Neural Computing and Applications (2022) 34:11691–11705
https://doi.org/10.1007/s00521-022-07061-3

ORIGINAL ARTICLE

AI-based user authentication reinforcement by continuous extraction of behavioral interaction features

Daniel Garabato[1] · Carlos Dafonte[1] · Raúl Santoveña[1] · Arturo Silvelo[1] · Francisco J. Nóvoa[1] · Minia Manteiga[2]

Received: 23 November 2021 / Accepted: 5 February 2022 / Published online: 28 March 2022

Abstract
In this work, we conduct an experiment to analyze the feasibility of a continuous authentication method based on the monitorization of the users' activity to verify their identities through specific user profiles modeled via Artificial Intelligence techniques. In order to conduct the experiment, a custom application was developed to gather user records in a guided scenario where some predefined actions must be completed. This dataset has been anonymized and will be available to the community. Additionally, a public dataset was also used for benchmarking purposes so that our techniques could be validated in a non-guided scenario. Such data were processed to extract a number of key features that could be used to train three different Artificial Intelligence techniques: Support Vector Machines, Multi-Layer Perceptrons, and a Deep Learning approach. These techniques demonstrated to perform well in both scenarios, being able to authenticate users in an effective manner. Finally, a rejection test was conducted, and a continuous authentication system was proposed and tested using weighted sliding windows, so that an impostor could be detected in a real environment when a legitimate user session is hijacked.

Keywords Behavioral features · Second level authentication · Neural networks · Deep learning

1 Introduction

Computer security or cybersecurity is one of the main issues and research topics since the origins of Information Technologies (IT). Cybersecurity refers to the protection mechanisms supplied to an IT system in order to preserve integrity, confidentiality, and availability of its resources [1, 2]. Over the years, a wide variety of security measures have been developed in order to protect different types of assets, from physical devices to software services and data. Many of such measures need the combination of Artificial Intelligence (AI) techniques with real-time data processing in a Big Data environment. Our research group has demonstrated its expertise in this field, and we have proposed an intrusion detection system (IDS) based on Artificial Intelligence techniques [3, 4]. Specifically, we used unsupervised neural networks (Self Organizing Maps) to detect and isolate anomalous network events that are potentially hazardous. In the present work, we study the feasibility of a second-phase authentication system by

✉ Daniel Garabato
daniel.garabato@udc.es

Carlos Dafonte
dafonte@udc.es

Raúl Santoveña
raul.santovena@udc.es

Arturo Silvelo
arturo.silvelo@udc.es

Francisco J. Nóvoa
fjnovoa@udc.es

Minia Manteiga
manteiga@udc.es

1 CIGUS CITIC - Department of Computer Science and Information Technologies, University of A Coruña, Campus de Elviña s/n, 15071 A Coruña, Spain

2 CIGUS CITIC - Department of Nautical Sciences and Marine Engineering, University of A Coruña, Paseo de Ronda 51, 15011 A Coruña, Spain

analyzing the users' interaction with the computer, using just a common pointing device.

Authentication systems [2] are one of the most extended security mechanisms, and almost every computer system or service relies on them to check the users' identity, so that access permissions could be granted to them according to their profile. This authentication process can be defined as a two-step procedure where an entity provides an identification to the system, and the system carries out a verification to confirm the correspondence between such an identification and the entity [5]. Through this process, it is possible to prevent and detect an identity theft, granting access and privileges to a legitimate user, and denying them to a dishonest user. This process can be done by means of different factors [6], being the most common ones:

- The users ability to authenticate themselves using some data or information that, theoretically, nobody else should know, apart from the users themselves and the authentication system. This is the case of a user/password scheme, a personal identification number (PIN) or even secret question/answer models.
- The authentication system provides the user with a device, which must be used to confirm the identity. There are many suitable daily-used devices, such as an identity card, a bank card, a smartphone or even an email account or a telephone number. Additionally, there are some specific devices (i.e., smart cards) that may be issued by the authentication entity.
- This process can be also based on biometric features, inherent to the user and typically invariant over time, such as fingerprint, voice, palm veins, iris or retina.
- Novelty systems can also use some features derived from the users' behavior during its interaction with an IT system [7], such as the type of applications used, the websites visited, keystroke patterns or mouse movements [8–11].
- Additionally, some systems also rely on the location of the user; that is, they keep track of the user movements. In this case, not only geographic locations of mobile devices could be used, but also Media Access Control (MAC) or Internet Protocol (IP) address are also suitable.

