R^{2}BN: An Adaptive Model for Keystroke-Dynamics-Based Educational Level Classification

Ioannis Tsimperidis, Paul D. Yoo, *Senior Member, IEEE*, Kamal Taha, *Senior Member, IEEE*, Alexios Mylonas, *Member, IEEE*, and Vasilis Katos

***Abstract*—Over the past decade, keystroke-based pattern recognition techniques, as a forensic tool for behavioral biometrics, have gained increasing attention. Although a number of machine learning-based approaches have been proposed, they are limited in terms of their capability to recognize and profile a set of an individual's characteristics. In addition, up to today, their focus was primarily gender and age, which seem to be more appropriate for commercial applications (such as developing commercial software), leaving out from research other characteristics, such as the educational level. Educational level is an acquired user characteristic, which can improve targeted advertising, as well as provide valuable information in a digital forensic investigation, when it is known. In this context, this paper proposes a novel machine learning model, the randomized radial basis function network, which recognizes and profiles the educational level of an individual who stands behind the keyboard. The performance of the proposed model is evaluated by using the empirical data obtained by recording volunteers' keystrokes during their daily usage of a computer. Its performance is also compared with other well-referenced machine learning models using our keystroke dynamic datasets. Although the proposed model achieves high accuracy in educational level prediction of an unknown user, it suffers from high computational cost. For this reason, we examine ways to reduce the time that is needed to build our model, including the use of a novel data condensation method, and discuss the tradeoff between an accurate and a fast prediction. To the best of our knowledge, this is the first model in the literature that predicts the educational level of an individual based on the keystroke dynamics information only.**

***Index Terms*—Data analytics, forensic analysis, keystroke dynamics, machine learning, user profiling.**

Manuscript received December 19, 2017; accepted August 27, 2018. This paper was recommended by Associate Editor Q. Ji. *(Corresponding author: Paul D. Yoo.)*

I. Tsimperidis is with the Department of Electrical and Computer Engineering, Democritus University of Thrace, 67100 Xanthi, Greece (e-mail: itsimper@ee.duth.gr).

P. D. Yoo is with the Cranfield School of Defence and Security, Defence Academy of the United Kingdom, Swindon SN6 8LA, U.K., and also with the CSIS Department, Birkbeck College, University of London, London WC1E 7HX, U.K. (e-mail: paul.d.yoo@ieee.org).

K. Taha is with the ECE Department, Khalifa University, Abu Dhabi, UAE (e-mail: kamal.taha@kustar.ac.ae).

A. Mylonas and V. Katos are with the Department of Computing and Informatics, Bournemouth University, Poole BH12 5BB, U.K. (e-mail: amylonas@bmth.ac.uk; vkatos@bmth.ac.uk).

Digital Object Identifier 10.1109/TCYB.2018.2869658

I. Introduction

GOOGLE has recently announced its ambitious plan to eliminate passwords in favor of systems that take into account a combination of sensor data such as typing patterns, gait patterns, coarse or exact location, etc. Keystroke dynamics, the patterns of rhythm and timing created when an individual types, has been used as a tool by many studies for the purpose of user authentication and user characterization. The lion's share of the research belongs to user authentication, since there are many studies that attempt to replace the traditional authentication with passwords, which suffers from many security and usability limitations.

Keyboard dynamics refers to the process of identifying the unique patterns of an individual's behavior with a computer-based keyboard device. It is closely related to the study of behavioral biometrics in digital forensics [1], [2]. Examples include gait, speech patterns, signatures, and keystrokes. Keystroke dynamics that measure an individual's unique typing rhythms have been the subject of considerable research over the past decade and their use as a tool for authentication has shown promising results [3]. From a digital forensics perspective, the ability to identify the user and link him or her to a set of activities performed within an information system is of paramount importance. This is seen as an attribution problem, and this practically relates to finding and correlating circumstantial evidence ranging from digital artifacts found on a suspect's hard disk and to analyzing the user's behavior from observable metrics.

User behavior-based biometrics technologies provide a number of advantages over traditional or physical biometric technologies. The information can be collected nonobtrusively or even without interfering with the users' ongoing work or consent. Collection of such behavioral data often does not require any additional hardware and thus is cost effective as well [4]. However, the efforts in the literature in profiling the user's characteristics using behavioral biometrics techniques have been limited to gender or age only. Some highlights are as follows. Yan and Yan [5] proposed a methodology that categorizes the authors of weblogs according to their gender. They used 75 000 blog entries and exploited the features of word appearance frequency, the blog's background color, font type and style, punctuations, and emoticons. Their methodology achieved the F-measure of 0.68. Mukherjee and Liu [6]

developed a machine-learning-based system that classifies the gender of blog's author. They collected data from 3100 blogs and classified the gender of authors using the features of styling words (like "hmm" and "lol"), gender preferential words (like "sorry," words ending with "−able," "−ful," and "−ous") and sequence of consecutive part-of-speech tags that satisfy some constraints. The system utilized naïve Bayes (NB) and support vector machine (SVM) classifiers and it achieved the accuracy of 88%. Cheng *et al.* [7] also proposed a machine-learning-based system that identifies the gender of author of a text. This paper was motivated by the rapid increase in crime on the Internet. Their dataset includes a collection of texts from Reuters' newsgroups and a collection of emails of Enron employees. They studied hundreds of text features and learned models using Bayesian-based logistic regression, AdaBoost decision tree, and SVM. The SVM showed the best results achieving 85% accuracy. Jones *et al.* [8] collected the data of user profiles and search keywords from Yahoo.com and learned a model using SVM-based classifier. Their SVM-based model achieved 83.8% accuracy on the gender classification and predicted the age of users with 63.9% accuracy. The study of Rangel *et al.* [9] had a consolidated list of 21 candidate models and classified the authors of English or Spanish texts based on their gender and age. Using decision trees, SVM, NB, and logistic regression models, they successfully classified the users into three age classes and the best accuracies were 59% for English speaking users and 65% for Spanish speaking users.

The aforementioned approaches, however, rely on computational machine learning models and have limitations. The most important of them is that all or some of the features used to classify the users are limited to certain phrases, words, N-grams, and the characters of a language. Most of the approaches are incapable of dealing with the heterogeneity of today's Internet as they were tested on English language only. It is reported that 25.9% of the Internet users are native English speakers and the half of websites worldwide are in English only [10]. Clearly, the keystroke dynamics information used in this paper could be seen as a remedy for such problem. Keystroke dynamics is defined as the detailed and precise timing information that describes when each key was pressed and when it was released as a user types on a keyboard and is first introduced in 1970s. Since then, many keystroke dynamics-based methods have been proposed to replace the password-based authentication.

The features used for analyzing the keystroke dynamics can be categorized into temporal and nontemporal. As temporal features are usually time-based, they are measured in milliseconds. The most well-accepted temporal features are keystroke-duration-based such as dwell time (the time a key pressed) and flight time (the time between "key up" and the next "key down") [11]. Other temporal features include the time associated with the trigrams and tetragrams [12]. Nontemporal features are nontime-based such as typing speed (e.g., words per minute), the frequency of errors, error correction mode, which key is used when there are two or more options ("Shift," "Ctrl," "Alt," "Enter," etc.) [13]. Other nontemporal features may include the time of day, applications used, and the frequency using the keyboard.

In this context, this paper introduces a novel learning architectured model, randomized radial basis function network (R^2BN). R^2BN can recognize and profile the educational level of an individual who stands behind the keyboard. The performance of the proposed model achieves a test error comparable to or better than the state-of-the-art machine learning models. The contribution of this paper can be summarized as follows.

