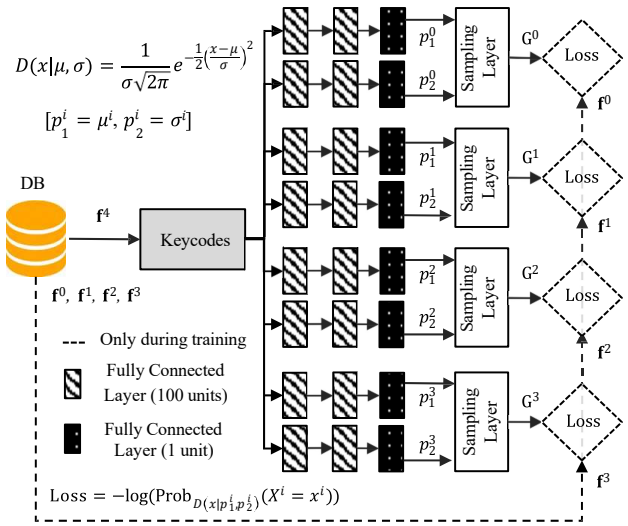

Figure 2. Proposed Generative Neural Network learning framework. Example based on a Gaussian distribution defined by μ and σ parameters.

works generate the 4 time features. The proposed GNN computes the required parameters for the distribution of each key-code and then randomly samples this distribution. Figure 2 shows an example of our proposed GNN learning framework based on a Gaussian distribution with parameters μ and σ. The proposed architecture is based on: two fully-connected layers with 100 units each (tanh activation function), one fully-connected layer with 1 unit (linear activation function) and one sampling layer (this layer creates the probability density function with the output of the previous layer and samples it).

The training process of the GNN is designed to learn the parameters of the statistical distributions of the features $\mathbf{f}^i$. In our experiments, each time feature ($\mathbf{f}^i$) is modelled by a parametric function defined by q parameters ($\mathbf{P}^i = [p_1^i, ..., p_q^i]$). For example, a Gaussian distribution can be modelled by two parameters: mean and variance ($\mathbf{P}^i = [\mu^i, \sigma^i]$). The loss function used computes the probability that feature i (i.e. X^i) takes a specific real value x^i from the training pool of data according to the distribution (D) generated with the parameters $\mathbf{P}^i = [p_1^i, ..., p_q^i]$ learned up to that training moment:

$$\text{Loss} = -\log\left(\text{Prob}_{D(\mathbf{x}|\,\mathbf{P}^i)}(X^i = x^i)\right) \quad (5)$$

This loss function acts as the Likelihood Function of the distribution we want to learn.

For the proposed Generative Neural Network we only consider a Universal Method since for a User-dependent one we would need a large number of samples from each subject (the Dhakal database is large in number of subjects but not in samples per subject). Finally, to generate the time series associated to a specific sequence of key-codes, we employed the learned functions G and the strategy presented in Eqs. 2 and 3.

4. Bot Detection Exploiting Synthetic Data

Most bots are not developed to generate realistic keystroke time series. They are usually developed to interact with a web service/platforms and this interaction usually includes introducing text as input (e.g., searching information) [33]. The code of a traditional bot is exclusively focused on the generation of the sequence of keys $\mathbf{k}$ necessary to produce a desired result. This work explores a more challenging scenario where the bot is developed to spoof a keystroke bot detection system, generating human-like keystroke time sequences $\mathbf{t}'$.

We propose the use of synthetic keystroke samples to train improved bot detection systems (see Figure 3) robust even against sophisticate human-like bots. We use the Dhakal Dataset [10] to model the real human keystroke patterns in our experiments (see Section 2.1 for details about the dataset). First, the synthetic samples are generated using the same text from the real ones, i.e., the human and bot key sequences are exactly the sames so that the classifier cannot differentiate them by the key codes. Note that the Dhakal Dataset was captured according to a semi-fixed text protocol. This protocol implies that the text varies for each human sample in most of the cases. Second, the human and synthetic keystroke sequences are truncated to L (L is equal to 30 in our experiments) or if smaller they are discarded. The reason for truncating these sequences is to be able to use the different classifiers that are presented below, under the same conditions. Third, each keystroke sequence is parameterized according to the features f^i presented in Section 2.1. Finally, we evaluate four different classification algorithms: Support Vector Machine (SVM), Recurrent Neural Network based on Long Short-Term Memory (LSTM), Random Forest (RF), and Gaussian Naive Bayes (GNB). These algorithms are trained using both human and bot feature vectors. We describe in Section 5.1 the experimental protocol details. These four classifiers have been chosen since they are some of the most relevant in related literature.

