IDNet: Smartphone-based Gait Recognition with Convolutional Neural Networks

Matteo Gadaleta, Student Member, IEEE, and Michele Rossi, Senior Member, IEEE

Abstract—Here, we present IDNet, an original user authentication framework from smartphone-acquired motion signals. Its goal is to recognize a target user from her/his way of walking, using the accelerometer and gyroscope (inertial) signals provided by a commercial smartphone worn in the front pocket of the user’s trousers. Our design features several innovations including: a robust and smartphone-orientation-independent walking cycle extraction block, a novel feature extractor based on convolutional neural networks, a one-class support vector machine to classify walking cycles, and the coherent integration of these into a multi-stage authentication system. Our system exploits convolutional neural networks as universal feature extractors for gait recognition, and uses classification results from subsequent recognition performance.

Experimental results show the superiority of our approach against state-of-the-art techniques, leading to misclassification rates (either false negatives or positives) smaller than 0.15% in fewer than five walking cycles. Design choices are discussed and motivated throughout, assessing their impact on the authentication performance.

Index Terms—Biometric gait analysis, smartphone inertial sensors, authentication systems, convolutional neural networks, support vector machines, feature extraction, signal processing, accelerometer, gyroscope.

I. INTRODUCTION

Wearable technology is advancing at a very fast pace. Many wearable devices, such as smart watches and wristbands are currently available in the consumer market and they often possess miniaturized inertial motion sensors (accelerometer and gyroscope) as well as other sensing hardware capable of gathering biological signals such as photoplethysmographic signals, skin temperature and so forth. Other wearables, such as the Zephyr’s Bioharness chestband [1], deliver a number of physiological signals via their wireless interfaces (e.g., Bluetooth low energy), including electrocardiogram, heart rate, respiration rate, body orientation and activity level. The same holds true for recent smartphones, that also possess unprecedented sensing capabilities, including motion sensors but also allowing for the collection of user’s feedback and for the realtime assessment of their health condition. A notable example is provided by Apple’s HealthKit, which turns mobile phones into personalized health data hubs. So, the sensing technology is already available, most wearables already have it and, thanks to smartphones’ pervasiveness and connectivity, in the near future these devices may give rise to the first worldwide human sensor network. This is already happening, as testified by, e.g., the mPower Parkinson’s disease study from Sage Bionetworks, a nonprofit biomedical research organization that in March 2016 released a huge dataset capturing the everyday experiences of thousands of people (featuring millions of data points), to help advance scientific progress toward effective medical treatments [2]. This data was collected through a mobile application designed for iPhones and allowed a revolutionary measurement campaign.

Two major problems are now related to the analysis of wearable data and to the authentication of the mobile users who provide it, so that we can assess with reasonably high accuracy whether the data sources are genuine. Note that certifying the data sources is a necessary step toward the widespread use of this technology in the medical field.

In this paper, we propose IDNet (IDentification Network), a new system for the authentication of mobile users from their smartphone-acquired motion data. In fact, as shown in [3], [4], modern phones possess highly accurate inertial sensors, which allow for non-obtrusive gait biometrics. IDNet features Convolutional Neural Networks (CNN) [5] and tools from machine learning, such as Support Vector Machines (SVM) [6], combining them in an innovative fashion. Specifically, we develop algorithms for i) walking cycle extraction, ii) feature identification and, finally, iii) user authentication. CNNs are used as universal feature extractors to discriminate gait signatures from different subjects. Single- as well as multi-step classifiers are finally combined with CNNs to authenticate the user from multiple walking cycles. As we show shortly, our solution authenticates the target user with high accuracy and outperforms state-of-the-art techniques such as [7]–[12].

The main contributions of this paper are:

• The design and validation of an original preprocessing techniques that includes: a robust algorithm for the extraction of walking cycles and an original transformation to move smartphone acquired motion signals into an orientation invariant reference system. Subsequent processing is carried out within this reference system, as this considerably improves authentication results, see Section III. As opposed to making motion data orientation independent, previous papers either use data acquired from a sensor in a known and fixed position [7], [8], [10], [11], [13]–[16], or use orientation independent features at the cost of losing information about the direction of the forces [17].

• The design of a new CNN-based feature extraction tool, which is trained only once on a representative set of users and then used at runtime as a universal feature extractor, see Section IV. Note that with CNNs, statistical features are automatically extracted as part of the CNN training phase (automatic feature engineering) as opposed to the
selection of predefined and often arbitrary features, as commonly done in the literature [8]–[10], [13].

- The combination of CNN-extracted features with a one-class SVM classifier [19], which is solely trained on the target subject, see Section V. The resulting SVM scores are then accumulated across multiple walking cycles to get higher accuracies, through a new multi-stage decision framework, see Section VI.

- The coherent integration of these techniques into the IDNet authentication framework, that uses smartphone-acquired accelerometer and gyroscope motion data (previous algorithms solely used the accelerometer signal).

- The experimental validation of IDNet, proving its superiority against state-of-the-art solutions, see Section V and achieving authentication errors below 0.15% using fewer than five walking cycles, see Section VI.

II. RELATED WORK

Interest in gait analysis began in the 60’s, when walking patterns from healthy people, termed as normal patterns, were investigated by Murray et al. [19]. These measurements were performed through the analysis of photos acquired using interrupted light photography. Murray compared normal gait parameters against those from pathologic gaits [20] and showed that gait is unique to each individual. Since then, many studies followed and human identification systems based on gait recognition have been enjoying a growing interest.

Although most of these systems are based on computer vision [21], our interest in this paper is in human gait identification through smartphone inertial sensors. Allisto et al. [22] were the first to look at this problem and they did it through accelerometer data. In their paper, they used a triaxial accelerometer worn on a belt with fixed orientation: the x-axis pointed forward, the y-axis to the left and the z-axis was aligned with the direction of gravity. Only data points from the x and z axes were used for identification purposes. Gait cycle extraction was performed through a simple peak detection method, and a template was built for each subject. User identification employed a template matching technique, for which different methods were explored: temporal correlation, frequency domain analysis and data distribution statistics.