1) R^2BN, a novel machine learning model predicting the educational level of users from keystroke dynamics only. We compare our model with other well-known machine learning models that have been used in the domain. Our experimental results suggest that our model is much more suitable model with regards to accuracy of predicting the educational level of the users. Moreover, we discuss the ways to address the limitation of the proposed model, namely the time needed to build the model (TBM). In this regard, we discuss the tradeoff of accuracy versus TBM when we: 1) use different iterations to build the model and 2) use different sampling rates (100% to 10% sampling) in a proposed data condensation method.
2) We create a dataset that can be used to study keystroke dynamics that have been captured from free text over long periods of time. To the best of our knowledge, a similar dataset, i.e., one containing keystrokes recorded from real users during the daily usage of their computer for a long period of time instead of typing 2 to 3 sentences, is not available in the literature.

Having the ability to identify the educational level of an individual who types a certain piece of text is of significant importance in digital forensics. This holds true, as it could be the source of circumstantial evidence for "putting fingers on keyboard" and for arbitrating cases where the true origin of a message needs to be identified. Moreover, if the proposed method is included as part of a text composing system, such as emails and instant texting, it could increase trust toward the applications that use it and may also work as a deterrent for crimes involving forgery. Also, accurately extracting the typing patterns of users who attempt to authenticate in a computing system that does not use the password scheme, like Google Abacus, effectively reduces types II errors. Finally, knowing the educational level of a user, could improve targeted advertising as the focus on his/her interests could be predicted easier. In this regard, we make the features used from our dataset available to the research community, hoping that we will inspire and aid more research in this domain.

The rest of this paper is organized as follows. Section II describes the data acquisition, the keystroke dynamics feature extraction, and the design and construction of novel data condensation method and R^2BN model. Section III summarizes the results obtained by comparing the performance of the proposed model and other nine well-known machine learning models. Section IV provides the related work and Section V concludes this paper.

II. Method

Our aim is to recognize and profile the educational level of an individual who stands behind a keyboard. To this end, our approach consists of two successive phases. First, we collect free text data from the volunteers who agreed to participate in this venture of capturing their daily, real-life keystrokes over a period of ten months. Then, we construct a novel machine learning model, R^{2}BN, predicting the educational level of users from keystroke dynamics only. Finally, we propose a data condensation method in order to reduce the training time of R^{2}BN.

A. Keystroke Dynamics Dataset

For the data collection, two main issues have been considered. We first considered Internet heterogeneity and standardization. The heterogeneity of the Internet comes along the issues in languages, locations, and structural and operational varieties. The users may have received education from different education systems around the world. They are particularly different in terms of length of time and grade. We addressed this by adopting the International Standard Classification of Education (ISCED) [16], which aims to attain a standardization and classification of different education systems around the world.

The second is related to the keystroke dynamics data itself. Data acquisition for the purpose of analysis of keystroke dynamics requires deploying a keylogger on a volunteer's computing device. The volunteer may either be requested to type a specific and fixed text, or a free text. The latter is preferred as it integrates better with the subject's regular typing activities and is less intrusive. However, the continuous recording of a volunteer's typing over an extended duration of time introduces the risks of disclosing passwords and personal messages to a third party. This is the main reason for the lack of existence of such recorded free text data in the literature.

We designed and developed a free text keylogger, called "IRecU" for the purpose of recording the user's free text. This can be installed into any Microsoft Windows-based device. The IRecU is available at [17]. The volunteers were asked to provide their educational information and the available levels of ISCED-2011 to be selected were ISCED-2, ISCED-3, ISCED-4, ISCED-5, ISCED-6, and ISCED-7-8. To mitigate the effect of the aforementioned risks, the volunteers were given an option to clear out what they have typed.

The IRecU creates a comma-delimited text (.txt) with the following data for each volunteer:

$$
\begin{aligned}
&75, \#2014-05-02\#, 47353342, \text{“dn”}\\
&65, \#2014-05-02\#, 47353436, \text{“dn”}\\
&75, \#2014-05-02\#, 47353441, \text{“up”}\\
&73, \#2014-05-02\#, 47353529, \text{“dn”}\\
&65, \#2014-05-02\#, 47353545, \text{“up”}\\
&73, \#2014-05-02\#, 47353639, \text{“up”}\\
&32, \#2014-05-02\#, 47353779, \text{“dn”}\\
&32, \#2014-05-02\#, 47353904, \text{“up”}.
\end{aligned}
$$

TABLE I
Educational Level Log Files

Levels	No. of Files	Percentage
ISCED-3	40	16.5
ISCED-4	15	6.2
ISCED-5	43	17.8
ISCED-6	85	35.1
ISCED-7-8	59	24.4
Total	242	100.0

The log files varied in size from 170 KB to 271 KB and contained data from 2,800 to 4,500 keystrokes. The class ISCED-2 is absent because there was no log file from a volunteer with this educational level.

Each line represents a record of the volunteer's action. The first field represents the virtual key code of the key that the volunteer pressed or released. The second field indicates the date the action took place in the format of yyyy-mm-dd. The third field is the elapsed time since the beginning of that day (12:00 A.M.) in milliseconds, and the fourth field is the action, "dn" for key-press and "up" for key-release. We then extracted all the features of keystroke dynamics from the text files. For example, the duration of keystroke is calculated from the subtraction of *ms* that correspond to the "up" action minus the *ms* that correspond to the "dn" action, for the same key. Similarly, all the digram latency representations are calculated. Examples are press–press, release–press, press–release, and release–release digram latency. The number of log files per educational level is shown in Table I.

There are many useful features that can be extracted from keystroke dynamics. Some of them are temporal, which are usually time-based and measured in millisecond. The most used features in the literature are keystroke duration, which is the time that a key is kept pressed, and the digram latency, which is the time between the pressing or releasing of a key and the pressing or releasing of the next key (this creates four combinations, down-down, down-up, up-down, and up-up). In addition, there are other temporal features that have been used or may be used, such as: 1) those which include the time associated with the trigrams, tetragrams, etc., [18]; 2) the number, the duration, and the frequency of pauses during typing [19]; and 3) the typing rate [20].

Except from temporal features, there are nontemporal features that are nontime-based, such as: typing speed (e.g., words per minute), the frequency of errors, error correction mode, finger pressure on the keys [21], which key is used when there are two or more options (e.g., "Shift" [22], "Ctrl," "Alt," "Enter," etc.) [23]. Other nontemporal features may include the time of day, applications used, and the frequency of using the keyboard.

In this paper, we use the most popular features of keystroke dynamics, i.e., keystroke durations and digram latencies. The reason for this is that for keystroke pressure features, a dedicated pressure-sensitive keyboard is essential, which contradicts with the main advantage of keystroke dynamics biometrics. Moreover, the frequency of word errors, typing rate, and duplicate keys features are merely practical for text with large number of characters. Finally, trigrams, tetragrams,

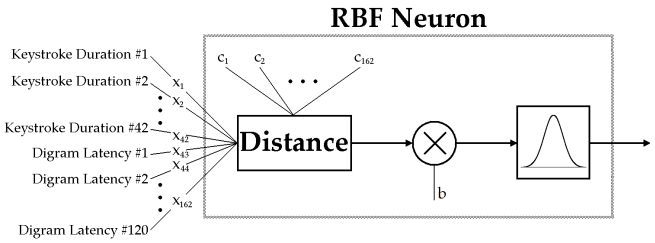

Fig. 1. Structure of the RBF neuron.

etc, are much rarer than monograms and digrams [24]. More specifically, we preferred down–down digram latencies to avoid negative values, because the second key may be pressed or may even be released before the releasing of the first key.