There could be a question whether the use of synthetic samples to train the classifiers is really useful and improves their performance or not. To shed some light on this, we introduced a One-Class Classifier (One-Class SVM) trained exclusively with human samples. With this result it is possible to compare whether, at least in this type of classifier (although it can certainly be extrapolated to the rest), it is being useful to include these samples.

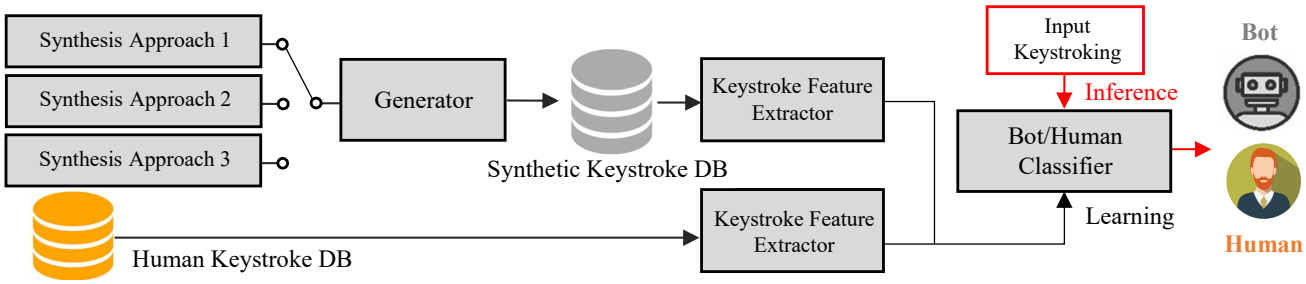

Figure 3. Application of the three proposed keystroke data synthesis approaches to bot detection.

5. Experiments and Results

5.1. Experimental Protocol

We divide the Dhakal database in two sets with 100,000 subjects for training and the remaining 68,000 subjects for testing. Including or not the key-codes in the classifier can have great relevance in the detection of bots since one of our synthesis approaches takes into account the key for the creation of times whereas the others not. As a result, we consider experiments with and without the key-codes to better understand its impact in the performance. In addition, the use of key-codes may be prevented to protect the privacy of subjects. It is therefore interesting to see the performance of the classifiers when this information is available or not.

The scarcity of labeled bot samples is a common challenge in bot detection [33]. For this reason, the experiments are divided into different scenarios depending on the data available to train the bot detector (from 20 subjects to 500). Each subject contains 15 real and 15 synthetic samples. Therefore, training with 20 subjects results in 300 synthetic samples and 300 human samples. All the models are evaluated using the same 500 bot and 500 human samples (15,000 samples in total).

Two main objectives are considered in the analysis. First, the quantitative evaluation of the synthesis methods (**O1**). Second, the evaluation of the keystroke bot detection (**O2**). Previous objectives (**O1**, **O2**) are analyzed depending on the multiple variables mentioned: *i)* the number of bot samples available to train the classifier; *ii)* the classification algorithm (LSTM, SVM, RF, and GNB) and *iii)* the availability of key-codes to train the classifiers.

The experimental protocol comprises two scenarios assuming the availability or not of synthetic data: *i)* Closed Set (Table 1); and *ii)* Open Set (Table 2). In the Closed Set scenario we trained the bot detector with samples generated with the same synthesizer used in test (different samples but same generation method). In the Open Set scenario we train the detector using samples generated with a synthesizer different to the one used for the test samples. This scenario allows to evaluate the generalization capacity of the bot detectors. Everything is evaluated according to the classification accuracy of the model (i.e., human/real vs synthetic/bot).

It is important to note that the aim of the experiments is not to confront the different generation approaches but to analyze how they can be used to detect bots generated with different synthesis approaches. The final aim is to generate synthetic databases that can be used to train these bot detectors.

5.2. Results

The first experiments aim to analyze the capacity of the generation methods to synthesize human-like data (**O1**) and also to solve the question whether including synthetic data in training can improve the classification results and is therefore useful. For this, we focus on the performances obtained by the one-class classifier (OC SVM) and binary SVM classifier (SVM). The OC SVM is trained using only human samples while the binary SVM is trained using both human and synthetic samples. Both classifiers are evaluated using the same bot and human samples. The results are presented in Table 1 (first 4 columns). The OC SVM classifier shows a much lower bot detection accuracy (around 50%) than the binary SVM (between 77% and 100%). On the one hand, the low performance of the OC SVM suggests that synthetic samples present realistic patterns similar to those obtained from real data (using a one-class classification algorithm). On the other hand, the high bot detection accuracy obtained for the binary SVM classifier answers the question about the usefulness of including synthetic samples in training.