In [23], Derawi et al. improved the detection system through more robust preprocessing, cycle detection and template comparison algorithms. Data were acquired using a mobile phone worn on the hip, and only the vertical z-axis was considered for their motion analysis. Dynamic Time Warping (DTW) [24] was used as the distance measure, to assure robustness against non-linear temporal shifts. This scheme was also tested in [15], where majority voting and cyclic rotation were compared as inference rules. In a further paper [16], Hidden Markov Models (HMM) were explored. Accelerometer data were split into windows of fixed length, which were then utilized to train HMMs. Good identification results were obtained, but at the cost of long authentication phases (longer than 30 seconds).

Classification algorithms based on machine learning were also investigated. Either gait cycles extraction [25] or fixed windows lengths [8] are possible signal segmentation methods. After that, a feature extraction technique is applied to each segment and statistical measures such as mean, standard deviation, root mean square, zero-crossing rate or histogram bin counts, are commonly used. However, more advanced features are required for better results, like cepstral coefficients, which are widely used in speech recognition systems [8], or features extracted through frequency analysis, i.e., using Fourier [7] or wavelet transforms [25]. Supervised algorithms are typically used for classification, including k-Nearest Neighbours (k-NN) [8], [10], [12], [13], Support Vector Machines (SVM) [9], [13], [25], Multi Layer Perceptrons (MLP) [9], [13] and Classification Trees (CT) [9], [13].

We stress that in most of the related work the acquisition system was placed according to a controlled and well known orientation on the subject body. In real scenarios, this is however unlikely to occur. It is thus important to implement an algorithm whose performance is invariant to the smartphone orientation. In the mPower parkinson’s measurement campaign, for example, participants were required to put their mobile phone in the right front pocket of their trousers, walk 20 steps in a straight line, turn around, stand still for 30 seconds and walk 20 steps back. The phone orientation in the pocket is somewhat unconstrained (and unknown) and the phone reference system is with good probability misaligned with respect to the direction of motion, which makes the definition of walking templates impossible. To deal with this, two different approaches can be used. The first consists in the extraction of features that are rotation invariant (e.g., correlation matrices of Fourier transforms) [17]. The second relies on the transformation of inertial signals [9], projecting them into a new orientation invariant three-dimensional reference system, which is extracted directly from the data. Here, we adopt the latter approach.

Accelerometer based gait analysis has also interest in the medical field. Using time-frequency analysis, Huang et al. showed that signals acquired by a waist-worn device on a patient with cervical disc herniation differed before and after the surgery [14]. In [13], classification algorithms were used to discriminate a group of subjects with non-specific chronic low back pain from healthy subjects. Complex parameters, e.g., dynamic symmetry and cyclic stability of gait, were extracted by Jiang et al. [26]. However, their evaluation requires to place sensors on the legs, and fine gait details are difficult to extract from signals acquired by a single waist-worn sensor, such as a smartphone.

III. SIGNAL PROCESSING FRAMEWORK

The aim of IDNet is to correctly recognize a subject from his/her way of walking, through the acquisition of inertial signals from a standard smartphone. The proposed processing workflow is shown in Fig. 1. Walking data is first acquired, then we perform some preprocessing entailing:

1) pre-filtering to remove motion artifacts (Section III-A),
2) the extraction of walking cycles (Section III-B),
3) a transformation to move the raw walking data into a new orientation independent reference system (Section III-C).
4) a normalization to represent each walking cycle (accelerometer and gyroscope data) through fixed length, zero mean and unit variance vectors (Section III-D).

After this, the walking cycles are ready to be processed to identify the user. The standard approach, called “Classical Machine Learning” entails the computation of a number of pre-established statistical features, the most informative of which are selected and used to train a classifier. Various machine learning techniques are usually exploited to this purpose, and are trained through a supervised approach. Hence, the classification performance is assessed and the whole process is usually iterated for a further feature selection phase. In this way, the features that are used for the classification task are progressively refined until a final feature set is attained. Note that statistical features are often assessed by the designer through educated guesses and a trial and error approach.

As opposed to this, with IDNet we advocate the use of convolutional neural networks (see Sections IV and V). These have been successfully used by the video processing community [27] but to the best of our knowledge have never been exploited for the analysis of inertial data from wearable devices. One of the main advantages of this approach is that statistical features are automatically assessed by the CNN as a result of a supervised training phase. In Section IV the CNN is trained to act as a universal feature extractor, whereas in Section V a one-class SVM is trained as the final classifier. Once the CNN is trained, our system operates assuming that the smartphone only has access to the walking patterns of the target user (i.e., the legitimate user) and the SVM is solely trained using his/her walking data. Our system is based on the premise that, at runtime, the CNN should be capable of identifying the user. The standard approach, called “Classical Machine Learning” entails the computation of a number of pre-established statistical features, the most informative of which are selected and used to train a classifier. Various machine learning techniques are usually exploited to this purpose, and are trained through a supervised approach. Hence, the classification performance is assessed and the whole process is usually iterated for a further feature selection phase. In this way, the features that are used for the classification task are progressively refined until a final feature set is attained. Note that statistical features are often assessed by the designer through educated guesses and a trial and error approach.

A. Data Acquisition and Filtering

Data were acquired from 50 subjects, during a period of six months through Android smartphones worn in the right front pocket of the users’ trousers. The following devices were used: Asus Zenfone 2, Samsung S3 Neo, Samsung S4, LG G2, LG G4 and a Google Nexus 5. Several acquisition sessions of about five minutes were performed for each subject, in variable conditions, e.g., with different shoes and clothes. We asked each subject to walk as she/he felt comfortable with, to mimic real world scenarios. For the data acquisition, we developed an Android inertial data logger application, which saves accelerometer, gyroscope and magnetometer signals into non-volatile memory and then automatically transfers them to an Internet server for further processing. The magnetometer signal is not used for identification purposes. In general, IDNet can be used carrying the device in other positions but we remark that each requires a dedicated training.