The total number of features to consider could have been almost 10 000 if we had used all the keystroke durations and digram latencies assuming an average computer's keyboard has 100 keys. However, some keys and digrams are used rarely or never. Based on our observation, we decided to include 42 keys and 120 digrams only. To extract features from the log files, we developed "ISqueezeU" software, which reads the text files created by "IRecU" and calculates the average values of durations or latencies. The keys that have at least ten appearances and the digrams with at least five appearances have been taken into account only. The values for the rest keys and digrams were marked as unknown.

The features from our dataset that were used in this paper are available on [25].

B. Randomized Radial Basis Function Network

In this section, we design and modify the radial basis function neural network (RBFN), which was initially proposed by Broomhead and Lowe [26]. RBFN shows faster convergence, smaller extrapolation errors, and higher reliability. It is built on a typical feedforward and three-layered architecture with a single hidden layer, where the activation functions for hidden units are defined as radially symmetric basis functions *phi*, such as the Gaussian function. Training an RBFN involves two phases: clustering like unsupervised on the hidden layer to determine N receptive field centroids in the training data set and the associated widths, and supervised on the output layer to estimate the connection weights w. The output layer is straightforward as it implements a simple multiple linear regression using the iterative gradient descent-based training method. In each RBF neuron, it stored a vector with as many dimensions as the number of the input layer neurons. This vector is called "center vector" and is denoted as c. Similarly, the input forms a vector of equal dimensions and the Euclidean distance between the input and the center vector is calculated. The calculated distance is then multiplied by a coefficient b and finally the product is applied to a radial basis function. This procedure is illustrated in Fig. 1.

The wavy line marks the boundaries of the RBF neuron, the $x_1, x_2, \ldots, x_{162}$ denote the components of the input vector and the $c_1, c_2, \ldots, c_{162}$ denote the components of the center vector. The output of the RBF neuron is given by the outcome of a radial function on the Euclidean distance, i.e.,

$$y^{(i)}(x) = r(\|x - c\|). \tag{1}$$

The index in (1) indicates that it is the output of the ith neuron of the network and the double bar denotes the Euclidean distance between vectors. Besides Euclidean distance, Mahalanobis distance could also be useful as it may give better results in some situations [27]. The radial function produces its largest response when the input vector is equal to the center vector. On the contrary, as the input moves near to the center, the response falls off exponentially as in

$$r(\|x - c\|) = e^{-b\cdot\|x-c\|^2}. \tag{2}$$

In the third layer, the number of neurons is as many as the number of the categories in which the data will be classified. This means that in our case there are five neurons in the output layer. Each output node computes a sort of score for the associated category. Typically, a classification decision is made by assigning the input to the category with the highest score. The score in every output neuron is computed by taking a weighted sum of the activation values from every RBF neuron, as shown

$$y(x) = \sum_{i=1}^{N} a^{(i)} \cdot e^{-b^{(i)}\cdot\|x-c^{(i)}\|^2} \tag{3}$$

where N is the number of neurons in hidden layer, $a^{(i)}$ is the weight assigned to the ith neuron, $b^{(i)}$ is the coefficient of the ith neuron, and $c^{(i)}$ is center vector of the ith neuron. During the training process, it selects three sets of parameters: the center vectors ($c^{(i)}$) and beta coefficient ($b^{(i)}$) for each of the neurons, and the matrix of output weights between the neurons and the output nodes ($a^{(i)}$). Schwenker *et al.* [28] provided an overview of common approaches to train such radial basis-based models. In short, they perform the k-means clustering on their training set and the cluster centers are used as the center vectors. The tradeoff in choosing the number of neurons in hidden layer is that the more neurons the higher the classifier's accuracy, while the less neurons the shorter the system's training time. The average distance between all instances in a cluster and the corresponding cluster center is given by

$$s = \frac{1}{m} \cdot \sum_{j=1}^{m} \|x_j - C\| \tag{4}$$

where m is the number of instances belonging to this cluster, x_j is the jth instance in the cluster, and C is the cluster center. Having the value s, the beta coefficient for the cluster is calculated as

$$b = \frac{1}{2 \cdot s^2}. \tag{5}$$

The output weights can be trained using the gradient descent optimization technique. The training inputs are the values obtained by the RBF neurons using the training set x. Gradient descent runs separately for each output node (that is, for each class in the data set).

Finally, the modified radial basis neural network is randomized [29]. A model that does not give satisfactory results we call it "weak." A strategy to enhance its performance is to combine many weak learners (WL) in a way that the output of each classifier can be aggregated to form a final decision. To provide a training set to the next-level classifier, weights are assigned to the instances, which determine the probability that should appear in the next training set. This probability is equal for every instance at the beginning of the procedure and therefore the first-level classifier uses the entire available training set. Instances with higher weights are more likely to be included in the next training set, and *vice versa.* The idea is to increase the weight on the misclassified instances so that these instances will make up a larger part of the next classifiers training set, and hopefully the next trained classifier will perform better on them. To explain how this works, let assume that there is a binary problem (only two classes) and every kth instance in the training set has the value p_k, while q_k is the correct output after the procedure. Because the problem is binary, the q_k may have the values $+1$ and -1. As mentioned earlier, weights are assigned to instances of the training set, which are stored to a vector W. Initially, these weights are equal for every instance and therefore all of them participate in the training procedure of the first WL. The weights are adjusted using the equation

$$W_{t+1}(k) = \frac{W_t(k) \cdot e^{-A_t \cdot q_k f_t(p_k)}}{S_t} \tag{6}$$

where $W_t(k)$ is the weight of the kth instance of the tth classifier training set, A_t is a coefficient corresponding to the tth classifier and indicates its importance to the final result, $f_t(p_k)$ is the prediction of the tth classifier for the kth instance, and S_t is a normalize factor which ensures that the sum of the instance weights is equal to 1. From above, it follows that the $q_k \cdot f_t(p_k)$ product will be positive when the prediction is correct and negative when it is incorrect. Because this product is part of a negative exponent entails that when the prediction is correct the weight of the instance will be decreased and when it is incorrect will be increased. Moreover, the W_t is a distribution because each weight $W_t(k)$ represents the probability that the kth instance will be selected as part of the next training set, and because every weight has value between 0 and 1 and their sum is 1. Once the training of all WL is completed, the output of the final classifier is given by

$$F(p) = \text{sign}\left(\sum_{t=1}^{T} A_t \cdot f_t(p)\right) \tag{7}$$

where T is the number of WL, $f_t(p)$ is the output of the tth WL, which is $+1$ or -1 in the case being studied and A_t is the coefficient that was assigned to the tth WL. Therefore, the final output is just a linear combination of all of the WLs and the final decision is simply the sign of this sum. It should also be noted that the A_t coefficients, each of which is computed after the training of the corresponding classifier as

$$A_t = \frac{1}{2} \cdot \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \tag{8}$$

Algorithm 1 Pseudo Code of R^{2}EN

procedure EDULEVELCLASSIF(Training Set x, Weak Learner RBFN$(\cdot,\cdot)$, Iterations T)
$W_1 = (1/n, \ldots, 1/n)$
for $t = 1$ to T
  $f_t = \text{RBFN}(x, W_t)$
  $\varepsilon_t = \sum_{i=1}^{n} W_t(i) \quad (f_t(x_i) \neq y_i)$
  $A_t = 1/2 \cdot \ln((1-\varepsilon_t)/\varepsilon_t)$
  for $i = 1$ to n
    $W_{(t+1)}(i) = (W_t(i) \cdot \exp(-A_t \cdot q_i\, f_t(p_i)))/S_t$
  end for
end for
return $F(p) = \text{sign}(\sum_{t=1}^{T} A_t \cdot f_t(p))$
end **procedure**

where ε_t is the number of misclassifications over the training set divided by the training set size. According to (8), when the ε_t quantity approaches 0 the A_t coefficient grows exponentially, which means that better classifiers play more important role to the final result. When the ε_t quantity is equal to 0.5, i.e., the classifier accuracy is 50%, the A_t is 0 and therefore every classifier with accuracy no better than random guessing is ignored. Finally, when the ε_t is over 0.5, the A_t is negative, which means that the opposite decision of the classifier is taken into account.