During the last years, classic authentication schemes have evolved towards a multi-factor verification process [12], where user identity is proven through different complementary factors that enhance the overall system robustness and security. This progress was possible due to the popularization of smart devices, such as phones, tablets or watches. Smart devices are able to provide a confirmation code to strengthen the primary authentication factor, usually based on a user/password scheme.

Despite this progress, once the user is granted with access and privileges using these schemes, there is no way to detect any identity theft over an open session. To mitigate this issue, the user could be periodically asked for authentication. However, the usage of a common scheme for this purpose becomes unfeasible, since it would interrupt the users' sessions repeatedly. An alternative or complementary way to do this, is to use the users' behavior as a second-phase authentication oriented to the continuous monitorization of the interaction between the user and the system [13]. In this sense, common input/output devices such as keyboards or pointing devices, are the most immediate choices to test [8]. Asking again for a password or sending an SMS code to verify that the users' identity could be triggered in case the continuous monitorization detects an anomaly on the users' behavior, but it could also raise an alert to the system administrators.

Some works in the literature have studied interaction features for authentication purposes in many different ways, from various devices (such as keyboards, pointing devices or touch-screen) to specific purpose schemes (such as monitoring the interaction just during the login phase or primary authentication by means of a graphical password). Regarding authentication through mouse movements, most of these works have conducted the study in a guided scenario, in which raw movements were collected and processed to extract spatial and temporal behavioral features based on actions (e.g. distances, angles, speed, acceleration, etc.) that were used to feed different Artificial Intelligence techniques, such as statistical models [11], classic Artificial Neural Networks [7, 9, 14], Support Vector Machines [10], Decision Trees [13], or Random Forests [15]. Furthermore, over the last years more sophisticated methods based on Deep Learning have been used in order to address authentication through interaction features, mostly for mobile and wearable devices [16–18].

In this work, we explore the usability of mouse behavior oriented to a second-phase authentication mechanism, that would monitorize users' activity over their sessions in order to assess the legitimacy of such a session. Mouse dynamics have the advantage of being not intrusive and not that sensitive as keystroke dynamics, which quite frequently involve critical information, such as passwords or banking information. To this purpose, we have collected mouse interaction via an ad-hoc application in a guided environment, extracting representative characteristics and, based on them, we have built specific user profiles using different Artificial Intelligence techniques, from base methods, such as Artificial Neural Networks (ANNs) [19, 20] or Support Vector Machines (SVMs) [21], to more advanced ones, such as those based on Deep Learning (DL) [22, 23]. In addition, the same procedures were applied to the Balabit Mouse Challenge Dataset [15], a public dataset that has been studied by different authors and that we used as a reference benchmark in order to assess the performance of our authentication schemes.

The remainder of this paper is organized as follows. Section 2 defines the problem statement and the objectives we aim to accomplish. Section 3 describes the datasets used to conduct the experiments, and how data was treated and preprocessed. The procedures used to build user profiles and the method proposed to perform continuous authentication are presented in Sects. 4 and 5, whereas the results obtained during the experimentation are presented in Sect. 6. Finally, in Sect. 7 we draw some conclusions from the work carried out in this paper, and discuss possible future lines of work.

2 Problem statement and objectives

A continuous authentication scheme requires not only a non-intrusive way to monitorize the user, but also a robust set of features that allow to accurately verify the users' identities in near-real-time. Hence, we have to take into account these aspects in order to build a feasible authentication mechanism, otherwise it will not be suitable for real environments. The main goals that we pretend to fulfill in this work can be summarized as follows:

- Develop a data acquisition procedure that is completely transparent to the user, so that it does not interfere with the users' activities. Therefore, data must be gathered in an effective manner, without disturbing the user or introducing an excessive computational overhead to the system. This also means that data collection must be lightweight enough to be able to efficiently send them through the communications network.
- Identify a set of relevant or key features that can be used to unequivocally verify the users' identities through their behavior on the use of a common pointer device. Since our goal is to develop a second-phase authentication mechanism, the identity that has to be verified is already known (the legitimate owner of the active user account). Thus, the activity being monitorized during a particular session can be checked against a pre-built user profile for the true owner of the account in order to detect any session hijacking.
- The entire process, from data acquisition to identity verification must be carried out in near-real-time, so that identity usurpation can be prevented. Therefore, the methods selected, while computationally feasible, should not compromise the accuracy of the authentication process. In particular, both the feature extraction process as well as the identity verification through the pre-built user profiles are critical steps regarding performance, and they must be accomplished in a short period of time.