In Fig. 2 we plot the power of accelerometer and gyroscope signals at different frequencies through the Welch’s method [28], considering a full walking trace and setting the Hanning window length to 1 s, with half window overlap. Most of the signal power is located at low frequencies, mostly below 40 Hz (where the power is at least 30 dB smaller than the maximum). The raw inertial signals were acquired using an average sample frequency ranging between 100 and 200 Hz (depending on the smartphone model), which is more than appropriate to capture most of the walking signal’s energy.

At the first block of IDNet processing chain, due to the non-uniform sampling performed by the smartphone operating system, we apply a cubic Spline interpolation to represent the input data through evenly spaced points (200 points/second). Hence, a low pass Finite Impulse Response (FIR) filter with a cutoff frequency of $f_{c1} = 40$ Hz is used for denoising and to reduce the motion artifacts that may appear at higher frequencies. In fact, given the power profile of Fig. 2 the
at the original signal \(a_{mag}(i)\) in a neighborhood of \(\hat{i}\). This minimum is then refined by looking at the original signal \(a_{mag}(i)\) in a neighborhood of \(\hat{i}\) and picking the minimum value of \(a_{mag}(i)\) in that neighborhood. This identifies a new index \(i^*\) for which \(a_{mag}(i^*)\) is a local minimum. As an example, in Fig. 3 we show this minimum through a red vertical (dashed-dotted) line. As a second step, we pick a window of one second centered in \(i^*\), which in Fig. 3 is represented through two vertical blue (dashed) lines. Now, the samples of \(a_{mag}(i)\) falling between the two blue lines define the first gait template, which we call \(T\), with \(|T| = N_s\) samples, where \(N_s\) corresponds to the number of samples measured in one second.

The extracted template is then iteratively refined and, at the same time, used to identify subsequent walking cycles. To this end, we first define the following correlation distance, for any two real vectors \(u\) and \(v\) of the same size \(n\) we have:

\[
\text{corr}_{\text{dist}}(u,v) = 1 - \frac{(u - \overline{u}_n) \cdot (v - \overline{v}_n)}{\|u - \overline{u}_n\| \|v - \overline{v}_n\|}. \tag{2}
\]

The template \(T\) is then processed with the acceleration magnitude through the following Eq. (3), leading to a further metric \(\varphi(i)\), where \(i = 1, 2, \ldots\) is the sample index:

\[
v_{\text{rect}}(a_{mag}(i)) = (a_{mag}(i), \ldots, a_{mag}(i + N_s - 1))^T \tag{3}
\]

\[
\varphi(i) = \text{corr}_{\text{dist}}(T, v_{\text{rect}}(a_{mag}(i))), \, i = 1, \ldots.
\]

As can be seen from Fig. 4, the function \(\varphi(i)\) exhibits some local minima, which are promptly located by comparing \(\varphi(i)\) with a suitable threshold \(\varphi_{th}\), and performing a fast search inside the regions where \(\varphi(i) < \varphi_{th}\). The indices corresponding to these minima determine the optimal alignments between the template \(T\) and \(a_{mag}(i)\). In particular, the second of these
identifies the beginning of the second gait cycle. From these facts we have that:

1) the samples between the second and the third minima correspond to the second gait cycle. It is thus possible to locate accelerometer and gyroscope vectors for this walking cycle, which are respectively defined as: \( \mathbf{a}_x, \mathbf{a}_y, \mathbf{a}_z \), and \( \mathbf{g}_x, \mathbf{g}_y, \mathbf{g}_z \), still expressed in the \((x, y, z)\) coordinate system of the smartphone. We remark that the number of samples does not necessarily match the template length and usually differs from cycle to cycle, as it depends on the length and duration of walking steps.

2) A second template \( \mathbf{T}' \) is obtained by reading \( N_s \) samples starting from the second minimum.

At this point, a new template is obtained through a weighted average of the old template \( \mathbf{T} \) and the new one \( \mathbf{T}' \):

\[
\mathbf{T} = \alpha \mathbf{T} + (1 - \alpha) \mathbf{T}',
\]

where for the results in this paper we used \( \alpha = 0.9 \). The new template \( \mathbf{T} \) is then considered for the extraction of the next walking cycle and the procedure is iterated. Note that this technique makes it possible to obtain an increasingly robust template at each new cycle.

A template matching approach exploiting a similar rationale was used in [11], where the authors employed the Pearson product-moment correlation coefficient between template and \( a_{\text{mag}}(i) \). The main differences between [11] and our approach are: we obtain the template \( \mathbf{T} \) in a neighborhood of \( i^* \), using a fixed number of samples \( N_s \), whereas they take the samples between two adjacent minima of \( \varphi(i) \) (which may then differ in size for different cycles). In Eq. (4), a discrete-time filter is utilized to refine the template \( \mathbf{T} \) at each walking cycle, making it more robust against speed changes. In previous work [11], the template is instead kept unchanged up to a point when minima cannot be longer detected, and a new template is to be obtained.

Finally, a normalization phase is required to represent all the cycles through the same number of points \( N \), as this is required by the following feature extraction and classification algorithms. Before doing this, however, a transformation of accelerometer and gyroscope signals is performed to express these inertial signals in a rotation invariant reference system, as described next.

### C. Orientation Independent Transformation

To evaluate the new orientation invariant coordinate system, three orthogonal versors \( \mathbf{\xi}, \mathbf{\zeta}, \mathbf{\psi} \) are to be found, whose orientation is independent of that of the smartphone and aligned with gravity and the direction of motion. Specifically, our aim is to express accelerometer and gyroscope signals in a coordinate system that remains fixed during the walk, with versor \( \mathbf{\zeta} \) pointing up (and parallel to the user’s torso), versor \( \mathbf{\xi} \) pointing forward (aligned with the direction of motion) and \( \mathbf{\psi} \) tracking the lateral movement and being orthogonal to the other two. This entails inferring the orientation of the mobile device carried in the front pocket from the acceleration signal acquired during the walk. To this end, we adopt a technique similar to those of [30], [31].

