Received February 23, 2020, accepted March 26, 2020, date of publication March 30, 2020, date of current version April 10, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2984214

Convolutional Neural Networks for User Identification Based on Motion Sensors Represented as Images

CEZARA BENEGUI AND RADU TUDOR IONESCU, (Member, IEEE)
Department of Computer Science, University of Bucharest, 010014 Bucharest, Romania

Corresponding author: Radu Tudor Ionescu (raducu.ionescu@gmail.com)

The research leading to these results has received funding from the EEA Grants 2014-2021, under Project contract no. EEA-RO-NO-2018-0496.

ABSTRACT In this paper, we propose a deep learning approach for smartphone user identification based on analyzing motion signals recorded by the accelerometer and the gyroscope, during a single tap gesture performed by the user on the screen. We transform the discrete 3-axis signals from the motion sensors into a gray-scale image representation which is provided as input to a convolutional neural network (CNN) that is pre-trained for multi-class user classification. In the pre-training stage, we benefit from different users and multiple samples per user. After pre-training, we use our CNN as feature extractor, generating an embedding associated to each single tap on the screen. The resulting embeddings are used to train a binary Support Vector Machines (SVM) model in a few-shot user identification setting, i.e. requiring only 20 taps on the screen during the registration phase. We compare our identification system based on CNN features with two baseline systems, one that employs handcrafted features and another that employs recurrent neural networks (RNN) features. All systems are based on the same classifier, namely SVM. To pre-train the CNN and the RNN models for multi-class user classification, we use a different set of users than the set used for few-shot user identification, ensuring a realistic scenario. The empirical results demonstrate that our CNN model yields a top accuracy of 90.75% in multi-class user classification and a top accuracy of 96.72% in few-shot user identification. We thus believe that our system is ready for practical use, having a better generalization capacity than both baselines. We also conduct experiments showing that the binary SVM provides better results than the one-class SVM, although the negative samples added during training do not belong to the attackers (known only at test time).

INDEX TERMS Convolutional neural networks, deep learning, user identification, biometric user identification, motion sensor analysis.

I. INTRODUCTION

Nowadays, common mobile device authentication mechanisms such as PINs, graphical passwords and fingerprint scans offer limited security. These mechanisms are susceptible to guessing (or spoofing in the case of fingerprint scans) and to side channel attacks [1] such as smudge [2], reflection [3], [4] and video capture attacks [5]–[7]. On top of this, a fundamental limitation of PINs, passwords, and fingerprint scans is that these mechanisms require explicit user interaction.

Due to the world wide adoption of mobile devices and the advancement of technologies, mobile devices are now equipped with multiple sensors such as accelerometers, gyroscopes, magnetometers, among others. The data recorded by these sensors during the interaction of the user with the mobile device can be used as biometric data to identify the user. Indeed, one-time or continuous user identification based on the data collected by the motion sensors of a mobile device is an actively studied task [8]–[22], that emerged after the integration of motion sensors into commonly used mobile devices.

In this paper, we propose a novel deep learning approach that can identify the user from a single tap on the smartphone's touchscreen, using the discrete signals recorded by the accelerometer and the gyroscope during the tap gesture. Our approach is not aimed at replacing explicit authentication methods such as face recognition or fingerprint scanning,

The associate editor coordinating the review of this manuscript and approving it for publication was Hazrat Ali.

which have typically higher accuracy rates. It is rather designed to serve as an additional (implicit) verification system. The main advantage is that our system does not require explicit authentication. Instead, we can simply record the motion signals generated while the user performs the authentication with an explicit factor, i.e. face recognition or fingerprint scanning. By adding the proposed implicit verification system while keeping the user's interaction to a minimum (no extra steps are required during verification), we provide an additional layer of security, eliminating many of the attacks enumerated above.

Our approach is based on transforming the discrete 3-axis signals from the accelerometer and the gyroscope into a gray-scale image representation that can be provided as input to deep convolutional neural networks (CNNs) [23], [24]. Our image representation is based on repeating the six one-dimensional (1D) signals using a modified version of de Brujin sequences [25], such that the 3×3 convolutional filters from the first layer of the CNN get to "see" every possible tuple of three 1D signals in their receptive field. After transforming the motion signals accordingly, we pre-train several CNN architectures in a multi-class user classification setting. In the pre-training stage, we can leverage the use of data from multiple users and multiple samples per user. After pre-training and selecting the best-performing CNN, we can employ the selected CNN as a deep feature extractor that generates useful embeddings for each tap gesture. The generated embeddings can then be used to train a lightweight model, e.g. Support Vector Machines (SVM) [26], to identify the user in a few-shot learning setting. We consider a few-shot learning setting with 20 samples per user in order to enable a fast registration process (20 taps on the screen are enough), similar in terms of time to the registration processes used by standard fingerprint or face authentication systems.

We conduct experiments in order to compare our user identification system based on CNN features with two baselines, one that is based on handcrafted features [18] and one that is based on recurrent neural network (RNN) features [17]. All models are evaluated in a few-shot user identification context using the same classifier, namely a binary SVM. Our CNN model (as well as the baseline RNN model) is pre-trained on a multi-class user classification task. The users involved in the multi-class user classification experiment are different from those involved in the user identification experiment, to simulate a realistic scenario. In order to conduct our experiments, we modify the HMOG data set [20] by extracting shorter signals from the original sessions and by splitting the users in half, using the first half for the preliminary multi-class user classification experiment and the second half for the user identification experiment. Our binary SVM based on CNN features exhibits a higher generalization capacity, surpassing both baselines in the user identification experiments. Moreover, according to McNemar's statistical testing [27] performed at a confidence level of 0.01, our improvements over the baseline based on handcrafted features are significant. With an accuracy of 96.72%, our SVM based on CNN features seems to be a viable solution for practical usage.

We note that user identification is typically modeled as an anomaly detection task, in which the anomalies are produced by attackers. Therefore, researchers [18]–[20] commonly employ outlier detection models, e.g. one-class SVM. We thus compare our binary SVM, which is trained on negative samples that do not belong to the attackers (known only at test time), with the one-class SVM. The results show the benefit of adding negative samples during training, even if the added data samples are not collected from the attackers, for a fair comparison.

In summary, our contribution is threefold:

- We propose a novel gray-scale image representation of the discrete signals, designed specifically to be useful as input for CNNs.
- We propose to pre-train CNNs on a multi-class user classification task in order to obtain useful embeddings for few-shot user identification.
- We perform comparative experiments showing that our method based on CNN embeddings surpasses both machine learning methods based on handcrafted features and deep learning methods based on RNN embeddings.

The rest of this paper is organized as follows. In Section II, we provide an overview of user identification systems for mobile devices, focusing mainly on systems based on analyzing motion sensors. In Section III, we present the proposed data representation, our CNN architectures and our user identification model. In Section IV, we present the data set, the evaluation metrics and the performed experiments. In Section V, we conclude our findings and propose some future directions of study.

II. RELATED WORK

The first studies in smartphone user identification used keystroke dynamics [28]–[30], since the early smartphone devices were equipped with hardware keyboards. To our knowledge, the first study to propose the analysis of accelerometer data in order to recognize the gait of a mobile device user appeared in 2006 [31]. The approach proposed by Vildjiounaite *et al.* [31] is to directly measure the similarity between the sample of signal recorded during authentication and a previously-recorded sample of signal that belongs to the user. The samples are compared based on statistical features extracted in the time domain or the frequency domain, without using machine learning. More recent studies explored the task of user identification based on machine learning models [8]–[22], attaining better results compared to statistical models such as [31]. By modeling the user identification task based on motion sensors as a classification task, various models following the standard training and evaluation pipeline used in machine learning can be tested out. The standard machine learning pipeline is essentially based on two steps. The first step is to extract handcrafted features from the discrete motion signals in the time domain or the frequency domain. The second step is to apply a standard machine learning classifier.