The rest of the columns of the Table 1 show the classification accuracy values of the RBF SVM, LSTM, RF, and GNB classifiers trained using the Key-Codes (f_j^5) together with the rest of features (f_j^i, $i \in [1, 4]$) (K=1) and without the key-codes (K=0).

Analyzing again the realism of the synthetic samples (**O1**) comparing the different synthesis methods, the classification accuracy of the models trained with the synthetic samples generated with the user-dependent model are lower than those generated with the GNN model (Table 1). Fur-

		Classification Model									
		OC SVM		SVM		GNB		RF		LSTM	
Gen Model	# Train Subjects	K=0	K=1	K=0	K=1	K=0	K=1	K=0	K=1	K=0	K=1
User-dep	20	0.43	0.44	0.79	0.77	0.65	0.65	0.87	0.88	0.50	0.50
	100	0.54	0.54	0.83	0.79	0.63	0.63	0.92	0.92	0.51	0.51
	500	0.53	0.54	0.93	0.90	0.64	0.64	0.94	0.95	0.93	0.99
Univ	20	0.43	0.44	0.89	0.82	0.68	0.68	0.94	0.94	0.53	0.53
	100	0.55	0.56	0.97	0.98	0.67	0.67	0.98	0.98	0.76	0.79
	500	0.53	0.54	1.00	1.00	0.70	0.70	1.00	1.00	1.00	1.00
GNN	20	0.49	0.47	0.88	0.80	0.68	0.68	0.95	0.95	0.53	0.52
	100	0.52	0.51	0.97	0.97	0.64	0.64	0.98	0.98	0.69	0.60
	500	0.53	0.53	0.99	0.99	0.68	0.68	0.99	0.99	1.00	1.00

Table 1. Bot detection classification accuracy for the different detectors and synthesis methods using a Closed Set. K=0 implies no use of key-codes when training the classifier and K=1 implies the use of key-codes. The detectors are: One-Class Support Vector Machine (OC SVM), Support Vector Machine (SVM), Gaussian Naive Bayes (GNB), Random Forest (RF), and Long Short-Term Memory (LSTM). Accuracy results for evaluation users.

thermore, the GNN model presents in general lower results than the universal one. A conclusion can be drawn from here: from the way in which the synthetic samples are generated and the models trained, it is more relevant a coherence between the different keystroke time features $f^i, i \in [0,4]$ than the coherence between each keystroke time feature $f^i, i \in [0,4]$ and the key-code $f^i, i = 5$.

The following analysis of the bot detection performance (**O2**), is divided according to the number of samples available to train the classifiers (Table 1).

Large (500 subjects): *Universal and GNN methods*: the results suggest that with enough subjects, perfect classification is achieved using SVM, LSTM, and RF. GNB does not achieve high accuracy because this algorithm does not take into account correlations between the different keystroke time features. This also explains why the result is the same with or without the use of key-codes. *User-dependent method*: perfect classification is only achieved with the LSTM classifier and using the key-codes. The LSTM classifier is the one with the highest accuracy when there is enough information to train it. Also, using the key-code information allows to identify better. The SVM and RF methods achieve similar performances while the GNB has the lowest accuracy. This is again due to the fact that this method does not take into account correlations between the different keystroke time features. For the same reason it also has a performance similar to the universal and GNN methods.

Medium (100 subjects): *Universal and GNN methods*: The performance of the LSTM classifier plummets (an average 30 %), the top results are achieved with SVM and RF classifiers. This is because these classifiers do not need as much training as LSTM. *User-dependent method*: The classifier with the best accuracy in this case is RF over SVM (1% to 5% better). The synthetic and real samples are closer together in the multidimensional space used by SVM as they are more similar to each other, so it needs more training to correctly tune the hyperplane that separates them. In this case also the LSTM classifier performs worse than GNB also because the synthesis is more complex and the classifier needs more training.

Limited (20 subjects): For both universal and user-dependent methods, RF offers the highest performance when there is a high sample sparsity. The RF algorithm based on tree decision favors detection with sparse samples.