Gravity is the main low frequency component in the accelerometer data, and will be our starting point for the transform. Moreover, although it is a constant vector, it continuously changes when represented in the \((x, y, z)\) coordinate system of the smartphone, due to the user’s mobility and the subsequent change of orientation of the device. So, even the gravity vector \( \mathbf{\rho} \) is not constant when expressed through the smartphone coordinates. As the first axis of the new reference system, we consider the mean direction of gravity within the current walking cycle. Let \( n_k \) be the number of samples in the current cycle \( k \), with \( k = 1, 2, \ldots \). We recall that, with \( \mathbf{a}_x, \mathbf{a}_y \) and \( \mathbf{a}_z \) we mean the acceleration samples in the current cycle \( k \) along the three axes \( x, y \) and \( z \), whereas with \( \mathbf{g}_x, \mathbf{g}_y \) and \( \mathbf{g}_z \) we indicate the gyroscope samples in the same cycle \( k \), with \( |\mathbf{g}_x| = |\mathbf{g}_y| = |\mathbf{g}_z| = n_k \). The gravity vector \( \mathbf{\rho}_k \) within cycle \( k \) is estimated as:

\[
\mathbf{\rho}_k = (\mathbf{\sigma}_x, \mathbf{\sigma}_y, \mathbf{\sigma}_z)^T.
\]

The first versor of the new system \( \mathbf{\zeta} \) is obtained as:

\[
\mathbf{\zeta} = \frac{\mathbf{\rho}_k}{\| \mathbf{\rho}_k \|}.
\]

Now, we define the acceleration matrix \( \mathbf{A} = [\mathbf{a}_x, \mathbf{a}_y, \mathbf{a}_z]^T \) of size \( 3 \times n_k \), whose rows corresponds to \( \mathbf{a}_x, \mathbf{a}_y \) and \( \mathbf{a}_z \). Likewise, the gyroscope matrix is \( \mathbf{G} = [\mathbf{g}_x, \mathbf{g}_y, \mathbf{g}_z]^T \), whose rows corresponds to \( \mathbf{g}_x, \mathbf{g}_y \) and \( \mathbf{g}_z \). The projected acceleration and gyroscope vectors along axis \( \mathbf{\zeta} \) are:

\[
\mathbf{a}_\zeta = \mathbf{A} \cdot \mathbf{\zeta}, \quad \mathbf{g}_\zeta = \mathbf{G} \cdot \mathbf{\zeta},
\]

where the new vectors have the same size \( n_k \). By removing this component from the original accelerometer signal, we project the latter on a plane that is orthogonal to \( \mathbf{\zeta} \). This is the horizontal plane (parallel to the floor). We represent this flattened acceleration data through a new matrix \( \mathbf{A}' = [\mathbf{a}'_x, \mathbf{a}'_y, \mathbf{a}'_z]^T \) of size \( 3 \times n_k \), where \( \mathbf{a}'_x, \mathbf{a}'_y \) and \( \mathbf{a}'_z \) are vectors of size \( n_k \) that describe the acceleration on the new plane:

\[
\mathbf{A}' = \mathbf{A} - \mathbf{\zeta} \mathbf{a}_\zeta^T.
\]

Analyzing this flattened acceleration, we see that during a walking cycle it is unevenly distributed on the horizontal plane. Also, the acceleration points on this plane are dispursed around a preferential direction, which has the highest excursin (variance). Here, we assume that the direction with the largest variance in our measurement space contain the dynamics of interest, i.e., it is parallel to the direction of motion, as it was also observed and verified in previous research [30]. Given this, we pick this direction as the second axis (versor \( \mathbf{\xi} \)) of the new reference system. This is done by applying the Principal Component Analysis (PCA) [32] on the projected points, which finds the direction along which the variance of the measurements is maximized. The PCA procedure is as follows:

1) Find the empirical mean along each direction \( x, y \) and \( z \) (rows 1, 2 and 3 of the flattened acceleration matrix \( \mathbf{A}' \)). Store the mean in a new vector \( \mathbf{u} \) of size \( 3 \times 1 \), i.e.:

\[
u_i = \frac{1}{n_k} \sum_{j=1}^{n_k} A'_{i,j}, \quad i = 1, 2, 3.
\]
An example of this transform is shown in Fig. 5, where accelerometer data in the smartphone reference system (left), and after the transformation (ξ, ψ, ζ) (right). IDNet implements a PCA-based transformation that makes walking data rotation invariant, i.e., subject-specific gait patterns emerge in the new coordinate system (see the red colored patterns in the right plots).

2) Subtract the empirical mean vector \( \mathbf{u} \) from each column of matrix \( \mathbf{A} \), obtaining the new matrix \( \mathbf{A}_{\text{norm}} \):

\[
\mathbf{A}_{\text{norm}} = \mathbf{A} - \mathbf{u}(\mathbf{1}_{N \times 1})^T.
\]  

3) Compute the sample 3 × 3 autocovariance matrix \( \mathbf{\Sigma} \):

\[
\mathbf{\Sigma} = \frac{\mathbf{A}_{\text{norm}}(\mathbf{A}_{\text{norm}})^T}{N - 1}.
\]  

4) The eigenvalues and the corresponding eigenvectors of \( \mathbf{\Sigma} \) are evaluated. The eigenvector \( \mathbf{v} \) associated with the maximum eigenvalue identifies the direction of maximum variance in the dataset (i.e., the first principal component of the PCA transform).

Hence, versor \( \hat{\xi} \) is evaluated as:

\[
\hat{\xi} = \frac{\mathbf{v}}{||\mathbf{v}||}.
\]  

Accelerometer and gyroscope data are then projected along \( \hat{\xi} \) through the following equations: \( \mathbf{a}_\xi = \mathbf{A} \cdot \hat{\xi} \) and \( \mathbf{g}_\xi = \mathbf{G} \cdot \hat{\xi} \). Being \( \xi \) placed on a plane that is orthogonal to \( \xi \), these two versors are also orthogonal. The third axis is then obtained through a cross product:

\[
\hat{\psi} = \hat{\xi} \times \hat{\zeta},
\]  

and the new accelerometer and gyroscope data along this axis are respectively obtained as: \( \mathbf{a}_\psi = \mathbf{A} \cdot \hat{\psi} \) and \( \mathbf{g}_\psi = \mathbf{G} \cdot \hat{\psi} \). The transformed vectors \( \mathbf{a}_\xi, \mathbf{a}_\psi, \mathbf{a}_\zeta \) and \( \mathbf{g}_\xi, \mathbf{g}_\psi, \mathbf{g}_\zeta \), along with the magnitude vectors \( \mathbf{a}_{\text{mag}}, \mathbf{g}_{\text{mag}} \) are the output of the Orientation Independent Transformation block of Fig. II.