However, some recent works [9]–[11], [19], [20] have proposed to change the standard pipeline in order to obtain improved performance. For instance, the method proposed by Shi *et al.* [19] uses different modalities for user identification. The approach builds a one-class classifier for each modality and aggregates the results using a meta-classifier. Another approach that employs multiple modalities is proposed by Buriro *et al.* [9]. Their method is based on motion patterns recorded by the motion sensors and voice patterns recorded by the microphone during a phone call. Some works modify the standard pipeline by adding a feature selection step before classification. The approach described by Sitova *et al.* [20] uses Principal Component Analysis for feature selection. The authors extracted statistical features specifically designed for tap gestures on the touchscreen of the mobile device. We also analyze tap gestures, but we extract deep CNN features (instead of statistical features) from the motion signals recorded during the taps. The approach proposed by Buriro *et al.* [11] performs user authentication using data recorded by the motion sensors as well as the touchscreen of the mobile device. The authors performed feature selection using the Recursive Feature Elimination method. There are other approaches [10] that use the Information Gain for feature selection.

The methods described in [17], [21] combine the two standard steps (feature extraction and classification) into a single step, by training deep neural network models in an end-to-end fashion. Similar to these works [17], [21], we propose an approach based on deep neural networks. Neverova *et al.* [17] proposed the use of recurrent connections to model temporal variations of the signals. We take a different approach and propose to use convolutional neural networks in order to obtain deep embeddings of the motion signals recorded by the accelerometer and the gyroscope. It is important to note that Neverova *et al.* [17] and Sun *et al.* [21] used convolutional neural networks as baselines, showing better results with their recurrent neural networks. While Neverova *et al.* [17] used the discrete temporal signals as input for their baseline CNNs, we propose a novel approach to convert the temporal signals into a 2D gray-scale image representation to be used as input for our CNN. As shown in our user identification experiments, the embeddings learned by our CNN are more robust than the embeddings learned by the baseline RNN, leading to significant performance improvements. Another approach based on training neural networks for biometric user identification is proposed by Sun *et al.* [21]. While they use recurrent neural networks to recognize users based on keystroke dynamics, we propose to use convolutional neural networks to recognize users based on motion data recorded during screen taps. While works such as [17], [21] require longer interactions from the user, our method is designed to identify the user based on 1.5 seconds of motion signals recorded during a single tap gesture. Furthermore, our method requires only 20 samples (taps) during the user registration phase.

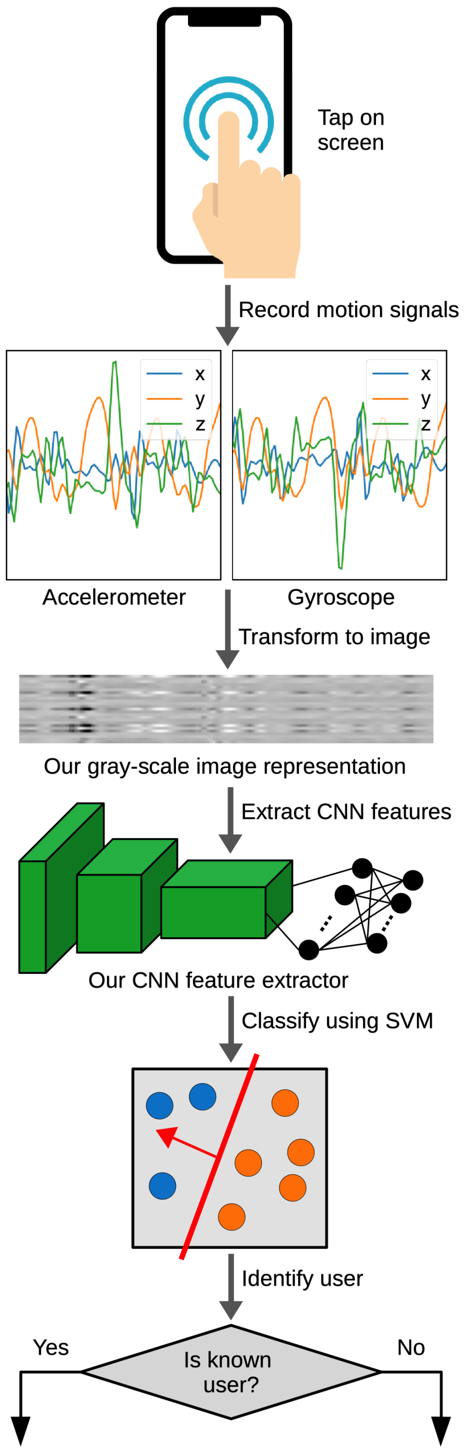

FIGURE 1. Our user identification pipeline based on analyzing motion signals recorded during the user's screen tap. The signals are combined into a gray-scale image representation which is provided as input to a CNN that is pre-trained on multi-class user classification. A binary SVM trained on a few examples is used to identify the user. Best viewed in color.

III. METHOD

A. IMAGE REPRESENTATION

As illustrated in Figure 1, our first step is to turn the discrete signals acquired from the smartphone's accelerometer and gyroscope sensors into gray-scale images. We start with six discrete signals of potentially different lengths, represented in the time domain. Although the motion signals are supposed to be recorded at 100 Hz, depending on the processes running on the mobile device, the operating system will not report exactly 100 values per second at perfectly equal time-intervals. Furthermore, the accelerometer and the gyroscope report motion events independently. Although the signals reported for the three axes of a motion sensor are of the same length, the signals reported by two different motion sensors could be of different lengths. We thus have to normalize the motion signals to a fixed length. Since we record signals for 1.5 seconds at 100 Hz, we expected them to be formed of 150 discrete values. Hence, the signals that are longer or shorter are resized using linear interpolation to a fixed length of 150 values. After resizing a signal, we subtract its minimum magnitude such that each value becomes positive. Each signal is now a vector of 150 positive components. Since the independent signals can have different magnitude scales, e.g. when the motion projected on one axis is much higher in terms of magnitude than the motion projected on another axis, we independently normalize the vectors using the L_2-norm. Then, we rescale the values to the interval $[0, 255]$, in order to use the full range of values available for gray-scale images.