Algorithm 1 summarizes the above-mentioned operation of R^{2}BN procedure.

The parameters of Algorithm 2 are the training set x, the WL, which is the RBFN, and the number of iterations T, while n is the training sample size. In each iteration the weighted error ε_t, the coefficient A_t, and the weights W_t are calculated.

C. Data Condensation Method

One of the problems in any classification procedure is that, irrespective of the sophistication of the classifier in use, as the dataset size increases so does the computational time. As with any other classifier, the training time of R^{2}BN is considerable and highly depends on the number of iterations. One possible solution is to build the classification model on a much smaller representative subset of the original dataset. The aim is to reduce the dataset size as much as possible with the minimum loss of classifier performance.

To this end, this section proposes a new data condensation method, which precedes R^{2}BN as shown in Fig. 2.

The proposed filter-based random subfield data condensation method consists of three stages. In the first stage, assuming that B is a given training set, with N instances and M features, then the dataset D is a concatenation of B and the target values t. In simple random sampling (SRS), a sample R is selected from B uniformly with replacement following a binomial distribution. Equivalent weights are given to all samples in the dataset, so that any sample is chosen with equal probability regardless of whether it was previously sampled or not [30]. Although SRS simplifies analysis results, it

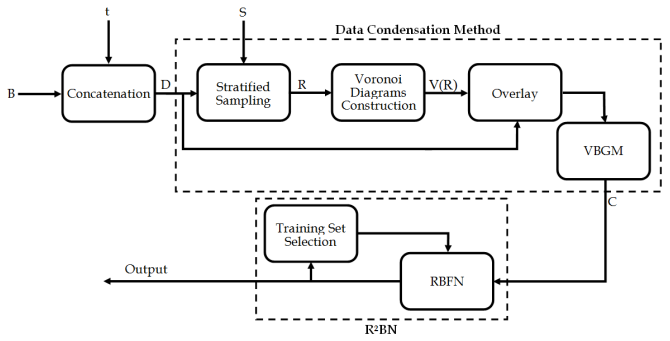

Fig. 2. R^{2}BN and data condensation method block diagram. D is stratified sampled by percentage S and R is produced. Voronoi diagrams $V(R)$ are produced based on R. Dataset D overlies over $V(R)$. VBGM clustering algorithm is used to produce dataset C. A number of RBFNs where each one uses different training set, calculated by the previous neural network, performs the classification.

suffers in performance when D is imbalanced, just as it happens in our case, where the class ISCED-4 occupies 6% of the instances and class ISCED-6 occupies 35%. In the filter-based random subfield data condensation, D is divided into a set of strata according to the different target values in t, and SRS is performed on each stratum independently. In this context, we use the proportional sampling approach, where the proportion sampled from each stratum is equal in the sample as it is in the original dataset. Thus, if D is highly imbalanced, then higher sampling percentage would be given to the small target stratum.

Upon sampling from D to produce R, we need to assess whether this sample is a good representative or not. There are two types of statistical tests, namely parametric and nonparametric statistical tests. Parametric tests make assumptions about the statistical distribution of the original dataset, while nonparametric tests assume no underlying statistical structure about the data. Thus, in order to maintain the generality in this framework, the nonparametric preference is adopted.

In this paper, the test was performed on the most significant features. The topic of feature subset selection has been well studied in the literature, whereby a feature selection algorithm consists of a search technique in conjunction with an evaluation metric. Here, a filter-based feature selection method, which utilizes the information gain as an evaluation metric with respect to the class value is used. It can be formalized as

$$\text{IG} = H(\text{class}) - H(\text{class}|\text{Attribute}) \tag{9}$$

where $H(x)$ is the entropy of x, given by

$$H(x) = -\sum_{i=1}^{E} \Pr(x[i]) 1\text{n } \Pr(x[i]) \tag{10}$$

where E is the length of the vector x and $\Pr(x[i])$ is the probability of the $x[i]$.

The second stage is the construction of a *Voronoi* diagram $V(R)$ based on the subsampled dataset R. The *Voronoi* diagram $V(R)$ is the collection of all *Voronoi* regions $VR(r_n, R)$ whose centers r_n are instances in R. For a brief definition of *Voronoi* diagram let $x, y \in \Re^M$. Then the bisector of x and y

$$B(x, y) = \left\{ \|x - s\|_p = \|y - s\|_p | s \in \Re^M \right\} \tag{11}$$

is the perpendicular line through the center of the line segment $\overline{xy}$. Where $\bar{A}$ denotes the closure of some set A and $\|.\|_p$ is the p-norm distance, defined as

$$\|w\|_p = \sqrt[p]{\sum_{i=1}^{M} w_i^p}. \tag{12}$$

The half plane $D(x, y)$ which is separated by the bisector is

$$D(x, y) = \left\{ \|x - s\|_p < \|y - s\|_p | s \in \Re^M \right\} \tag{13}$$

and the corresponding *Voronoi* region of x with respect to R is written as

$$VR(x, R) = \bigcap_{y \in R, y \neq x} D(x, y). \tag{14}$$

The main method for computing a Voronoi diagram is by the brute force approach. Thus, it could be defined as

$$VR(r_n, R) = \left\{ \|r_n x\|_p < \left\|r_j x\right\|_p, \forall j \neq n | x \in \Re^M \right\}. \tag{15}$$

Variety of efficient algorithms exists to compute the *Voronoi* diagram, such as the incremental construction, divide & conquer, and plane sweep. According to [31], the divide & conquer algorithm and the plane sweep construct a *Voronoi* diagram of n points within time $O(n \cdot \log n)$ and linear space, in the worst case, where both bounds are optimal.

Here, we used the Euclidean distance measure to calculate the *Voronoi* regions. Nonetheless, it is worth mentioning that variety of other distance metrics can be chosen depending on the nature of the problem and the data. For example, for continuously valued attributes, one can employ the *Mikowsky*, *Mahalanobis*, *Chebychev*, *Camberra*, *Quadratic*, *Correlation*, *Chi-square*, and many others. This stage is very crucial since it reduces the complexity of the next clustering stage, where we assume that data points that are far apart do not have an effect on each other. As you may have noticed, this approach allows the condensation process performed in parallel to the different *Voronoi* regions.