In this section, we present the chosen Convolutional Neural Network (CNN) architecture for IDNet (Section IV-A), its optimization, training and quantitative comparison against the most common classifiers from the literature (Section IV-B).
weights, which are convolved with the input in a way that the same set of weights, i.e., the same kernel, is applied to all the input data, moving the convolution operation across the input span. Note that, as the same weights are reused (shared weights), and each kernel operates on a small portion of the input signal, it follows that the network connectivity structure is sparse. This leads to advantages such as a considerably reduced computational complexity with respect to fully connected feed forward neural networks. For more details the reader is referred to [33]. CNNs have been proven to be excellent feature extractors for images [34] and here we prove their effectiveness for motion data. The CNN architecture that we designed to this purpose is shown in Fig. 6. It is composed of a cascade of two convolutional layers, followed by a pooling and a fully-connected layer. The convolutional layers perform a dimensionality reduction (or feature extraction) task, whereas the fully-connected one acts as a classifier. Accelerometer and gyroscope data from each walking cycle are processed according to the algorithms of Section III. We refer to the input matrix for a generic walking cycle to capture any correlation among different accelerometer and gyroscope axes. The activation functions are linear and the output activation functions are non-linear hyperbolic tangents. Max pooling is applied to the output of CL2 to further reduce its dimensionality and increase the spatial invariance of features [35]. With \( N_{K2} \) we mean the number of convolutional kernels used for CL2.

- **FL1** This is a fully connected layer, i.e., each output neuron of CL2 is connected to all input neurons of this layer (weights are not shared). Hyperbolic tangent activation functions are used at the output neurons. FL1 output vector is termed \( \mathbf{f} = (f_1, \ldots, f_F)^T \), and contains the \( F \) features extracted by the CNN.

- **FL2** Each output neuron in this layer corresponds to a specific class (one class per user), for a total of \( K \) neurons, where \( K \) is the number of subjects considered for the training phase. The \( K \) dimensional output vector \( \mathbf{y} = (y_1, \ldots, y_K)^T \) is obtained by a softmax activation function, which implies that \( y_j \in (0,1), j=1,\ldots,K \) and \( \sum_{j=1}^{K} y_j = 1 \) (stochastic vector). Also, \( y_j \) can be thought of as the probability that the current data matrix \( \mathbf{X} \) belongs to class (user) \( j \).

The network is trained in a supervised manner for a total of \( K \) subjects solving a multi-class classification problem, where each of the input matrices \( \mathbf{X} \) in the dataset is assigned to one of \( K \) mutually exclusive classes. The target output vector \( \mathbf{t} = (t_1, \ldots, t_K)^T \) has binary entries and is encoded using a 1-of-\( K \) coding scheme, i.e., they are all zero except for that corresponding to the subject that generated the input data.

### B. CNN Optimization and Results

In this section, we propose some approaches for the optimization of the CNN, quantify its classification performance and compare it against classification techniques from the...
As said above, the output of layer FL2 is the stochastic vector \( y \), whose \( j \)-th entry \( y_j \), \( j = 1, \ldots, K \), can be seen as the probability that the input pattern belongs to user \( j \), i.e., \( y_j = y_j(w, X) = \text{Prob}(t_j = 1|w, X) \), where \( w \) is the vector containing all the CNN weights, \( X \) is the current input matrix (walking cycle) and \( t_j = 1 \) if \( X \) belongs to class \( j \) and \( t_j = 0 \) otherwise. If \( \mathcal{X} \) is the set of all training examples, we define the batch set as \( B \subset \mathcal{X} \). Let \( X \in B \) and denote the corresponding output vector by \( y(w, X) \) and its \( j \)-th entry by \( y_j(w, X) \). The corresponding target vector is \( t(X) = (t_1(X), \ldots, t_K(X))^T \). The CNN is then trained through a stochastic gradient descend algorithm which minimizes a categorical cross-entropy loss function \( L(w) \), defined as [6] Eq. (5.24) of Section 5.2:

\[
L(w) = -\sum_{X \in B} \sum_{j=1}^{K} t_j(X) \log(y_j(w, X)).
\]

During training, Eq. (14) is iteratively minimized, by rotating the walking cycles (training examples) in the batch set \( B \) so as to span the entire input set \( \mathcal{X} \). Training continues until a stopping criterion is met (see below).

Walking patterns from \( K \) subjects are used to train the CNN, and the same number of cycles \( N_c \) is considered for each of them, for a total of \( KN_c \) training cycles. \( N_c \) randomly chosen walking cycles from each subject are used to obtain a test set \( \mathcal{P} \). The remaining cycles are split into training \( \mathcal{T} \) and validation \( \mathcal{V} \) sets, with \( |\mathcal{P}| = KN_c \), \( |\mathcal{T}| = K \times N_c \), \( \mathcal{X} = \mathcal{P} \cup \mathcal{T} \cup \mathcal{V} \), where all the sets have null pairwise intersection and are built picking input patterns from \( \mathcal{X} \) evenly at random. Set \( \mathcal{V} \) is used to terminate the training phase, and termination occurs when the loss function \( L(w) \) evaluated on \( \mathcal{V} \) does not decrease for twenty consecutive training epochs. After that, the network weights which led to the minimum validation loss are used to assess the CNN performance on set \( \mathcal{P} \). This is done through an accuracy measure, defined as the number of walking cycles correctly classified by the CNN divided by the total number of cycles in \( \mathcal{P} \). In the following graphs, we show the mean accuracy obtained averaging the test set performance over ten different networks, all of them trained through the just explained approach by considering \( K = 35 \) subjects from our dataset and \( N_t = 100 \) cycles per subject.