Next, we choose to consider each 1D discrete signal as a row vector and concatenate the vectors column-wise in a specific sequence that allows repetition. The sequence is chosen by taking into consideration that recent CNN architectures, e.g. GoogLeNet [32] and ResNet [33], use convolutional filters with a receptive field (spatial support) not higher than 3×3. Given that we want to employ such modern design recommendations into our CNN architectures, we need to make sure that every possible tuple of three 1D signals is "seen" by the first convolutional layer. We build our sequence of signals based on the principles of de Brujin sequences [25]. We first associate digits from 0 to 5 to identify our signals in the generated sequence. A *de Brujin sequence* of order n (length of tuples, in our case, $n = 3$) on an alphabet Σ of size k (number of signals, in our case, $k = 6$) is a cyclic sequence in which every possible string of length n (in our case, triplet) on Σ (in our case, $\Sigma = \{0, 1, 2, 3, 4, 5\}$) occurs exactly once as a contiguous subsequence. Different from de Brujin sequences, we do not need to include triplets of signals that represent rearrangements of triplets that are already included in our sequence. We also do not need triplets of repeating symbols, e.g. $(3, 3, 3)$. In order to build our sequence of signals we employ a simple Greedy algorithm, which adds the minimum amount of signals to the sequence in order to include a new triplet, which was not previously included in the sequence. While the minimum length of a de Brujin sequence for $n = 3$ and $k = 6$ is 218, we obtain a shorter sequence of length 25. Our sequence is 0, 1, 2, 3, 4, 5, 0, 2, 4, 5, 1, 3, 0, 4, 1, 2, 5, 3, 0, 2, 0, 5, 1, 3, 4. We consider our sequence as a pseudo-de Brujin sequence. Using the generated pseudo-de Brujin sequence, we build our gray-scale image representation of 25×150 pixels. In Figure 2, we illustrate some image

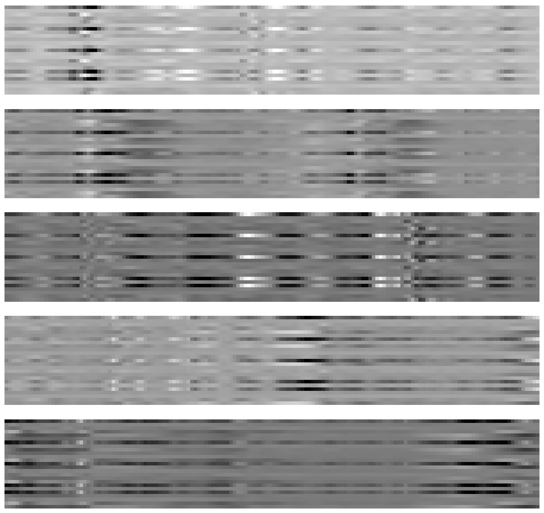
FIGURE 2. Gray-scale images of 25 × 150 pixels resulted after our conversion of the discrete signals recorded at 100 Hz for 1.5 seconds.

representations constructed for a set of randomly selected recording sessions, each of 1.5 seconds in length. After obtaining the image representations, we provide them as the input for the convolutional neural networks described in the following section.

B. CNN-BASED FEATURE EXTRACTION

Recent methods for object recognition [24], [32]–[34] and other computer vision tasks [35]–[40] are based on deep learning [41], [42]. The main approach in this area is represented by CNNs [24], [32], [34]. Convolutional neural networks are a particular type of feed-forward neural networks that are designed to efficiently process images through the use of a special kind of layer inspired by the human visual cortex, namely the *convolutional layer*. Following the success of transfer learning from pre-trained CNNs [38]–[40], [43], we consider them as potentially-useful feature extractors for smartphone user identification, given our custom gray-scale image representation derived from motion signals. Instead of considering pre-trained CNN models on ImageNet [44] as others [38]–[40], we devise a multi-class user classification task in order to train our models, before transferring them to the user identification task. This ensures that our CNN models are specifically adapted to the same kind of input images.

In this work, we propose four CNN architectures of different depths. Each architecture is composed of a different number of convolutional (conv) layers followed by a fixed number of fully-connected (fc) layers. We use Rectified Linear Units (ReLU) [45] as activation functions on all layers, except for the classification layer which has Softmax activations.

Our first CNN architecture is composed of a conv layer, 2 fc layers and a Softmax classification layer. The conv layer is composed of 32 filters with a 3×3 spatial support. The filters are applied at a stride of 1 and the input is zero-padded to preserve the spatial dimension. The convolutional layer is followed by a max-pooling layer with a pool size of 2×2. The max-pooling layer is followed by 2 fc layers, each of 256 neurons. We use dropout [46] on each fc layer, with the dropout rate set to 0.4. The fourth and final layer is a fc (Softmax classification) layer with 50 neurons, corresponding to the number of classes (users) from our multi-class user classification task.

The second CNN architecture is composed of 6 layers, namely 3 conv layers, 2 fc layers and a Softmax classification layer. The first conv layer is composed of 32 filters with a 3×3 spatial support. The filters are applied at a stride of 1 and the input is zero-padded to preserve the spatial dimension. The second conv layer is composed of 64 filters, while the third conv layer is composed of 128 filters. As the first conv layer, the filters from the second and the third conv layers have a receptive field of 3×3 components and are applied at a stride of 1. All activation maps are zero-padded to preserve the spatial dimension. Each convolutional layer is followed by a max-pooling layer with a pool size of 2×2. After the last max-pooling layer, we have 2 fc layers with identical settings as in the first CNN architecture. The sixth and final layer is a Softmax classification layer composed of 50 neurons.

The third CNN architecture is composed of 9 layers, namely 6 conv layers, 2 fc layers and a Softmax classification layer. The 9-layer CNN architecture is derived from the 6-layer CNN architecture, by replicating each conv layer exactly once. Therefore, the first and the second conv layers of the 9-layer CNN have 32 filters, as the first conv layer of the 6-layer CNN. Similarly, the third and the fourth conv layers of the 9-layer CNN have 64 filters, as the second conv layer of the 6-layer CNN. The same rule applies to the fifth and the sixth conv layers of the 9-layer CNN, which have 128 filters. In the 9-layer CNN, only the second, the fourth and the sixth conv layers are followed by max-pooling layers with a pool size of 2×2. The other layers and parameters are the same as in the 6-layer CNN architecture.

Our fourth and deepest CNN architecture has 12 layers, namely 9 conv layers, 2 fc layers and a Softmax classification layer. The 12-layer CNN architecture is derived from the 6-layer CNN architecture, by replicating each conv layer exactly twice. The first 3 conv layers contain 32 filters, the following 3 conv layers contain 64 filters and the last 3 conv layers contain 128 filters. In the 12-layer CNN, only the third, the sixth and the ninth conv layers are followed by max-pooling layers. We note that, in the 12-layer CNN architecture, we preserve the other (fc) layers and parameters (stride, kernel size, pool size, dropout rate) from the previously-presented CNN architectures.

All models are trained using the Adam optimizer [47] with the categorical cross-entropy loss function. We opted for the Adam optimizer because Kingma *et al.* [47] demonstrated that it converges faster than other stochastic gradient descent alternatives. We train and test our CNN architectures on the multi-class user classification task, in order to select the best-performing architecture. After finding the best CNN model, we remove its Softmax layer and use the activation maps from the last remaining fc layer as feature vectors (deep embeddings). Given that the last fc layer is formed of 256 neurons irrespective of the CNN architecture, we will obtain 256-dimensional embeddings.

C. FEW-SHOT USER IDENTIFICATION

In a realistic user identification scenario, we do not have access to a set of samples performed by attackers (impersonators) in order to build a binary classification model. Therefore, we can either train an outlier detection model, e.g. a one-class SVM [19], or use a pool of negative data samples that belong neither to the rightful user nor to the attackers and train a binary classifier. We opt for the second approach and model the user identification task as a binary classification task, in which the positive samples belong to the rightful user and the negative samples belong to a set of users that have nothing in common with the attackers (involved only at test time). In this context, we employ the binary SVM classifier. Nevertheless, we later show empirical evidence in favor of binary SVM over one-class SVM.