The use or not of key-codes does not affect the RF and GNB classifiers at all. The SVM classifier perform worse when the key-codes are included in the feature vector. Note that the text used to generate the bot samples was directly extracted from human samples, therefore, the inclusion of the key-codes did not result in an advantage during the bot detection for this classifier. Nevertheless, the LSTM classifier is capable of associating each key with a time and for this reason it detects impostor samples better in Universal and User-dependent models. In the case of the GNN model, these times have been taken into account to create the synthetic samples and therefore it is more difficult for the classifier to detect them.

The generalization ability of the bot detector was presented in Table 2 with a cross database experiment. We focus on the Universal and GNN methods (both methods are user independent so their comparison is fair). In this case, the samples used to train the bot detectors were generated with a method different of the one used to generate the test samples. The results demonstrate that, in general, training with samples synthesized with the GNN model allows to better generalize against unseen synthetic samples of the counter model.

In the last experiment (Table 3) we compare our classifiers with previous state-of-the-art keystroke bot detection

Gen Model			Classification Model									
			OC SVM		SVM		GNB		RF		LSTM	
Train	Test	# Train Subjects	K=0	K=1	K=0	K=1	K=0	K=1	K=0	K=1	K=0	K=1
Univ	GNN	20	0.49	0.48	0.70	0.72	0.63	0.63	0.84	0.82	0.48	0.47
		100	0.54	0.55	0.67	0.70	0.63	0.63	0.68	0.74	0.66	0.65
		500	0.53	0.54	0.59	0.62	0.64	0.64	0.51	0.54	0.99	0.99
GNN	Univ	20	0.49	0.48	0.78	0.68	0.65	0.65	0.88	0.89	0.56	0.58
		100	0.53	0.53	0.94	0.91	0.64	0.64	0.91	0.93	0.69	0.68
		500	0.53	0.54	0.97	0.98	0.64	0.64	0.94	0.95	1.00	1.00

Table 2. Bot detection classification accuracy for the different detectors and synthesis methods using an Open Set. K=0 implies no use of key-codes when training the classifier and K=1 implies the use of key-codes. The detectors are: One-Class Support Vector Machine (OC SVM), Support Vector Machine (SVM), Gaussian Naive Bayes (GNB), Random Forest (RF), and Long Short-Term Memory (LSTM). Accuracy results for evaluation users.

	Gen Model			
	Train	Test	Train	Test
Method	Univ	GNN	GNN	Univ
[5] (Euclidean)	0.49		0.49	
[34] (SVM)	0.62		0.98	
Ours (OCSVM)	0.54		0.54	
Ours (RF)	0.54		0.95	
Ours (GNB)	0.64		0.64	
Ours (LSTM)	**0.99**		**1.00**	

Table 3. Classification accuracy comparison between the proposed approaches and existing methods. The experiments have been carried out assuming a large number of training subjects (500), the use of the key-codes (K=1) and Open Set environment.

approaches [5, 34]. The features used in [5, 34] were similar to the time features employed in our methods. For a fair comparison, we train and evaluate the methods proposed in [5, 34] with the same synthetic and real samples used in our experiments. This comparison has been carried out assuming a large number of training subjects (500), using key-codes and Open-set environment. The results in the table show that the GNN synthetic samples represent a more difficult challenge for the detectors. The detection performance of these samples varies from 49% to 99%. The results suggest that GNN samples can be used to detect synthetic samples generated with a different synthesizer approach. Our LSTM classifier presents the highest detection performance, achieving a 100% bot detection accuracy for both types of synthetic samples.

6. Conclusions and Limitations

In this work we have analyzed the feasibility of using a behavioral trait (dynamic typing) as a passive CAPTCHA where the subject does not need to perform any cognitive challenge in order for the system to determine if this subject is a bot or a human.

To train and test the classification models, synthetic samples have been created. We have analyzed three different synthesis methods (Universal, User-dependent, and GNN).

We then trained multiple bot detectors using the synthetic data generated with the proposed methods. We employed different classification algorithms including SVM, RF, GNB, and LSTM network, observing that each one has different behaviour and performance. Depending on the classification system, the generation part has a different performance but with enough training data the classification system is able to perfectly classify between humans and bots. We therefore conclude that keystroke dynamics can be used as a passive CAPTCHA.

Another important result of this work is the proposal of a novel Generative Neural Network. This network allows learning the distribution followed by the different classes within a data set. It is a pioneering network both for its architecture and for the way it learns from the data, with a loss function that evaluates the distribution.