In third stage, the original dataset D overlies over the *Voronoi* diagram $V(R)$. The goal of this stage is to fetch representative centroids from each *Voronoi* region, for which the collection of all centroids comprise the reduced dataset C. It is remained to be decided which clustering algorithm must be used and how many centroids per *Voronoi* region should be fetched. By specifying as clustering validity measure that the clustering algorithm should attempt to minimize the intercluster distance measures and maximize the intracluster distance measures, the VBGM clustering algorithm [32] is used to fetch at most $\sqrt{U_n/2}$ centroids from each nth *Voronoi* region, with U_n to be the M-dimensional data instances that constitute the *Voronoi* region. In this paper, α_0 is set to 0.01, which most often results in selecting less than $\sqrt{U_n/2}$. Although, the sampling percentage is specified in the beginning of the algorithm, the final sampling percentage most often ends up being slightly less, where the highest worst case condensation percentage PER_{wc} is

$$\text{PER}_{\text{wc}} = N - K_n \cdot \sum_{n=1}^{k_n} \sqrt{\frac{U_n}{2}} \tag{16}$$

Algorithm 2 Pseudo Code of the Subsampling Framework

1: **procedure** REDUCEDATASET(Dataset D, Sampling Percentage S)
2: **Stage 1:**
3: $R = StratifieldSampling(D, S)$
4: **Stage 2:**
5: $V(R) = ConstructVoronoi(R)$
6: Overlay D onto V(R)
7: **Stage 3:**
8: while $n \leq S \times length(R)$ do
9: C = C+VBGM($VR(r_n, R)$, $\sqrt{U_n/2}$)
10: end while
11: end **procedure**

where K_n is the number of clusters. The three stages of the filter-based random subfield data condensation are described in Algorithm 2.

Algorithm 2 has two main parameters the Dataset D and a sampling percentage S which takes values between 0 and 1. In stage 1, a representative set R from D is selected by calling a stratified random sampling. The number of selected instances corresponds to the specified parameter S. In stage 2, based on R, a *Voronoi* diagram $V(R)$ is constructed by partitioning the dataset space into $L = S \times N$ *Voronoi* regions $\in V(R)$. Finally, in stage 3, the original instances from D is overlayed back onto $V(R)$ and VBGM clustering algorithm is used to fetch at most $\sqrt{U_n/2}$ centroids from each nth*Voronoi* region. Thus, now it can be assumed that for each local region in the input space represented by a center vector and there is a corresponding scalar output into which it maps then the centroids can be followed by convolution process as below.

In addition, for datasets of high dimensions, one might employ cheap distance metrics. For instance, only the most significant features can be chosen for the distance metric, which in turn reduces the time latency of computing the distances drastically. Moreover, one could rely on the construction of an approximate *Voronoi* diagrams as in the proposed architecture. This stage is very crucial since it reduces the complexity of the next clustering/learning stage. Another note that merits a mention is that this framework allows for parallel condensation to be performed to the different *Voronoi* regions.

III. MODEL EVALUATION AND ANALYSIS

In this section, we evaluate and compare the performances of the proposed R^{2}BN model and other well-known nine machine learning models: 1) radial basis function network (RBFN); 2) multilayer perceptron (MLP); 3) SVM; 4) random forest (RF); 5) logistic model tree; 6) NB tree (NBTree); 7) best first tree; 8) NB classifier; and 9) simple logistic. The models are tested on the benchmark keystroke dynamics dataset in terms of: 1) the model accuracy (Acc.), which is the percentage of correctly classified instances; 2) the stability (σ), which is the measure of deviation of all the model accuracies over tenfolds; and 3) time complexity (TBM), which is the CPU time required to build the model.

More specifically, to assess the performance of the proposed model fairly with regards to the above-mentioned criteria (i.e., accuracy, stability, and time complexity), the well-referenced cross-validation is used [33]. In the tenfold cross-validation, which is the most commonly used version, the dataset is divided into ten subsets. Each time, one of the 10 subsets is used as the testing set and the other 9 subsets are put together to form a training set. Then, the average error across all ten trials is computed. The advantage of this method is that the way the data gets divided matters less. Every data point gets to be in a testing set exactly once, and in a training set 9 times.

Moreover, to validate the quality of the proposed model, the F-score and ROC index are computed and provided. For the F-score, we calculated the harmonic mean of the specificity and sensitivity [34], where its value falls in the range between 0 and 1. Accuracy is measured by ROC index, the area under the ROC curve [35].

However, as the ISCED-4 class makes only 6.2% of the whole dataset and the learning models rarely see the samples and adjust the weights, increasing the probability of unreliable results (i.e., imbalance effect), ISCED-3 and ISCED-4 classes are merged forming the new ISCED-3-4 classes. Now the dataset has the following four classes: 1) those who do not have tertiary education; 2) those who have short cycle tertiary education; 3) those who have university degree; and 4) those who have education higher than tertiary. Again, ISCED-3-4 and ISCED-5 are merged to form a new ISCED-3-4-5 class dataset. This balances up the number of samples in each class for fair model evaluation and comparisons. The experimental results are summarized in Table II.

R^{2}BN excels all other models in terms of Acc., in 3-, 4-, and 5-classes datasets, about 8% improved from its base model (RBFN). More specifically, R^{2}BN achieves an accuracy of 86.8% with 5-classes dataset. Similarly, with 4-classes dataset, the difference between the accuracies of the R^{2}BN predictions and the baseline is over 62%, while in 3-classes case this difference is over 54%. In addition, R^{2}BN becomes the second in the stability behind the RBFN model, achieving 7.9±1.3 deviation of all the model accuracies over tenfolds. Moreover, R^{2}BN has the greatest ROC index, over all the other models, reaching a value only 0.043 less than the optimum 1.0 and followed by RF and MLP which have 0.904 and 0.886, respectively. However as our results suggest, almost all the other models require less training time (see Table II). Specifically, R^{2}BN's is only better than MLP and NBTree with regards to TBM. It requires considerably more time to learn a model than NB (approx. 3075×), SVM (approx. 88×), RBFN (approx. 41×), and RF (approx. 8×).

Fig. 3 visualizes the experimental results of accuracy for the ten models, over three different datasets. As suggested by our experimental results, R^{2}BN, seems to be a much more suitable model, achieving 88.43% Acc. We believe that the utility of R^{2}BN can be confirmed with these experimental results, however, there is a drawback that should be addressed. Specifically, it is obvious that learning an optimal model for R^{2}BN is still time consuming. We performed additional experiments on 3, 4, and 5 classes. As shown in Table III, the weights were converged over 70 iterations, which we believe

TABLE II
RESULTS OF MODEL COMPARISONS ON DIFFERENT CLASSES

Model	Class	Acc.	σ	ROC	TBM
R·BN	3	88.43	9.20	0.959	73.82
	4	87.19	7.73	0.949	49.09
	5	86.78	6.61	0.966	67.70
Avg.		87.6±0.8	7.9±1.3	0.957±0.008	61.5±12.4
RBFN	3	80.99	8.70	0.880	1.26
	4	77.27	5.19	0.860	1.67
	5	79.34	4.60	0.885	1.58
Avg.		79.1±1.9	6.7±2.1	0.873±0.013	1.5±0.2
MLP	3	75.62	7.58	0.898	77.27
	4	67.77	7.67	0.874	85.82
	5	69.83	14.08	0.876	80.73
Avg.		71.7±3.9	10.8±3.2	0.886±0.012	81.5±4.3
SVM	3	69.83	5.49	0.802	0.61
	4	63.22	12.23	0.808	0.49
	5	64.88	10.82	0.839	0.90
Avg.		66.5±3.3	8.8±3.4	0.820±0.018	0.7±0.2
RF	3	79.34	9.32	0.899	5.80
	4	71.49	9.36	0.896	9.50
	5	71.49	11.67	0.911	6.46
Avg.		75.4±3.9	10.5±1.2	0.904±0.008	7.7±1.9
LMT	3	71.07	10.53	0.862	4.43
	4	67.77	8.06	0.847	3.38
	5	62.40	10.56	0.817	6.68
Avg.		66.7±4.3	9.3±1.3	0.840±0.023	5.0±1.7
NBTree	3	66.12	12.73	0.771	132.19
	4	60.74	8.41	0.795	102.73
	5	61.16	10.26	0.783	107.58
Avg.		63.4±2.7	10.6±2.2	0.783±0.012	117.5±14.7
BFTree	3	71.90	7.95	0.802	13.53
	4	59.50	11.00	0.724	1.30
	5	58.68	8.56	0.750	2.50
Avg.		65.3±6.6	9.5±1.5	0.763±0.039	7.4±6.1
NB	3	57.25	10.12	0.699	0.01
	4	51.24	10.81	0.712	0.03
	5	55.37	8.10	0.722	0.02
Avg.		54.2±3.0	9.5±1.4	0.711±0.012	0.02±0.01
SL	3	69.83	8.01	0.834	2.04
	4	66.53	11.84	0.824	1.67
	5	55.79	8.10	0.795	5.63
Avg.		62.8±7.0	9.9±1.9	0.815±0.020	3.7±2.0