For binary classification problems, kernel classifiers [48], such as SVM, look for a discriminative function g that assigns positive labels (+1) to examples that belong to one class and negative labels (−1) to examples that belong to the other class. The function g is linear in the feature space and can be expressed as follows:

$$g(x) = \text{sign}(\langle w, x \rangle + b), \tag{1}$$

where x is a feature vector, w and b are the weight vector and the bias term learned by the kernel classifier and $\langle \cdot, \cdot \rangle$ is the dot product. In our case, x is a 256-dimensional feature vector provided by a CNN.

Different kernel classifiers may use different criteria to find an optimal vector of weights. The SVM classifier [26] aims at finding the vector w and the bias b that define the hyperplane which separates the training samples by a maximum margin. Mathematically, the SVM classifier chooses the weights w and the bias term b that satisfy the following optimization criterion:

$$\min_{w,b} \frac{1}{n} \sum_{i=1}^{n} [1 - y_i(\langle w, x_i \rangle + b)]_+ + C\|w\|^2, \tag{2}$$

where n is the number of training samples, y_i is the label (+1 or −1) of the training example x_i, C is a regularization parameter, $[x]_+ = \max\{x, 0\}$ and $\|\cdot\|^2$ is the L_2-norm.

Kernel classifiers rely on a kernel function to embed the data in a high-dimensional space, in which non-linear relations become linear. A kernel function captures the intuitive notion of similarity among pairs of data samples from a specific domain, and can be any function defined on the respective domain that is symmetric and positive definite. We opt for two popular kernel functions, namely the linear kernel, which is given by the dot product between pairs of samples, and the Radial Basis Function (RBF) kernel, which is given by the following equation:

$$k_{RBF}(x_i, x_j) = exp(-\gamma \|x_i - x_j\|^2), \tag{3}$$

where x_i and x_j are two data samples, $exp(\cdot)$ is the exponential function and γ is a parameter that controls the range of possible output values for the RBF kernel.

IV. EXPERIMENTS

A. DATA SET

In order to successfully test the identification of users, we use a data set which contains values from two motion sensors, the accelerometer and the gyroscope, collected from 100 users while sitting [20]. We record the motion sensor values while users perform a single tap on the screen. The recording starts with 0.5 seconds before the tap event and ends with 1 second after the tap event. Both sensors report values on three axes (x, y, z) at about 100 Hz. Hence, an example is composed of six discrete signals, three from each sensor, corresponding to the three axes (x, y, z), respectively. Given that each signal is recorded for 1.5 seconds at about 100 Hz, it is represented by roughly 150 values. For each user, we collect motion signals for the first 200 tap events. In total, we have 20.000 data samples.

The data set is randomly split into two equal parts, such that each half contains a disjoint set of 50 users (users in one half are different from users in the other half). The first part of the data set, containing 50 users (with 200 samples per user), is used to train the neural models for the task of multi-class user classification. We use 160 samples per user for training and the rest of 40 samples per user for validation, corresponding to an 80%–20% split of the data. Hence, there are 8.000 samples for training and 2.000 samples for validation. The first part of the data set is also used for hyperparameter tuning of the neural models and the SVM models.

The second part of the data set, containing the other 50 users (with 200 samples per user), is used for the user identification experiments. To simulate a realistic setting, we train a binary SVM classifier for each user, including only 20 positive samples and 100 negative samples. Since we have 50 users, we obtain 50 binary classification problems. The binary models are tested on 100 positive samples and 100 negative samples. When a binary model is trained and tested, we are careful to select the negative training and test examples from disjoint subsets of users. By using a disjoint subset of users during inference, we make sure the classification models do not cheat by making use of features specific to the negative users seen during training. This ensures that our user identification experiments simulate a realistic setting, in which samples coming from potential impersonators are not available during training. On the second part of the data set, we also compare the binary SVM models with one-class SVM models that employ similar features and settings, to ensure a fair comparison. The only difference is that the one-class models are trained on 20 positive samples (the 100 negative samples are simply excluded).

TABLE 1. Confusion matrix (also known as contingency table) of a binary classifier with labels +1 or −1.

		Ground-truth labels	
	Labels	$+1$	-1
Classifier predictions	$+1$	true positive (TP)	false positive (FP)
	-1	false negative (FN)	true negative (TN)

B. EVALUATION METRICS

In order to evaluate the deep learning models on the multi-class user classification task, we employ the classification accuracy. To evaluate our baseline and proposed models on the user identification task, we compute the binary confusion matrix for each user. The binary confusion matrix contains the number of true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN), as shown in Table 1. We then extract metrics such as the accuracy (ACC), the false acceptance rate (FAR) and the false rejection rate (FRR).

The *accuracy* of the model is given by the total number of correct predictions divided by the total number of predictions. Thus, the accuracy is given by:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4}$$

The *false acceptance rate* is the ratio between the number of false acceptances and the sum of all negative samples (false positives and true negatives):

$$FAR = \frac{FP}{FP + TN}. \tag{5}$$

The *false rejection rate* is the ratio between the number of false rejections and the sum of all positives samples (false negatives and true positives):

$$FRR = \frac{FN}{FN + TP}. \tag{6}$$

C. BASELINES

In this subsection, we present in detail the baseline methods. The first baseline method is based on a recent work by Shen *et al.* [18], that uses handcrafted features. The second baseline method is based on another recent work by Neverova *et al.* [17], that uses features learned by Long Short-Term Memory (LSTM) networks [49] with convolutional layers (ConvLSTM). The LSTM is a type of RNN architecture that solves the vanishing gradient problem known to affect RNN models. As for our four CNN models, the ConvLSTM is pre-trained on the multi-class user classification task, then used as feature extractor on the user identification task. In the user identification experiments, both baseline models employ an SVM classifier, for a fair comparison to our CNN model.

1) BASELINE BASED ON HANDCRAFTED FEATURES

The values recorded by the motion sensors cannot be used directly as input for a user classifier, as the signals contain a large amount of noise. In order to extract useful characteristics for a given user, one approach is to compute several statistical features that can embed some particularities of the users from our data set. Therefore, on each of the six discrete 1D signals that form an example, we compute the following handcrafted features, as Shen *et al.* [18]:

- **Mean** — is the mean value of a discrete 1D signal;
- **Min** — is the minimum value of a discrete 1D signal;
- **Max** — is the maximum value of a discrete 1D signal;
- **Variance** — represents the variance value of a discrete 1D signal;
- **Kurtosis** — describes the width of the peak of a discrete 1D signal;
- **Skewness** — represents the orientation of the peak of a discrete 1D signal;
- **Quantiles** — are the quantiles of a discrete 1D signal, computed from 30% to 80%, increasing by a 10% step.

Moreover, we add to the handcrafted features, Pearson and Kendall's Tau correlation values between all the possible combinations of 1D signal pairs, irrespective of sensor type. An example is thus represented by 72 statistical features (12 features $\times$ 6 signals) and 30 correlation features (2 features $\times$ 15 signal pairs). We provide the handcrafted features as input to an SVM classifier.

2) BASELINE BASED ON CONVLSTM FEATURES

In this section, we present another baseline method, which is based on a pre-trained ConvLSTM as feature extractor. As for our four CNN models, we resize the recorded 1D signals to a fixed length of exactly 150 components. We resize the signals using linear interpolation and obtain an input size of 6 $\times$ 150 components for the ConvLSTM. Since the LSTM network handles temporal inputs directly, we do not need to convert the temporal signals to gray-scale images, as for the CNN models.