The utilities of this network are many, whenever you want to learn a distribution or focus the learning of a network on distributions instead of individual values. The potential of this network lies in using the network as a unit and creating a network formed by these units, in this way one could learn complex functions (even non-linear) and have a non-deterministic network in classification.

The main line of future work is a system that presents both intra-user dependencies and key dependencies. To this end, different generative systems can be trained for different subjects (Generative user-dependent) or a certain correlation between the different keystroke time features can be included in the learning of the generative network itself.

Acknowledgment

This work has been supported by project BBforTAI (PID2021-127641OB-I00 MICINN/FEDER). The work of D. deAlcala is supported by a FPU Fellowship (FPU21/05785) from the Spanish MIU.

References

[1] A. Acien, A. Morales, J. Fierrez, and R. Vera-Rodriguez. BeCAPTCHA-Mouse: Synthetic Mouse Trajectories and Improved Bot Detection. *Pattern Recognition*, 127:108643, 2022. 1

[2] A. Acien, A. Morales, J. Fierrez, R. Vera-Rodriguez, and O. Delgado-Mohatar. BeCAPTCHA: Behavioral Bot Detection using Touchscreen and Mobile Sensors benchmarked on HuMIdb. *Engineering Applications of Artificial Intelligence*, 98:104058, 2021. 1

[3] A. Acien, A. Morales, John V. Monaco, R. Vera-Rodríguez, and Julian Fierrez. TypeNet: Deep learning keystroke biometrics. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 4(1):57–70, 2022. 2, 3

[4] Yousof Al-Hammadi and Uwe Aickelin. Detecting bots based on keylogging activities. In *Third International Conference on Availability, Reliability and Security*, pages 896–902, 2008. 2

[5] Emtethal K Alamri, Abdullah M Alnajim, and Suliman A Alsuhibany. Investigation of using captcha keystroke dynamics to enhance the prevention of phishing attacks. *Future Internet*, 14(3):82, 2022. 2, 8

[6] Md Liakat Ali, Kutub Thakur, Charles C Tappert, and Meikang Qiu. Keystroke biometric user verification using hidden markov model. In *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, pages 204–209, 2016. 2

[7] Blaine Ayotte, Mahesh Banavar, Daqing Hou, and Stephanie Schuckers. Fast free-text authentication via instance-based keystroke dynamics. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 2(4):377–387, 2020. 2

[8] Francesco Bergadano, Daniele Gunetti, and Claudia Picardi. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security*, 5(4):367–397, 2002. 2

[9] Vallance Chris. ChatGPT: New AI chatbot has everyone talking to it. *https://www.bbc.com/news/technology-63861322*, BBC, 7 December 2022. 1

[10] Vivek Dhakal, Anna Maria Feit, Per Ola Kristensson, and Antti Oulasvirta. Observations on typing from 136 million keystrokes. In *Proceedings of CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018. 3, 5

[11] Julian Fierrez, Aythami Morales, Ruben Vera-Rodriguez, and David Camacho. Multiple classifiers in biometrics. part 2: Trends and challenges. *Information Fusion*, 44:103–112, November 2018. 4

[12] J. Fierrez-Aguilar, D. Garcia-Romero, J. Ortega-Garcia, and J. Gonzalez-Rodriguez. Bayesian adaptation for user-dependent multimodal biometric authentication. *Pattern Recognition*, 38(8):1317–1319, August 2005. 4

[13] J. Fierrez-Aguilar, J. Ortega-Garcia, and J. Gonzalez-Rodriguez. Target dependent score normalization techniques and their application to signature verification. *IEEE Trans. on Systems, Man Cybernetics - Part C*, 35(3):418–425, August 2005. 4

[14] R. Gaines, S. Press, W. Lisowski, and N. Shapiro. Authentication by keystroke timing : some preliminary results. *RAND Corporation*, 1980. 2

[15] H Gamboa, ALN Fred, and AK Jain. Webbiometrics: User verification via web interaction. In *Biometrics Symposium*, pages 1–6, 2007. 1

[16] Radauskas Gintaras. AI-enabled cyberattacks might become norm in next five years. *https://cybernews.com/news/ai-enabled-cyberattacks-new-norm/*, Cybernews, 15 December 2022. 1

[17] Nahuel González, Enrique P Calot, Jorge S Ierache, and Waldo Hasperué. Towards liveness detection in keystroke dynamics: Revealing synthetic forgeries. *Systems and Soft Computing*, 4:200037, 2022. 2