R·BN (30 clusters for K-Means, 0.9 minimum standard deviation for the clusters and 55 iterations for adaptive boosting, for 5-classes dataset, 30 clusters for K-Means, 1.9 minimum standard deviation for the clusters and 45 iterations for adaptive boosting, for 4-classes dataset, and 60 clusters for K-Means, 1.0 minimum standard deviation for the clusters and 70 iterations for adaptive boosting, for 3-classes dataset), RBFN (70 clusters for K-Means and 1.0 minimum standard deviation for the clusters, for 5-classes dataset, 100 clusters for K-Means and 1.0 minimum standard deviation for the clusters, for 4-classes dataset, and 40 clusters for K-Means and 1.2 minimum standard deviation for the clusters, for 3-classes dataset) MLP (0.7 learning rate and 0.4 momentum, for 5-classes dataset, 0.5 learning rate and 0.6 momentum, for 4-classes dataset, and 0.8 learning rate and 0.6 momentum, for 3-classes dataset), SVM (1.0 C-value and Polykernel as kernel type, for 5-classes dataset, 1.0 C-value and Polykernel as kernel type, for 4-classes dataset, and 3.0 C-value and Polykernel as kernel type, for 3-classes dataset), RF (100 trees with 100 random features each, for 5-classes dataset, 100 trees with 162 random features each, for 4-classes dataset, and 90 trees with 120 random features each, for 3-classes dataset), LMT (10 iterations for LogitBoost, 1 as minimum number of instances for splitting a node and 0.01 beta value for LogitBoost, for 5-classes dataset, 4 iterations for LogitBoost, 1 as minimum number of instances for splitting a node and 0.0 beta value for LogitBoost, for 4-classes dataset, and 10 iterations for LogitBoost, 1 as minimum number of instances for splitting a node, and 0.01 beta value for LogitBoost, for 3-classes dataset), BFTree (5 folds in internal cross validation and 1 as minimum number of instances at the terminal nodes, for 5-classes dataset, 3 folds in internal cross validation and 1 as minimum number of instances at the terminal nodes, for 4-classes dataset, and 30 folds in internal cross validation and 2 as minimum number of instances at the terminal nodes, for 3-classes dataset), and SL (no iterations for LogitBoost and 40 as heuristic stop, for 5-classes dataset, 130 iterations for LogitBoost and 50 as heuristic stop, for 4-classes dataset, and no iterations for LogitBoost and 30 as heuristic stop, for 3-classes dataset).

is causing the issue. Even though the standard RBFN has an advantage of faster convergence, the meta nature of the modified R^{2}BN increases the model complexity leading to slow overall convergence.

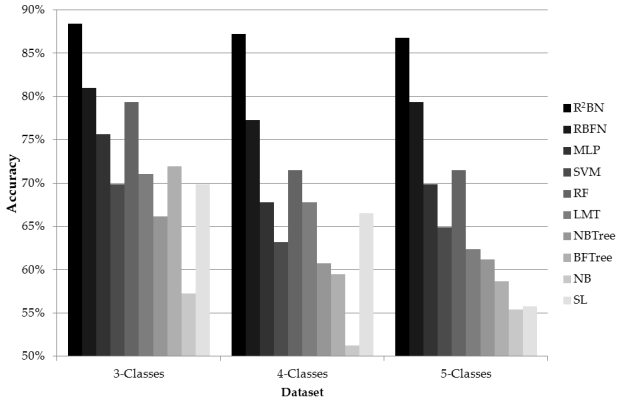

Fig. 3. Accuracy comparison over three different classes' datasets.

TABLE III
PERFORMANCE OF R^{2}BN MODEL ACCORDING TO NUMBER OF ITERATIONS

Iterations	3 Classes Acc. (%)	3 Classes TBM (sec)	4 Classes Acc. (%)	4 Classes TBM (sec)	5 Classes Acc. (%)	5 Classes TBM (sec)
5	84.71	6.09	80.99	6.67	79.75	7.75
10	86.36	12.07	83.06	13.55	84.30	13.71
15	86.78	19.14	84.30	17.53	84.30	19.56
20	87.19	24.56	85.12	24.27	85.12	25.73
25	87.19	29.89	85.95	30.69	85.54	30.54
30	87.19	34.62	85.95	34.62	85.12	34.86
35	87.19	41.17	86.36	40.54	85.12	41.55
40	87.19	44.19	86.78	44.94	85.12	47.55
45	87.19	54.24	**87.19**	49.09	85.54	56.63
50	87.60	58.22	86.78	54.32	85.95	58.81
55	87.60	66.08	86.36	61.28	**86.78**	67.70
60	88.02	69.79	86.36	63.82	86.36	73.88
65	88.02	70.54	86.36	70.55	**86.78**	79.64
70	**88.43**	73.82	86.36	75.99	85.95	86.08
75	**88.43**	83.76	**87.19**	82.16	86.36	91.85
80	**88.43**	90.13	**87.19**	86.58	86.36	97.53

The highest accuracy on each dataset is bolded and underlined.

As it was expected, the TBM is increasing linearly as the number of iterations increases, while the accuracy of the system seems to tend to reach a maximum value. This is clearer in Fig. 4, which visualizes the findings of Table III.

As seen in Table III, however, with the configurations of 15 iterations for 3 classes, 25 iterations for 4 classes, and 20 iterations for 5 classes, the time complexities could be reduced by 74%, 60% and 62%, respectively, which makes the R^{2}BN the most accurate model with time complexity of 24.9±5.8, meaning that the time required to build the model is reduced by 60% with less than 2% drop of accuracy.

An alternative way to reduce TBM is the use of a data condensation method. Table IV presents the Acc and TBM of R^{2}BN when the data condensation method that was described in Section II-C is used, with sampling percentage from 100% to 10%. As shown by our results, there are instances in which the accuracy of R^{2}BN does not significantly degrade, while TBM reduces remarkably. For example, in 3-classes dataset,

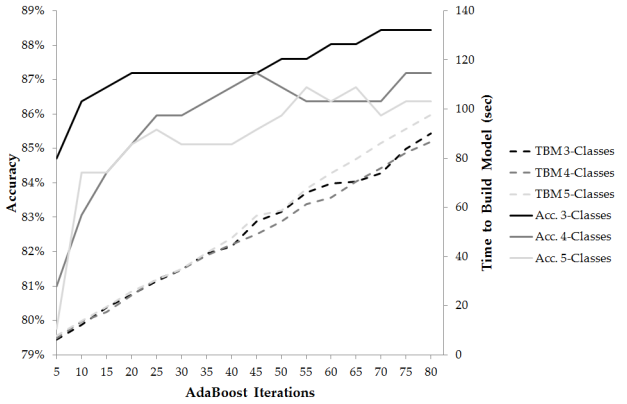

Fig. 4. Accuracy and time needed to build model of different classes' datasets according the number of iterations.