The architecture of the ConvLSTM is composed of 6 layers and is similar in size to our best-performing CNN architecture. The first layer of the model is a convolutional LSTM layer with 64 filters and a kernel size of 1 $\times$ 3. In this layer, we use ReLU [45] as the activation function. The first layer is followed by another convolutional LSTM layer, having the same kernel size and the same activation function, but a higher number of filters, i.e. 128. The second convolutional LSTM layer is followed by yet another convolutional LSTM layer with 256 filters. Typical LSTM architectures have no more than two or three LSTM layers, which are sufficient to model the temporal variations of the input signals. Therefore, after the third convolutional LSTM layer, the activation maps are flattened. Then, we have a fully-connected layer of 256 neurons with ReLU activations, followed by another fully-connected layer of 256 neurons with ReLU activations. The sixth and final layer is the classification layer, which contains 50 neurons (corresponding to the number of users in the multi-class user classification data set) having Softmax activations. The chosen loss function is the categorical cross-entropy. As for our CNN models, we employ the Adam optimizer [47]. We compare our four CNN architectures with the ConvLSTM architecture in terms of the number of learnable parameters in Table 2. We observe that the CNN models have about the same number of parameters as the ConvLSTM model. In all CNN architectures, the first fully-connected layer has the largest number of parameters (over 2.4 million). Because we variate only the number of conv layers, the total number of parameters does not change by much from one CNN to another. We note that the 4-layer CNN has slightly more parameters than the other CNN models because it has only one max-pooling layer and the activation maps just before the first fc layer are larger in size.

TABLE 2. Number of learnable parameters of the considered deep neural networks based on convolutional or convolutional LSTM layers.

Model	Number of parameters
6-layer ConvLSTM	3,181,874
4-layer CNN	4,040,000
6-layer CNN	3,055,154
9-layer CNN	3,248,914
12-layer CNN	3,442,674

After training the ConvLSTM model on the multi-class user classification task, we remove the classification layer and use the activation maps from the last fully-connected layer of 256 neurons as features. Hence, we obtain 256 features, which we provide as input to an SVM classifier.

D. PARAMETER AND IMPLEMENTATION CHOICES

For our four CNN models, we set the hyperparameters as described next. We train each CNN model for 50 epochs using a learning rate of 10^{-3}. By applying early stopping to prevent overfitting, the training stops after about 40 epochs for every model. We also considered weight decay in order to prevent overfitting, but the results were slightly worse. We use dropout [46] on the two fully-connected layers, using a dropout rate of 0.4. Regarding the mini-batch size, we experiment with four sizes: 16, 32, 64 and 128. We hereby note that Jastrzębski *et al.* [50] observed that the learning rate and the mini-batch size are strongly correlated. Furthermore, they note that similar performance can be obtained with specific combinations of learning rate and mini-batch size, i.e. the learning rate should be proportional to the mini-batch size. Instead of trying various learning rates and mini-batch size combinations, we decided to fix the learning rate to 10^{-3} and find the optimal mini-batch size that corresponds to our fixed learning rate. After conducting the experiments on multi-class user classification (presented in Section IV-E), we decided to use the shallower 6-layer CNN with a mini-batch size of 32 in the subsequent experiments. With the CNN architecture and the mini-batch size fixed, we proceed by trying out different image representations as input. Based on the empirical evidence, we decided to use the image representation based on de Brujin sequences. We also explored thinner or wider architectures which produce 128 or 512 features instead of 256. Here, we opted for the CNN model that produces 256-dimensional feature vectors. Further details are provided in Section IV-E.

For the baseline ConvLSTM model, we keep the same hyperparameters as for the CNN models, for a fair comparison. We thus set the learning rate to 10^{-3} and train the model for 50 epochs using mini-batches of 32 samples. By applying early stopping to prevent overfitting, the training stops after 40 epochs. We use dropout on the two fully-connected layers, using a dropout rate of 0.4. The ConvLSTM model produces 256-dimensional feature vectors, just as the CNN model.

For the binary or the one-class SVM models, we try out two kernels, namely the linear kernel and the RBF kernel. For the RBF kernel, we set the parameter γ as follows:

$$\gamma = \frac{1}{m \cdot Var(X)}, \tag{7}$$

where m is the number of features, X is a matrix containing the training data and $Var(\cdot)$ is a function that computes the variance.

In order to obtain optimal results, we adjust the SVM model by tuning the regularization parameter C using grid search on the multi-class user classification data set. As possible values for C, we consider values in the set $\{0.1, 1, 10, 100\}$. When using the linear kernel, the best value for the parameter C is 1, irrespective of the features (handcrafted or deep). When using the RBF kernel, we obtained better results with $C = 100$ for the handcrafted features and $C = 1$ for the deep (CNN or ConvLSTM) features. The bias term of each SVM model is independently adjusted, such that the difference between the FAR and the FRR is less than 1%. This ensures a fair comparison in terms of accuracy between models.

While the neural models are implemented in TensorFlow [51], we employ the Scikit-learn [52] implementation of the binary and the one-class SVM.

E. MULTI-CLASS USER CLASSIFICATION RESULTS

In Table 3, we present the results obtained by our four different CNN architectures on the multi-class user classification task. Since neural networks are sensitive to the initialization of the weights, we train each model for 5 times, reporting the average results. The 4-layer CNN attains the worst results, having some trouble even in fitting on the training data. We note that, as the architecture grows from 6 layers to 12 layers, the validation accuracy tends to drop slightly. Given that our training data is limited to 8.000 examples, with 160 samples per class, we conclude that the deeper networks are perhaps too deep with respect to our training set size. Besides trying different architectures, we also tested various mini-batch sizes: 16, 32, 64 or 128. The empirical results presented in Table 3 indicate that a mini-batch size of 32 is the optimal value for all CNN architectures. While the 6-layer CNN trained on mini-batches of 16 samples yields a better training accuracy (94.77%), the 6-layer CNN trained on mini-batches of 32 samples has a stronger generalization capacity, attaining an accuracy of 90.75% on the validation set. We thus choose the 6-layer CNN based on mini-batches of 32 samples for the subsequent experiments.

TABLE 3. Train and validation accuracy rates of our CNN architectures with various depths, on the multi-class user classification task. Each architecture is trained with four different mini-batch sizes. The best results are highlighted in bold.

Model	Batch size	Accuracy	
		Training	Validation
4-layer CNN	16	83.21%	84.80%
	32	88.82%	87.30%
	64	68.90%	83.70%
	128	56.96%	76.75%
6-layer CNN	16	**94.77%**	88.85%
	32	94.17%	**90.75%**
	64	91.56%	89.15%
	128	90.85%	88.35%
9-layer CNN	16	88.80%	85.75%
	32	93.62%	90.25%
	64	93.16%	88.85%
	128	91.82%	87.40%
12-layer CNN	16	92.26%	85.90%
	32	93.56%	88.50%
	64	92.25%	87.60%
	128	89.30%	84.35%

TABLE 4. Train and validation accuracy rates of our 6-layer CNN architecture with various image representations, on the multi-class user classification task. The best results are highlighted in bold.

Model	Image representation	Accuracy	
		Training	Validation
6-layer CNN	zero-padding	94.13%	90.05%
	pseudo-de Brujin sequence	**94.17%**	**90.75%**
	all triplet arrangements	90.52%	87.80%

Having determined the optimal CNN architecture, we performed a set of experiments to determine if the optimal image format should be based on de Brujin sequences. We compare our representation generated with the help of a pseudo-de Brujin sequence with two baselines, one that considers a single copy for each of the 6 signals (zero-padding the input until it reaches the same size as our image representation) and one that considers all possible triplets formed of the 6 signals. We report the corresponding results in Table 4. The empirical results indicate that formatting the input image using a pseudo-de Bruijn sequence is a better choice than applying zero-padding or than using all arrangements at once. For the rest of the experiments, we opt for our image representation generated as detailed in Section III-A.