[18] Aditya Grover, Manik Dhar, and Stefano Ermon. Flow-GAN: Combining maximum likelihood and adversarial learning in generative models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 2, 4

[19] Anil Jain, Karthik Nandakumar, and Arun Ross. 50 years of biometric research: Accomplishments, challenges, and opportunities. *Pattern Recognition Letters*, 79:80–105, 2016. 1

[20] JooSeuk Kim and Clayton D Scott. Robust kernel density estimation. *The Journal of Machine Learning Research*, 13(1):2529–2565, 2012. 3

[21] Qiao Liu, Jiaze Xu, Rui Jiang, and Wing Hung Wong. Density estimation using deep generative neural networks. *Proceedings of the National Academy of Sciences*, 118(15):e2101344118, 2021. 2, 4

[22] Sebastien Marcel, Julian Fierrez, and Nicholas Evans. *Handbook of Biometric Anti-Spoofing: Presentation Attack Detection and Vulnerability Assessment*. Springer, 2023. 1

[23] Abir Mhenni, Denis Migdal, Estelle Cherrier, Christophe Rosenberger, and Najoua Essoukri Ben Amara. Vulnerability of adaptive strategies of keystroke dynamics based authentication against different attack types. In *International Conference on Cyberworlds (CW)*, pages 274–278, 2019. 2

[24] Denis Migdal and Christophe Rosenberger. Analysis of keystroke dynamics for the generation of synthetic datasets. In *International Conference on Cyberworlds (CW)*, pages 339–344, 2018. 2

[25] Denis Migdal and Christophe Rosenberger. Statistical modeling of keystroke dynamics samples for the generation of synthetic datasets. *Future Generation Computer Systems*, 100:907–920, 2019. 2

[26] John V Monaco. Robust keystroke biometric anomaly detection. *arXiv preprint arXiv:1606.09075*, 2016. 2

[27] John V Monaco, Md Liakat Ali, and Charles C Tappert. Spoofing key-press latencies with a generative keystroke dynamics model. In *IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–8, 2015. 2

[28] Aythami Morales, Julian Fierrez, Alejandro Acien, Ruben Tolosana, and Ignacio Serna. SetMargin loss applied to deep keystroke biometrics with circle packing interpretation. *Pattern Recognition*, 122:108283, 2022. 2, 4

[29] A. Morales, J. Fierrez, and J. Ortega-Garcia. Towards predicting good users for biometric recognition based on keystroke dynamics. In *Proc. of European Conference on Computer Vision Workshops*, volume 8926 of *LNCS*, pages 711–724. Springer, September 2014. 4

[30] A. Morales, J. Fierrez, R. Tolosana, J. Ortega-Garcia, J. Galbally, M. Gomez-Barrero, A. Anjos, and S. Marcel. Keystroke biometrics ongoing competition. *IEEE Access*, page 7736–7746, 2016. 2

[31] Christopher Murphy, Jiaju Huang, Daqing Hou, and Stephanie Schuckers. Shared dataset on natural human-computer interaction to support continuous authentication research. In *IEEE International Joint Conference on Biometrics (IJCB)*, pages 525–530, 2017. 2

[32] Christian Rathgeb, Ruben Tolosana, Ruben Vera-Rodriguez, and Christoph Busch. *Handbook of Digital Face Manipulation and Detection: From DeepFakes to Morphing Attacks*. Springer, 2022. 1

[33] Manmeet Singh, Maninder Singh, and Sanmeet Kaur. Issues and challenges in DNS based botnet detection: A survey. *Computers & Security*, 86:28–52, 2019. 1, 5, 6

[34] D. Stefan, S. Xun, and D. Yao. Robustness of keystroke-dynamics based biometrics against synthetic forgeries. *Computers & Security*, pages 109–121, 2012. 2, 8

[35] Giuseppe Stragapede, Paula Delgado-Santos, Ruben Tolosana, Ruben Vera-Rodriguez, Richard Guest, and Aythami Morales. Mobile keystroke biometrics using transformers. In *Proc. IEEE Intl. Conf. on Automatic Face and Gesture Recognition (FG)*, 2023. 2

[36] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016. 2, 4

[37] Chengyi Wang, Sanyuan Chen, Yu Wu, Ziqiang Zhang, Long Zhou, Shujie Liu, Zhuo Chen, Yanqing Liu, Huaming Wang, Jinyu Li, et al. Neural codec language models are zero-shot text to speech synthesizers. *arXiv preprint arXiv:2301.02111*, 2023. 1

[38] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2223–2232, 2017. 1