TABLE IV
PERFORMANCE OF R^{2}BN MODEL USING DATA CONDENSATION METHOD

	3 Classes		4 Classes		5 Classes	
R-Rate	Acc. (%)	TBM (sec)	Acc. (%)	TBM (sec)	Acc. (%)	TBM (sec)
0.0	88.43	73.82	87.19	49.09	86.78	67.70
0.1	81.82	54.68	80.82	40.25	79.09	54.05
0.2	81.68	36.39	78.50	40.76	72.28	50.95
0.3	71.82	29.73	70.62	26.18	73.30	40.12
0.4	77.71	26.28	76.40	23.99	70.66	45.55
0.5	71.14	24.12	66.89	18.64	66.44	32.85
0.6	68.50	19.44	61.72	13.87	57.94	26.68
0.7	63.30	16.21	61.68	11.79	57.28	18.00
0.8	69.51	7.60	56.79	6.97	57.83	15.18
0.9	66.04	5.57	55.36	4.59	59.62	4.76

Results from 0% to 90% condensation.

when the sampling percentage is 0.4, the model works 3 times faster with a loss of 10% in accuracy.

IV. RELATED WORK

To the best of our knowledge, we are one of the first to provide a dataset containing keystroke dynamics, which have been collected from free text, recorded from real users during the daily usage of their computer for a long period of time. It also includes demographic data to be used for user classification, such as educational level. The datasets in the literature contain keystroke data, which in most cases, are collected from users typing 2 to 3 sentences only [36]. Rybnik *et al.* [37] created a free text dataset, which contains keystrokes from nine volunteers. Each volunteer typed a long text of more than 250 characters twice in five sessions. Similarly, the dataset of Messerman *et al.* [38] included keystrokes that were recorded over a 12-month period by 55 volunteers using a Web-mail application.

Compared to our work, these datasets were created by users' typing in specific environments rather than from normal use. Some other datasets were created on specific devices rather than on the user's device, while the recording was done in a designated area and not in the familiar space (home, office, etc.) of the volunteers. Finally, few datasets were created by long time user recording than in some sessions, and few are those that contain data from thousands of keystrokes in each logfile, as is the case with our own readers interested in a survey of free text datasets that were created to test user authentication methods may refer to Alsultan and Warwick [39].

Regarding to data condensation, because it is a significant procedure in machine learning, especially in cases where the speed of a system is crucial, many methods have been proposed to reduce the training time needed. Liu *et al.* [40] used three different methods to speed up the computation of SVM classifier, namely one with random selection of data and two other using proximity graphs. Similarly, Gamboni *et al.* [41] aimed to increase the speed of SVMs on large datasets using three methods, i.e., blind random sampling and two linear-time methods for guided random sampling. A different approach was proposed by Rizwan and Anderson [42], where an adaptive data condensation scheme for k-NN classifiers is used by reducing the instances in the training data based on the observed similarities.

In most cases, the proposed data condensation methods significantly reduce the computational time with a slight decrease of the classification performance. The novelty of the proposed method of this paper lies in the possibility of applying to the highly unbalanced datasets, as with the problem we are dealing with.

V. CONCLUSION

Keystroke-based pattern recognition techniques, as a tool for behavioral biometrics, are of great importance in digital forensics. This paper introduced a novel model which recognizes and profiles the educational level of an individual who stands behind a keyboard.

To accomplish this objective, a new keystroke dynamics dataset was created, by recording free text captured from participants' daily usage of their computers over a period of ten months. We captured 242 log files from users who belong into five educational level classes. Each file contains 2800 to 4500 keystrokes. We extracted 162 features corresponding to the dwell times and flight times of the participants keystrokes.

Recruiting users and capturing their everyday keystroke is a hard task, especially if one considers that keystrokes often contain sensitive information. This increases the complexity of finding participants that can provide real keystroke dynamics and not synthesized ones. While our dataset is limited and affected by our participants' demographics, to the best of our knowledge, no other dataset with real keystroke dynamics (i.e., one captured from free text) exists. Instead, previous datasets required participants to write free text that was 2 to 3 sentences long only. By making the features used from our dataset available to the research community, we hope that we will inspire and aid more research in keystroke-based pattern recognition.

In this paper, the dataset was fed into a proposed model, named R^{2}BN, which is the randomized modification of a well-known radial basis neural network. R^{2}BN is built on

a shallow-learning architecture, which aims to achieve faster convergence and smaller extrapolation errors. It improved its performance in terms of model accuracy, stability, and learning complexity. The experimental results on the dynamic keystroke dataset proved that the proposed R^2BN can achieve a test error comparable to or better than the state-of-the-art models.

Our experimental results uncovered a limitation of R^2BN, namely the long time required to build the model. This can be addressed by either performing fewer iterations on the boosting algorithm, or by using a new data condensation method. Our evaluation uncovered cases where high accuracy is maintained while TBM is considerably reduced.

For a given set of keystroke data, the randomization of radial function-based model provides a way of fine-tuning the global model, as well as improving its predictive performance. Having the ability to identify the educational level of a user who types a certain piece of text has significant value in digital forensics. To the best of our knowledge, this paper introduces the first model that can achieve above 85% model accuracy, as well as the greatest model stability over three different classes in the keystroke-dynamic-based educational level classification task. The utility of the proposed model have been proven in dealing with the source of circumstantial evidence for "putting fingers on keyboard" and for profiling the characteristics of the users. However, we note that the deployment of such a system must be in accordance with the current, enforced legal and regulatory framework, as the unauthorized analysis of keystrokes entails privacy violations, which might involve sensitive personal information (e.g., in accordance to the EU legislation).

Our plans for future work include further classification tasks such as predicting handedness. We also plan further work on examining alternatives for the data condensation method as a means to reduce the time that is needed to build the R^2BN.

REFERENCES

[1] K. Taha and P. D. Yoo, "SIIMCO: A forensic investigation tool for identifying the influential members of a criminal organization," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 811–822, Apr. 2016.

[2] M. Alzaabi, K. Taha, and T. A. Martin, "CISRI: A crime investigation system using the relative importance of information spreaders in networks depicting criminals communications," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 10, pp. 2196–2211, Oct. 2015.

[3] F. Monrose and A. D. Rubin, "Keystroke dynamics as a biometric for authentication," *Future Gener. Comput. Syst.*, vol. 16, no. 4, pp. 351–359, 2000.

[4] R. V. Yampolskiy and V. Govindaraju, "Behavioural biometrics: A survey and classification," *Int. J. Biometrics*, vol. 1, no. 1, pp. 81–113, 2008.

[5] J. X. Yan and L. Yan, "Gender classification of weblog authors," in *Proc. Amer. Assoc. Artif. Intell. Spring Symp.*, 2006, pp. 228–230.

[6] A. Mukherjee and B. Liu, "Improving gender classification of blog authors," in *Proc. Conf. Empir. Methods Nat. Lang. Process.*, 2010, pp. 207–217.

[7] N. Cheng, R. Chandramouli, and K. P. Subbalakshmi, "Author gender identification from text," *Digit. Invest.*, vol. 8, no. 1, pp. 78–88, 2011.