After setting out the CNN architecture and the optimal image format, we conducted further experiments to determine the optimal size of the embedding. In addition to the original 6-layer CNN that produces 256-dimensional embeddings, we try out a thinner model producing 128-dimensional embeddings and a wider model producing 512-dimensional embeddings. The corresponding results are presented in Table 5. We note that Table 5 contains the average accuracy rates computed over 5 runs for each model. While the wider CNN seems to fit better on the training set, yielding an accuracy of 95.92%, it does not surpass the CNN producing 256-dimensional embeddings, on the validation set.

TABLE 5. Train and validation accuracy rates of our 6-layer CNN architecture with various embedding sizes, on the multi-class user classification task. The best results are highlighted in bold.

Model	Embedding size	Accuracy	
		Training	Validation
6-layer CNN	128	93.28%	89.15%
	256	94.17%	**90.75%**
	512	**95.92%**	86.95%

TABLE 6. Train and validation accuracy rates of our 6-layer CNN architecture versus the 6-layer ConvLSTM, on the multi-class user classification task. Both models are trained with mini-batches of 32 samples and produce 256-dimensional embeddings. The marker $*$ indicates that the CNN model is significantly better than the ConvLSTM model, according to a paired McNemar's test performed at a significance level of 0.01. The best results are highlighted in bold.

Model	Accuracy	
	Training	Validation
6-layer ConvLSTM	87.92%	88.35%
6-layer CNN	**94.17%***	**90.75%***

The thinner CNN attains the lowest accuracy rates on both training and validation sets. In conclusion, we decided to stick with the CNN architecture that gives us 256-dimensional feature vectors.

Our final aim is to use our best CNN model as a pre-trained feature extractor for the user identification task. For a fair comparison, we apply the ConvLSTM in a similar way, i.e. as a pre-trained feature extractor. Therefore, the first step is to train the ConvLSTM on the multi-class user classification task. The corresponding accuracy rates are presented in Table 6. It is important to note, once again, that the hyper-parameters of the 6-layer ConvLSTM are similar to our best 6-layer CNN (see Section IV-D). Compared to our CNN, it seems that the LSTM units are not able to properly capture the particularities of the discrete temporal signals. The validation accuracy of the ConvLSTM (88.35%) is 2.4% below the validation accuracy of our CNN (90.75%). As it currently seems, our CNN is a model with higher learning capacity than the ConvLSTM. Indeed, our model attains significantly higher training and validation accuracy rates, according to a paired McNemar's test performed at a significance level of 0.01.

F. USER IDENTIFICATION RESULTS

In Table 7, we present the comparative results on the few-shot user identification task. The experiments are aimed at validating our modeling choices regarding feature extraction and classification. First of all, we compare our binary SVM classifier based on CNN features with two baseline binary SVM classifiers, one based on handcrafted features and one based on ConvLSTM features. This comparison is aimed at revealing the best feature extraction method. Second of all, we compare the binary SVM models with one-class SVM models, for all three kinds of features. This comparison is aimed at showing that the idea of modeling user identification as a binary classification problem is viable. For each binary or one-class SVM model, we experiment with two kernel functions, the linear kernel and the RBF kernel.

Regarding the kernel functions, we note that the RBF kernel gives generally better results than the linear kernel, the only exception being the model based on CNN features. In the case of handcrafted features, the number of features (72) is smaller than the number of training samples (120). Since the classification problem is likely not linearly separable (because we have less features than data samples), the SVM based on handcrafted features benefits from the use of the RBF kernel, which is known to embed the features in a higher-dimensional space, in which samples can be linearly separated. As the feature vectors provided by the CNN contain 256 features, the 120 training samples are already linearly separable (because we have more features than data samples). Further increasing the feature space through the use of the RBF kernel, might lead to a typical case of the Hughes phenomenon, i.e. the models start to suffer from the curse of dimensionality [53]. This seems to be the case for the CNN features, since the accuracy of the binary SVM based on the RBF kernel (96.37%) is lower than the accuracy of the binary SVM based on the linear kernel (96.72%). Nevertheless, it seems that the RBF kernel brings a large performance gain for the ConvLSTM features (from 83.16% to 96.18%). Overall, it seems that the RBF kernel is a better choice than the linear kernel.

With respect to the features, we note that the binary SVM based on handcrafted features attains accuracy rates between 83% and 88%. We believe that these accuracy rates are not high enough for the system to be used in practice. The binary SVM based on ConvLSTM features yields accuracy rates between 83% and 96%, while our binary SVM model based on CNN features surpasses both baselines, attaining accuracy rates between 96% and 97%. We believe that the performance gap between the ConvLSTM features and our CNN features is caused by the fact that the ConvLSTM model has a lower modeling capacity. Interestingly, our results confirm the recent trends from the deep learning community, advocating in favor of using alternative approaches instead of RNN and LSTM architectures in order to model temporal data, e.g. by employing attention mechanisms [54]. Overall, we conclude that our CNN features represent a better choice than the handcrafted or the ConvLSTM features.

Regarding the classifier, we observe that the binary SVM attains better results than the one-class SVM, with only one exception: the linear kernel applied on ConvLSTM features. Similar to the binary SVM, it seems that the one-class SVM works better in combination with the RBF kernel rather than the linear kernel. We believe that the high accuracy differences between the binary SVM and the one-class SVM are due to the fact that the one-class SVM has to find an optimal boundary relying only on positive training data samples. We thus conclude that the idea of including negative training samples is useful, even if the added samples do not belong to the attackers from the test set.

We also performed McNemar's statistical tests [27], at a confidence level of 0.01, in order to determine if the differences between the handcrafted and the deep features

TABLE 7. Few-shot user identification results provided by our binary SVM based on CNN features versus two baselines, a binary SVM based on handcrafted features and a binary SVM based on ConvLSTM features. Results obtained with one-class SVM instead of binary SVM are also included. All results are reported for two kernel functions, linear and RBF. The reported accuracy, FAR and FRR values represent the average values determined on the 50 users involved in the few-shot user identification task. The best result on each column is highlighted in bold. The marker $*$ indicates that the corresponding model is significantly better than the binary SVM baseline based on handcrafted features, according to a paired McNemar's test performed at a significance level of 0.01. The marker $\diamond$ indicates that the binary SVM is significantly better than the corresponding one-class SVM, according to a paired McNemar's test performed at a significance level of 0.01.