As a first set of results, we look at the impact of \( F \) (neurons in layer FL1) and of the number of convolutional kernels in CL1 and CL2. Since the last layer FL2 acts as a classifier, \( F \) can be seen as the number of features extracted by the CNN. In general, a too small \( F \) can lead to poor classification results; too many features, instead, would make the state space too big to be effectively dealt with (curse of dimensionality) [36]. Besides \( F \), we also investigate the right number of kernels to use within each convolutional layer. Three networks are considered by picking different \((N_{k1}, N_{k2})\) pairs. For network 1 we use \((N_{k1} = 10, N_{k2} = 20)\), network 2 has \((N_{k1} = 20, N_{k2} = 40)\) and network 3 has \((N_{k1} = 30, N_{k2} = 50)\). In Fig. 7 we show the accuracy performance of these networks as a function of \( F \). From this plot, it can be seen that at least \( F = 20 \) neurons have to be used at the output of FL1 and that the accuracy performance stabilizes around \( F = 40 \), leading to negligible improvements as \( N \) grows beyond this value. As for the number of kernels, we conclude that small networks (network 1) perform worse than bigger ones (networks 2 and 3), but increasing the number of kernels beyond that used for network 2 does not lead to appreciable improvements. Hence, for the results of this paper we used \( F = 40 \) with \((N_{k1} = 20, N_{k2} = 40)\).

A key performance comparison is shown in Fig. 8, where the accuracy is plotted against \( N_c \) for a CNN classifier and four selected classification algorithms from the literature, i.e., Classification Trees (CT) [37], Naive Bayes (NB) classifiers [38], \( k \)-Nearest Neighbors (\( k \)-NN) [39] and Support Vector Machines (SVM) [40]. These approaches were used in a large number of papers including [8–10, 13, 25]. For their training, 112 features were extracted from the signal samples in \( X \), including their variance, mean trend, windowed mean difference, variance trend, windowed variance difference, maxima and minima, spectral entropy, zero crossing rate and bin counts. These features, were then utilized to train the selected classifiers in a supervised manner. Note that, while the CNN automatically extracts its features (vector \( f \)), with previous techniques these are manually selected based on experience.

From Fig. 8 we see that the CNN-based classification approach surpasses all the previous classifiers from the literature, delivering better accuracies across the entire range of \( N_c \). Also, the accuracy increases with an increasing \( N_c \) until it saturates and no noticeable improvements are observed. While a higher \( N_c \) is always beneficial, a higher number of cycles also entails a longer acquisition time, which we would rather avoid. For this reason, for the following results we have used \( N_c = 40 \) as it provided a good tradeoff between accuracy and complexity across all our experiments.

To illustrate the superiority of CNN features with respect

---

1 For SVM, we considered a linear kernel, as it outperformed polynomial and radial basis function ones (results are omitted in the interest of space). A one-versus-all strategy was used solve the considered multiclass problem for the binary classifiers.
to manually extracted ones, in the following we conduct an instructive experiment. We consider CNN as a feature extraction block, by removing the output vector \( y \) and using the inner feature vector \( f \) to train the above classifiers from the literature (CT, NB, \( k \)-NN and SVM). The corresponding accuracy results are provided in Fig. 9. All the classifiers perform better when trained using CNN features, with typical improvements in the test accuracy of more than 10%. For instance, for a \( k \)-NN classifier trained with \( N_c = 30 \) cycles per subject, the accuracy increases from 71% (manually extracted features) to 94% (CNN features). The best performance is provided by the combined use of CNN features and SVM.

A last consideration is in order. Most of the previous papers only used accelerometer data, but our results show that using both gyroscope and accelerometer provides further improvements, see Fig. 10.

V. ONE-CLASS SUPPORT VECTOR MACHINE TRAINING

In this section, we further extend the IDNet CNN-based authentication chain through the design of an SVM classifier which is trained solely using the motion data of the target subject. This is referred to as One-Class Classification (OCC) and is important for practical applications where motion signals of the target user are available, but those belonging to other subjects are not. More importantly, with this approach the classification framework can be extended to users that were not considered in the CNN training.

A. Revised Classification Architecture

Due to the generalization properties of convolutional deep networks, once trained, the CNN can be used as a universal feature extractor, providing meaningful features even for subjects that were not included in the training. To take advantage of this, we discard the output neurons of FL2 and utilize the CNN as a dimensionality reduction tool that, given an input matrix \( X \), returns a user dependent feature vector \( f \). The CNN is then trained only once considering the optimizations of Section IV-B. All its weights and biases are then precomputed and will not be modified at classification time. Considering the diagram of Fig. 6 at the output of the CNN we obtain the feature vector \( f \). We then apply a feature selection block to reduce the number of features from \( F \) to \( S \leq F \) (dimensionality reduction). PCA is used to accomplish this task and the new feature vector is called \( s \). Hence, we have \( s = \Upsilon(f) \), where \( \Upsilon(\cdot) : \mathbb{R}^F \to \mathbb{R}^S \) is the PCA transform.