[8] R. Jones, R. Kumar, B. Pang, and A. Tomkins, "'I know what you did last summer': Query logs and user privacy," in *Proc. 16th ACM Conf. Inf. Knowl. Manag.*, New York, NY, USA, 2007, pp. 909–914.

[9] F. Rangel, P. Rosso, M. Koppel, E. Stamatatos, and G. Inches, "Overview of the author profiling task at PAN 2013," in *Proc. Conf. Labs Eval. Forum*, Valencia, Spain, 2013, pp. 23–26.

[10] G. Rehm and H. Uszkoreit, *The English Language in the Digital Age*. Heidelberg, Germany: Springer, 2012.

[11] D. Song, P. Venable, and A. Perrig. (1997). *User Recognition by Keystroke Latency Pattern Analysis*. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.37.7619&rep=rep1&type=pdf

[12] Y. Zhao, "Learning user keystroke patterns for authentication," in *Proc. World Acad. Sci. Eng. Technol.*, vol. 14. Karnataka, India, 2006, pp. 65–70.

[13] K. Hempstalk, "You are what you type?" in *Proc. New Zealand Comput. Sci. Res. Student Conf.*, Christchurch, New Zealand, 2008, pp. 24–31.

[14] P. V. P. Sundar and A. V. S. Kumar, "An enhanced disengagement detection in online learning using quasi framework," *Int. J. Appl. Eng. Res.*, vol. 10, no. 55, pp. 1298–1302, 2015.

[15] I. Tsimperidis, S. Rostami, and V. Katos, "Age detection through keystroke dynamics from user authentication failures," *Int. J. Digit. Crime Forensics*, vol. 9, no. 1, pp. 1–16, 2017.

[16] *ISCED: International Standard Classification of Education*. Accessed: Sep. 21, 2018. [Online]. Available: http://www.uis.unesco.org/Education/Pages/international-standard-classification-of-education.aspx

[17] *IRecU—BU Cyber Security Research Group*. Accessed: Sep. 21, 2018. [Online]. Available: https://cybersecurity.bournemouth.ac.uk/?page_id=265

[18] R. Moskovitch *et al.*, "Identity theft, computers and behavioral biometrics," in *Proc. IEEE Int. Conf. Intell. Security Informat.*, Piscataway, NJ, USA, 2009, pp. 155–160.

[19] A. Goodkind and A. Rosenberg, "Muddying the multiword expression waters: How cognitive demand affects multiword expression production," in *Proc. Conf. North Amer. Ch. Assoc. Comput. Linguist. Human Lang. Technol.*, Denver, CO, USA, 2015, pp. 87–95.

[20] M. Villani *et al.*, "Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions," in *Proc. Conf. Comput. Vis. Pattern Recognit. Workshop*, New York, NY, USA, 2006, p. 39.

[21] H. Saevanee and P. Bhattarakosol, "Authenticating user using keystroke dynamics and finger pressure," in *Proc. 6th IEEE Conf. Consum. Commun. Netw.*, Las Vegas, NV, USA, 2009, pp. 1078–1079.

[22] N. Bartlow and B. Cukic, "Evaluating the reliability of credential hardening through keystroke dynamics," in *Proc. 17th Int. Symp. Softw. Rel. Eng.*, Raleigh, NC, USA, 2006, pp. 117–126.

[23] A. Kumar, A. Patwari, and S. Sabale, "User authentication by typing pattern for computer and computer based devices," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 3, no. 10, pp. 8132–8134, 2014.

[24] P. S. Teh, S. Yue, and A. B. J. Teoh, "Feature fusion approach on keystroke dynamics efficiency enhancement," *Int. J. Cyber Security Digit. Forensics*, vol. 1, no. 1, pp. 20–31, 2012.

[25] *Keystroke Dynamics Datasets—BU Cyber Security Research Group*. Accessed: Sep. 21, 2018. [Online]. Available: https://cybersecurity.bournemouth.ac.uk/?page_id=624

[26] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Syst.*, vol. 2, no. 3, pp. 321–355, 1988.

[27] A. G. Bors and I. Pitas, "Median radial basis function neural network," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1351–1364, Nov. 1996.

[28] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural Netw.*, vol. 14, nos. 4–5, pp. 439–458, 2001.

[29] O. Y. Al-Jarrah *et al.*, "Data randomization and cluster-based partitioning for botnet intrusion detection," *IEEE Trans. Cybern.*, vol. 46, no. 8, pp. 1796–1806, Aug. 2016.

[30] K. Josien, M. C. Liu, T. W. Liao, and E. Triantaphyllou, "An evaluation of sampling methods for data mining with fuzzy C-means" in *Data Mining for Design and Manufacturing*. Norwell, MA, USA: Kluwer, 2001, pp. 355–369.

[31] F. Aurenhammer and R. Klein, "Voronoi diagrams," in *Handbook of Computational Geometry*. Amsterdam, The Netherlands: Elsevier, 2000, ch. 5, pp. 201–290.

[32] C. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.

[33] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation," in *Encyclopedia of Database Systems*. Boston, MA, USA: Springer, 2009, pp. 532–538.

[34] N. Ye, K. M. A. Chai, W. S. Lee, and H. L. Chieu, "Optimizing F-measures: A tale of two approaches," in *Proc. 29th Int. Conf. Mach. Learn.*, Edinburgh, U.K., 2012, pp. 289–296.

[35] D. M. W. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," *J. Mach. Learn. Technol.*, vol. 2, no. 1, pp. 37–63, 2011.

[36] E. Vural, J. Huang, D. Hou, and S. Schuckers, "Shared research dataset to support development of keystroke authentication," in *Proc. IEEE Int. Joint Conf. Biometrics*, Clearwater, FL, USA, 2014, pp. 1–8.
[37] M. Rybnik, M. Tabedzki, M. Adamski, and K. Saeed, "An exploration of keystroke dynamics authentication using non-fixed text of various length," in *Proc. Int. Conf. Biometrics Kansei Eng.*, Tokyo, Japan, 2013, pp. 245–250.
[38] A. Messerman, T. Mustafić, S. A. Camtepe, and S. Albayrak, "Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics," in *Proc. Int. Joint Conf. Biometrics*, Washington, DC, USA, 2011, pp. 1–8.
[39] A. Alsultan and K. Warwick, "Keystroke dynamics authentication: A survey of free-text methods," *Int. J. Comput. Sci. Issues*, vol. 10, no. 4, pp. 1–10, 2013.
[40] X. Liu, J. F. Beltran, N. Mohanchandra, and G. T. Toussaint, "On speeding up support vector machines: Proximity graphs versus random sampling for pre-selection condensation," *Int. J. Comput. Inf. Technol.*, vol. 7, no. 1, pp. 133–140, 2013.
[41] M. Gamboni *et al.*, "An empirical comparison of support vector machines versus nearest neighbour methods for machine learning applications," in *Proc. Int. Conf. Pattern Recognit. Appl. Methods*, Angers, France, 2014, pp. 110–129.
[42] M. Rizwan and D. V. Anderson, "Investigation on adaptive data condensation for exemplar based method in speech task," in *Proc. IEEE Glob. Conf. Signal Inf. Process.*, Montreal, QC, Canada, 2017, pp. 66–70.

Ioannis Tsimperidis, photograph and biography not available at the time of publication.

Paul D. Yoo (SM'13), photograph and biography not available at the time of publication.

Kamal Taha (SM'18), photograph and biography not available at the time of publication.

Alexios Mylonas (M'14), photograph and biography not available at the time of publication.

Vasilis Katos, photograph and biography not available at the time of publication.