Model	Kernel	Accuracy	FAR	FRR
Handcrafted features + one-class SVM	linear	70.12%	29.40%	30.36%
	RBF	80.16%	20.12%	19.56%
Handcrafted features + binary SVM	linear	83.80%$^{\diamond}$	16.50%$^{\diamond}$	15.90%$^{\diamond}$
	RBF	87.31%$^{\diamond}$	12.44%$^{\diamond}$	12.95%$^{\diamond}$
ConvLSTM features + one-class SVM	linear	83.50%*	16.60%*	16.40%*
	RBF	92.63%*	7.12%*	7.62%*
ConvLSTM features + binary SVM	linear	83.16%*	17.08%*	16.61%*
	RBF	96.18%$^{*,\diamond}$	4.00%$^{*,\diamond}$	3.64%$^{*,\diamond}$
CNN features + one-class SVM	linear	89.66%*	10.48%*	10.20%*
	RBF	90.53%*	9.32%*	9.62%*
CNN features + binary SVM	linear	**96.72%**$^{*,\diamond}$	**3.10%**$^{*,\diamond}$	**3.45%**$^{*,\diamond}$
	RBF	96.37%$^{*,\diamond}$	3.30%$^{*,\diamond}$	3.96%$^{*,\diamond}$

(ConvLSTM and CNN) are statistically significant. Without exception, the accuracy rates reached by our binary SVM model based on CNN features are significantly better than the accuracy rates of the binary SVM based on handcrafted features. We also note that the binary SVM based on ConvLSTM features is significantly better than the binary SVM based on handcrafted features. We perform another set of statistical tests to verify if the accuracy differences between the binary SVM models and the corresponding one-class SVM models are significant. The McNemar's statistical tests reveal that the binary SVM models are in most cases significantly better than the one-class SVM models.

The empirical results presented in Table 7 indicate that the best approach is to employ a binary SVM based on the linear kernel computed over CNN embeddings.

V. CONCLUSION

In this paper, we have presented an approach based on pre-trained CNN features that can identify (authenticate) users by analyzing data recorded by motion sensors incorporated in mobile devices, while the user performs a single tap gesture on the screen. Our approach is based on transforming the discrete signals from motion sensors into a gray-scale image representation which is then provided as input to a convolutional neural network (CNN) that is pre-trained on a multi-class user classification task. After pre-training, we used the CNN as feature extractor, generating an embedding associated to each single tap on the screen. We compared our user identification system based on CNN features with two baseline systems, one that employs handcrafted features and another that employs ConvLSTM features. All systems are based on the SVM classifier, for a fair comparison. To pre-train the CNN and the ConvLSTM models for multi-class user classification, we used a different set of users than the set used for few-shot user identification, ensuring a realistic scenario. The empirical results demonstrate that (*i*) our system attains a top accuracy of 96.72% with a FAR of 3.10% and a FRR of 3.45%, using only 20 samples per user during training, and (*ii*) our system is significantly better than the considered baselines. We thus conclude that our SVM model based on pre-trained CNN features is suitable for practical usage, having a high accuracy rate while requiring only 20 taps from the user during registration.

We note that our solution can be used as an implicit additional authentication factor during an explicit authentication, e.g. based on face recognition. For example, our system can be used in an iPhone banking application that uses FaceID. When the user enters his credentials inside the application, we can record the motion signals until the user taps on the log in button. Then, the app will delegate authentication to FaceID and, in the same time, it will analyze the motion sensors to identify the user. The user is not required to perform any additional steps during authentication, besides scanning his face for FaceID. If FaceID decides to verify the user and our system rejects the user, then perhaps an attacker might have forced the user to login into his bank account (FaceID would not be able to detect this situation). If the attacker is the one handling the iPhone, our system will be able to prevent the user from logging in.

In future work, we aim to combine the compared models into an ensemble model that should be able to further improve the identification performance of users based on motion patterns. Here, we could explore various ensemble learning approaches. We also aim to add an attention mechanism to our CNN model, which could further improve its performance. We also consider creating and improving authentication systems by implementing our system as a passive (implicit) factor in a two-factor authentication scheme.

REFERENCES

[1] P. Andriotis, T. Tryfonas, G. Oikonomou, and C. Yildiz, "A pilot study on the security of pattern screen-lock methods and soft side channel attacks," in *Proc. 6th ACM Conf. Secur. Privacy Wireless Mobile Netw. (WiSec)*, 2013, pp. 1–6.

[2] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," in *Proc. WOOT*, 2010, pp. 1–7.

[3] Y. Xu, J. Heinly, A. M. White, F. Monrose, and J.-M. Frahm, "Seeing double: Reconstructing obscured typed input from repeated compromising reflections," in *Proc. CCS*, 2013, pp. 1063–1074.

[4] Y. Zhang, P. Xia, J. Luo, Z. Ling, B. Liu, and X. Fu, ''Fingerprint attack against touch-enabled devices,'' in *Proc. 2nd ACM Workshop Secur. Privacy Smartphones Mobile Devices (SPSM)*, 2012, pp. 57–68.

[5] L. Simon and R. Anderson, ''PIN skimmer: Inferring PINs through the camera and microphone,'' in *Proc. 3rd ACM Workshop Secur. Privacy Smartphones Mobile Devices (SPSM)*, 2013, pp. 67–78.

[6] D. Shukla, R. Kumar, A. Serwadda, and V. V. Phoha, ''Beware, your hands reveal your secrets!'' in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2014, pp. 904–917.

[7] G. Ye, Z. Tang, D. Fang, X. Chen, K. I. Kim, B. Taylor, and Z. Wang, ''Cracking Android pattern lock in five attempts,'' in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2017.

[8] C. Bo, L. Zhang, T. Jung, J. Han, X.-Y. Li, and Y. Wang, ''Continuous user identification via touch and movement behavioral biometrics,'' in *Proc. IEEE 33rd Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2014, pp. 1–8.

[9] A. Buriro, B. Crispo, F. Del Frari, J. Klardie, and K. Wrona, ''ITSME: Multi-modal and unobtrusive behavioural user authentication for smartphones,'' in *Proc. PASSWORDS*, 2015, pp. 45–61.

[10] A. Buriro, B. Crispo, and Y. Zhauniarovich, ''Please hold on: Unobtrusive user authentication using smartphone's built-in sensors,'' in *Proc. IEEE Int. Conf. Identity, Secur. Behav. Anal. (ISBA)*, Feb. 2017, pp. 1–8.

[11] A. Buriro, B. Crispo, F. Delfrari, and K. Wrona, ''Hold and sign: A novel behavioral biometrics for smartphone user authentication,'' in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2016, pp. 276–285.

[12] A. Buriro, B. Crispo, S. Gupta, and F. Del Frari, ''DIALERAUTH: A motion-assisted touch-based smartphone user authentication scheme,'' in *Proc. CODASPY*, 2018, pp. 267–276.

[13] G. Canfora, P. di Notte, F. Mercaldo, and C. A. Visaggio, ''A methodology for silent and continuous authentication in mobile environment,'' in *Proc. ICETE*, M. S. Obaidat, Ed. 2017, pp. 241–265.

[14] M. Ehatisham-Ul-Haq, M. Awais Azam, U. Naeem, Y. Amin, and J. Loo, ''Continuous authentication of smartphone users based on activity pattern recognition using passive mobile sensing,'' *J. Netw. Comput. Appl.*, vol. 109, pp. 24–35, May 2018.

[15] Y. Ku, L. H. Park, S. Shin, and T. Kwon, ''Draw it as shown: Behavioral pattern lock for mobile user authentication,'' *IEEE Access*, vol. 7, pp. 69363–69378, 2019.

[16] H. Li, J. Yu, and Q. Cao, ''Intelligent walk authentication: Implicit authentication when you walk with smartphone,'' in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Dec. 2018, pp. 1113–1116.

[17] N. Neverova, C. Wolf, G. Lacey, L. Fridman, D. Chandra, B. Barbello, and G. Taylor, ''Learning human identity from motion patterns,'' *IEEE Access*, vol. 4, pp. 1810–1820, 2016.

[18] C. Shen, T. Yu, S. Yuan, Y. Li, and X. Guan, ''Performance analysis of motion-sensor behavior for user authentication on smartphones,'' *Sensors*, vol. 16, no. 3, p. 345, 2016.