A One-class Support Vector Machine (OSVM) is then used as the classification algorithm (Section V-B). It defines a boundary around the feature (training) vectors belonging to the target subject. At runtime, as a new walking cycle is processed, the OSVM takes the feature vector \( s \) and outputs a score, which is a distance measure between the current feature vector and the SVM boundary [6, Chapter 7]. As we discuss shortly, this score relates to the likelihood that the current walking cycle belongs to the target user.
B. One-Class SVM Design

Next, we design the OSVM block of Fig. 6. It differs from a standard binary SVM classifier as the SVM boundary is built solely using patterns from the positive class (target user). The strategy proposed by Schölkopf is to map the data into the feature space of a kernel, and to separate them from the origin with maximum margin [41]. The corresponding minimization problem is similar to that of the original SVM formulation [40]. The trick is to use a hyperplane (in the space transformed by a suitable kernel function) to discriminate the target vectors. OSVM takes as input the reduced feature vector \( s = (s_1, \ldots, s_S)^T \) and we use the following Radial Basis Function (RBF) kernel, that for any \( s, s' \in \mathbb{R}^S \) is defined as:

\[
\Psi(s, s') = (\Phi(s) \cdot \Phi(s')) = \exp\left(-\gamma \|s - s'\|^2\right),
\]

where \( \Phi(s) \) is a feature map and \( \gamma \) is the RBF kernel parameter, which intuitively relates to the radius of influence that each training vector has for the space transformation. With \( \ell \) we mean the number of training points (feature vectors), \( \omega \) and \( b \) are the hyperplane parameters in the transformed domain (through Eq. (15)) and \( \varepsilon = (\varepsilon_1, \ldots, \varepsilon_\ell)^T \) is the vector of slack variables, which are introduced to deal with outliers. Given this, the following quadratic program is defined to separate the feature vectors in the training set, \( s_1, \ldots, s_\ell \), from the origin:

\[
\begin{align*}
\min_{\omega, \varepsilon, b} & \quad \frac{1}{2} \|\omega\|^2 + \frac{1}{\nu \ell} \sum_{j=1}^{\ell} \varepsilon_j - b \\
\text{subject to} & \quad (\omega \cdot \Phi(s_j)) \geq b - \varepsilon_j, \quad \varepsilon_j \geq 0, \quad j = 1, \ldots, \ell \\
\end{align*}
\]

\( \nu \in (0, 1) \) is one of the most important parameters and sets an upper bound on the fraction of outliers and a lower bound on the fraction of Support Vectors (SV) [41]. The decision function for a generic feature vector \( s \) is defined as \( d(s) \in \{-1, +1\} \), is obtained solving Eq. (16), and only depends on the training vectors through the following relations:

\[
d(s) = \text{sgn}(h(s)), \quad h(s) = \sum_{j=1}^{\ell} \alpha_j \Psi(s_j, s) - b. \tag{17}
\]

Now, \( \alpha_j \geq 0, \forall j \), and only some of the training vectors have \( \alpha_j > 0 \). These are the support vectors associated with the classification problem and are the only ones who count in the definition of the SVM boundary. \( h(s) \) is the score associated with vector \( s \). It weighs the distance from the SVM boundary, i.e., is greater than zero if \( s \) resides inside the boundary, zero if it lies on it and negative otherwise.

Hence, the SVM is trained using a set of \( \ell \) feature vectors from the target user, obtaining the SVM boundary (and the related decision function) through Eq. (17). After training, we test the performance of the obtained SVM classifier considering feature vectors from the positive class \( C_1 \) (target user) and the negative one \( C_0 \) (any other user). Note that the vectors used for this test were not considered during the SVM training.

As it is customary for binary classification approaches, the two most important metrics to assess the goodness of a classifier are the precision and the recall. The precision is the fraction of true positives, i.e., the fraction of patterns identified of the target class that in fact belong to the target user, while the recall corresponds to the fraction of target patterns that are correctly classified out of the entire positive class of samples [42]. Often, these two metrics are combined into their harmonic mean, which is called F-measure and is used as the single quality parameter.

In Fig. 11 the F-measure is plotted as a function of the two SVM parameters \( \gamma \) and \( \nu \). As seen from this plot, the area where the classifier’s performance is maximum is quite ample. This is good as it means that even selecting \( \gamma \) and \( \nu \) once for all at design stage, the performance of the SVM classifier is not expected to change much if the signal statistics changes or a new target user is considered. In other words, this relatively weak dependence on the parameters entails an intrinsic robustness for the classifier. For the results that follow we have used \( \gamma = 0.3 \) and \( \nu = 0.02 \).

Two last considerations are in order. The first relates to the PCA transformation \( \Upsilon(\cdot) \) and in particular to how many and which principal components have to be retained for the output feature vectors. In fact, as pointed out in [43], two options are possible to go from the CNN-extracted feature vector \( f \) to \( s \). The first is to retain the \( S \leq F \) entries of the transformed vector (expressed in the PCA basis) that correspond to the principal components with highest variance, whereas a second option is to retain those with the smallest. Fig. 12 shows the F-measure of the OSVM classifier as a function of \( S \) for \( F = 40 \) (number of CNN-extracted features). From this plot we see that picking \( S < F \) in general provides better results and also that considering the principal components with lowest variance provides better results for this class of problems. This is in accordance with [43].

The last consideration regards the amount of feature vectors belonging to the target user that should be used for the OSVM training. Note that this number is related to the walking time required for a new subject to train his/her personal authentication system. To perform this analysis, a fixed number of cycles were randomly extracted from the whole target dataset and were used to train the OSVM. The remaining walking cycles were used as the positive test set. In Fig. 13 we show the F-measure as a function of this number of cycles. From these
results, it follows that increasing the number of cycles beyond 1,000 leads to little improvement. This number corresponds to about 15 minutes of walking activity, distributed among different acquisition sessions. Multiple sessions are recommended to account for some statistical variation due to wearing different clothes.

Once all the model’s parameters are defined, the OSVM score can be analyzed. Let \( p_0(h(s)) = p(h(s) \mid s \in C_0) \) be the estimated probability density function (pdf) of the OSVM score \( h(s) \in \mathbb{R} \), provided that the walking cycle belongs to a user of class \( C_0 \) with \( \theta \in \{0, 1\} \). Empirical pdfs \( p_0(h(s)) \) obtained from our dataset are provided in Fig. 14.

### VI. Multi-Stage Authentication

The so far discussed processing pipeline returns a score for each walking cycle. However, as seen in Fig. 14 when a score falls near the point where the two pdfs intersect, there is a high uncertainty about the identity of the user who generated it. In IDNet, we resolve this indetermination by jointly considering the scores from successive walking cycles.