[19] W. Shi, J. Yang, Y. Jiang, F. Yang, and Y. Xiong, ''SenGuard: Passive user identification on smartphones using multiple sensors,'' in *Proc. IEEE 7th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2011, pp. 141–148.

[20] Z. Sitova, J. Sedenka, Q. Yang, G. Peng, G. Zhou, P. Gasti, and K. S. Balagani, ''HMOG: New behavioral biometric features for continuous authentication of smartphone users,'' *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 5, pp. 877–892, May 2016.

[21] L. Sun, Y. Wang, B. Cao, S. Y. Philip, W. Srisa-An, and A. D. Leow, ''Sequential keystroke behavioral biometrics for mobile user identification via multi-view deep learning,'' in *Proc. ECML-PKDD*, 2017, pp. 228–240.

[22] R. Wang and D. Tao, ''Context-aware implicit authentication of smartphone users based on multi-sensor behavior,'' *IEEE Access*, vol. 7, pp. 119654–119667, 2019.

[23] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, ''Gradient-based learning applied to document recognition,'' *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ''ImageNet classification with deep convolutional neural networks,'' in *Proc. NIPS*, 2012, pp. 1097–1105.

[25] A. Ralston, ''De Bruijn sequences—A model example of the interaction of discrete mathematics and computer science,'' *Math. Mag.*, vol. 55, no. 3, pp. 131–143, 1982.

[26] C. Cortes and V. Vapnik, ''Support-vector networks,'' *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[27] T. G. Dietterich, ''Approximate statistical tests for comparing supervised classification learning algorithms,'' *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, Oct. 1998.

[28] N. L. Clarke and S. M. Furnell, ''Advanced user authentication for mobile devices,'' *Comput. Secur.*, vol. 26, no. 2, pp. 109–119, Mar. 2007.

[29] P. Campisi, E. Maiorana, M. Lo Bosco, and A. Neri, ''User authentication using keystroke dynamics for cellular phones,'' *IET Signal Process.*, vol. 3, no. 4, pp. 333–341, 2009.

[30] E. Maiorana, P. Campisi, N. González-Carballo, and A. Neri, ''Keystroke dynamics authentication for mobile phones,'' in *Proc. ACM Symp. Appl. Comput. (SAC)*, 2011, pp. 21–26.

[31] E. Vildjiounaite, S.-M. Mäkelä, M. Lindholm, R. Riihimäki, V. Kyllönen, J. Mäntyjärvi, and H. Ailisto, ''Unobtrusive multimodal biometrics for ensuring privacy and information security with personal devices,'' in *Proc. Pervas.*, 2006, pp. 187–201.

[32] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, ''Going deeper with convolutions,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[33] K. He, X. Zhang, S. Ren, and J. Sun, ''Deep residual learning for image recognition,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[34] K. Simonyan and A. Zisserman, ''Very deep convolutional networks for large-scale image recognition,'' in *Proc. ICLR*, 2014, pp. 1–14.

[35] M.-I. Georgescu, R. T. Ionescu, and M. Popescu, ''Local learning with deep and handcrafted features for facial expression recognition,'' *IEEE Access*, vol. 7, pp. 64827–64836, 2019.

[36] S. Ren, K. He, R. Girshick, and J. Sun, ''Faster R-CNN: Towards real-time object detection with region proposal networks,'' in *Proc. NIPS*, 2015, pp. 91–99.

[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ''You only look once: Unified, real-time object detection,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[38] R. T. Ionescu, B. Alexe, M. Leordeanu, M. Popescu, D. P. Papadopoulos, and V. Ferrari, ''How hard can it be? Estimating the difficulty of visual search in an image,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2157–2166.

[39] R. K. Samala, H.-P. Chan, L. Hadjiiski, M. A. Helvie, J. Wei, and K. Cha, ''Mass detection in digital breast tomosynthesis: Deep convolutional neural network with transfer learning from mammography,'' *Med. Phys.*, vol. 43, no. 12, pp. 6654–6666, Nov. 2016.

[40] N. Wahab, A. Khan, and Y. S. Lee, ''Transfer learning based deep CNN for segmentation and detection of mitoses in breast cancer histopathological images,'' *Microscopy*, vol. 68, no. 3, pp. 216–233, Jun. 2019.

[41] Y. LeCun, Y. Bengio, and G. Hinton, ''Deep learning,'' *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[42] I. Goodfellow, A. Courville, and Y. Bengio, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[43] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, ''How transferable are features in deep neural networks?'' in *Proc. NIPS*, 2014, pp. 3320–3328.

[44] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, ''ImageNet large scale visual recognition challenge,'' *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[45] V. Nair and G. E. Hinton, ''Rectified linear units improve restricted Boltzmann machines,'' in *Proc. ICML*, 2010, pp. 807–814.

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, ''Dropout: A simple way to prevent neural networks from overfitting,'' *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jan. 2014.

[47] D. P. Kingma and J. Ba, ''Adam: A method for stochastic optimization,'' in *Proc. ICLR*, 2015, pp. 1–15.

[48] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[49] S. Hochreiter and J. J. Schmidhuber, ''Long short-term memory,'' *Neural Comput.*, vol. 9, no. 8, pp. 80–1735, 1997.

[50] S. Jastrzebski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey, ''Width of minima reached by stochastic gradient descent is influenced by learning rate to batch size ratio,'' in *Proc. ICANN*, vol. 11141, 2018, pp. 392–402.

[51] M. Abadi *et al.*, ''TensorFlow: A system for large-scale machine learning,'' in *Proc. OSDI*, 2016, pp. 265–283.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. Dubourg, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[53] G. V. Trunk, "A problem of dimensionality: A simple example," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, no. 3, pp. 306–307, Jul. 1979.
[54] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS*, 2017, pp. 5998–6008.

CEZARA BENEGUI received the degree and M.Sc. Diploma degree from the Faculty of Mathematics and Computer Science, University of Bucharest, Romania, in 2016 and 2018, respectively, where she is currently pursuing the Ph.D. degree. She is a Teaching Assistant with the University of Bucharest. Her research interests include machine learning, security, user behavior, artificial intelligence, and deep learning.

RADU TUDOR IONESCU (Member, IEEE) received the Ph.D. degree from the University of Bucharest, Romania, in 2013. He is currently a Professor with the University of Bucharest. He has published over 70 articles at international peer-reviewed venues (e.g., CVPR, ICCV, ACL, EMNLP, NAACL, and WACV), and a research monograph with Springer. His research interests include machine learning, computer vision, image processing, text mining, and computational biology. He received the 2014 Award for Outstanding Doctoral Research in the field of computer science from the Romanian Ad Astra Association. He received the Caianiello Best Young Paper Award at ICIAP 2013 for the article Kernels for Visual Words Histograms. He also received the Young Researchers in Science and Engineering Prize organized by Prof. R. Mihalcea for young Romanian researchers in all scientific fields. He participated at several international competitions obtaining top ranks: fourth place in the Facial Expression Recognition Challenge of the WREPL Workshop of ICML 2013, third place in the Native Language Identification Shared Task of the BEA-8 Workshop of NAACL 2013, second place in the Arabic Dialect Identification Shared Task of the VarDial Workshop of COLING 2016, first place in the Arabic Dialect Identification Shared Task of the VarDial Workshop of EACL 2017, first place in the Native Language Identification Shared Task of the BEA-12 Workshop of EMNLP 2017, and first place in the Arabic Dialect Identification Shared Task of the VarDial Workshop of COLING 2018.