Let \( \mathcal{O} = (o_1, o_2, \ldots, o_{n}) \) be a sequence of subsequent OSVM scores from the same subject, where \( o_i = h(s_i) \in \mathbb{R} \) and \( i = 1, 2, \ldots \) is the walking cycle index. From our previous analysis, \( o_i \) can be thought of as a random process having probability density function \( p_0(h(s_i)) = p_0(o_i), \theta \in \{0, 1\} \), and our objective is to reliably estimate \( \theta \) from the scores in \( \mathcal{O} \). Toward this, we assume that subsequent scores belong to the same user and that they are independent and identically distributed (i.i.d), i.e., they are independently drawn from \( p_0(\cdot) \), with \( \theta \) unknown.

For the estimation of \( \theta \) we use Wald’s probability ratio test (SPRT) [44], [45]. We define the two hypotheses \( \{H_1 : \theta = 1\} \), meaning that the sequence \( \mathcal{O} \) belongs to the target user (class \( C_1 \)), and \( \{H_0 : \theta = 0\} \), meaning that another user generated it (class \( C_0 \)). Hence, we assess which one of these is true through SPRT sequential binary testing. That is, we keep measuring new scores and use them to decrease our uncertainty about \( \theta \). Considering \( n \) samples \( (o_1, o_2, \ldots, o_n) \), the final decision takes on two values \( D_n = 0 \) or \( D_n = 1 \), where \( D_n = j, j \in \{0, 1\} \) means that hypothesis \( H_j \) is accepted and therefore the alternative hypothesis is rejected.

Owing to our assumptions (i.i.d. scores, generated by the same subject), for \( n \) scores \( \mathcal{O}_n = (o_1, o_2, \ldots, o_n) \) the joint pdf is:

\[
\tilde{p}_\theta(\mathcal{O}_n) = \prod_{j=1}^{n} p_\theta(o_j), \quad \theta \in \{0, 1\}. \tag{18}
\]

Defining \( \lambda_j = p_1(o_j)/p_0(o_j) \), the likelihood ratio of the sequence \( \mathcal{O} \) truncated at index \( n \), \( \mathcal{O}_n \), is

\[
\frac{\tilde{p}_1(\mathcal{O}_n)}{\tilde{p}_0(\mathcal{O}_n)} = \prod_{j=1}^{n} \frac{p_1(o_j)}{p_0(o_j)} = \prod_{j=1}^{n} \lambda_j, \tag{19}
\]

and applying the logarithm, we get:

\[
\Lambda_n = \log \left( \frac{\tilde{p}_1(\mathcal{O}_n)}{\tilde{p}_0(\mathcal{O}_n)} \right) = \sum_{j=1}^{n} \log (\lambda_j). \tag{20}
\]

If we wait a further step \( n + 1 \) before making a decision, from Eq. (20) the new log-likelihood \( \Lambda_{n+1} \) is conveniently obtained as \( \Lambda_{n+1} = \Lambda_n + \log(\lambda_{n+1}) \). The SPRT test starts...
Experimental Results

B = log((1 − HN CNN feature extractor, with

Moreover, defining

One user out of the remaining

thresholds (i.e., the target is not recognized) are

rates (i.e., a user is mistakenly authenticated as the target)

were obtained in previous papers lead to error rates ranging

from time 1, obtaining one-class OSVM scores \( o_1, o_2, \ldots \) for each successive walking cycle. After \( n \) cycles, the cumulative

log-likelihood ratio is \( \Lambda_n = \Lambda_{n-1} + \log(\lambda_n) \), with \( \Lambda_0 = 0 \). Two suitable thresholds \( A \) and \( B \) are defined and the test continues to the next cycle \( n + 1 \) if \( A < \Lambda_n < B \), \( H_1 \) is accepted if \( \Lambda_n \geq B \), whereas \( H_0 \) is accepted if \( \Lambda_n \leq A \). Moreover, defining \( \alpha \) as the probability of accepting \( H_1 \) when \( H_0 \) is true and \( \beta \) that of accepting \( H_0 \) when \( H_1 \) is true, \( A \) and \( B \) can be approximated as: \( A = \log(\beta/(1−\alpha)) \) and \( B = \log((1−\beta)/\alpha) \), see [44].

A. Experimental Results

The motion data from \( K = 35 \) subjects was used to train the

CNN feature extractor, with \( N_c = 40, F = 40 \) and \( S = 20 \).

One user out of the remaining 15 was considered as the target

user and 14 as the negatives for the final tests. The following results are obtained through a leave-one-out cross-validation approach for the sessions of the target user, i.e., out of twelve sessions, eleven are used for training and one for the final tests. The session that is left out is rotated and the final results are averaged across all trials. The authentication results of the multi-stage framework are shown in Fig. [15]. False positive and negative rates are shown in the top graphs, the number of walking cycles required to make a final decision on the user’s identity is shown in the bottom ones. Upper shaded areas extend for a full standard deviation from the mean and include about 80% of the events.

As for our assumptions, in light of the small number of cycles required, it is reasonable to presume that the same subject generates the scores in \( O \). For the i.i.d. assumption, we extended the decision framework to the first-order autoregressive model of [45] Chapter 3, p. 158], which allows tracking the correlation across successive cycles. However, this did not lead to any appreciable performance improvement and only implied a higher complexity. The reason is that scores are lightly correlated in time.

VII. Conclusions

In this paper we have proposed IDNet, a user authentication framework for smartphone-acquired inertial signals. Various schemes performing manual feature extraction and using the selected features for user classification have appeared in the recent literature. In sharp contrast with these, IDNet exploits convolutional neural networks, as they allow for an automatic feature engineering and have excellent generalization capabilities. These deep neural networks are then used as universal feature extractors to feed classification techniques, combining them with one-class support vector machines and a novel multi-stage decision algorithm. With our framework, the neural network is trained once for all and subsequently utilized for new users. The one-class classifier is solely trained using motion data from the target subject; it returns a score weighing the dissimilarity of newly acquired data with respect to that of the target. Subsequent scores are then accumulated through a multi-stage decision approach.

Experimental results show the superiority of IDNet against prior work, leading to misclassification rates smaller than 0.15% in fewer than five walking cycles. Design choices and the optimization of the various processing blocks were discussed and compared against classical approaches.