It should be also taken into account that a real environment may have to authenticate multiple users at the same time (potentially thousands of users), so the entire process needs to be scalable. Hence, in the mid-term, we are facing a Big Data scenario where Data Mining methodologies will be required in order to achieve such scalability. In particular, we can point to a series of relevant processes, such as data acquisition on the client device, data transmission over a computer network, and feature extraction and identity verification through such features in an ad-hoc computing platform, as well as user profile updates. All of them must be handled in distributed computing platforms.

To summarize, this is a complex process that requires many different components that must be appropriately orchestrated in order to achieve a near-real-time, accurate, and scalable solution capable of authenticate users through mouse dynamics in a transparent manner. The present work focuses on analyzing the feasibility of mouse behavioral features as a second-phase authentication mechanism intended for continuous monitorization of the users' activity by means of Artificial Intelligence techniques. Any further development, such as a parallel and distributed implementation or its integration in a real environment, are well beyond the scope of this work and they are proposed as future work.

3 Data acquisition and treatment

In order to analyze the capabilities of mouse behavior as an authentication factor, we firstly decided to define a guided scenario where the user is exposed to different test cases oriented to frequent user interactions: read some texts of different sizes, so that the user has to scroll down; try to replicate some predefined shapes, such as circles or curves; close different pop-up windows that are randomly placed all over the screen; click on a button once the mouse pointer is randomly placed in the screen, so that the user has to locate it; and, forced waiting periods, where the application seems to be frozen. To this purpose, we developed an ad-hoc application that guides the user through these test cases and collects mouse movement data in a transparent manner to the user. These data include mouse position in the screen and a timestamp for each event, which are persisted to log files together with information that allows us to identify the owner of such a session.

Our application was circulated among 27 volunteers who completed the list of test cases, so that we can rely on the origin and authenticity of their data. Such a dataset also includes captures made with different types of pointing devices that work with different DPI (dots per inch) resolutions, as it would occur in a real environment. As a result, we were able to gather more than 1.5 million mouse events from three different user sessions, so that we could check data variability over time, allowing us to successfully conduct the experiment. In order to anonymize the dataset, and since our research group has an extensive background in the Big Data Astronomy field, we decided to use star names instead of user names, and therefore we named this set the Constellation Mouse Movements. Such a dataset is available to the community,[1] so that it can be used to conduct further studies on behavioral authentication based on mouse movements.

On the other hand, we have also used the Balabit Mouse Challenge Dataset [15], which is a public dataset that has been widely studied by different authors. Such a dataset gathers mouse events from ten users over remote desktop sessions, collecting mouse positions and timestamps among other information. It is divided into two different subsets: the training sample contains legitimate user sessions that are intended for algorithm training; and, the test sample, which provides data on sessions of unknown users along with an estimation of their identities. Such predictions were made by a number of models built during the actual challenge, so that such labels may not be completely reliable. Therefore, we decided to focus on the training set, which allowed us to test our techniques using external data that has not been gathered by our own system, and in a non-guided scenario. This dataset allows us to demonstrate the capability of the extracted features and the proposed methods to carry out the authentication process in a real and general-purpose environment.

3.1 Feature extraction

The raw data that was collected in both scenarios must be processed in order to extract those features relevant to the problem that can be appropriately handled by the machine learning techniques that will be presented in Sect. 4. To this purpose, we analyzed different proposals from the literature that used features of a diverse nature, mainly based on movement properties [10, 11, 14] such as velocities, accelerations, and angle variations, but also others based on movement analysis by means of time series. In this work, we focused on the first type of features, since we pretend to study the feasibility of mouse movement as an authentication factor for general purpose, rather than to mimic a specific pattern. Accordingly, we have selected a number of reference features from [14], which are summarized in Table 1.

In addition, we also considered some new features, derived from the previous ones: absolute values, computed for most of the reference features described above; a decomposition of some features into basic components, such as horizontal velocity which was split into velocity towards left and velocity towards right, for a clear physiological reason; we divided each session into different sequences of consecutive movements (hereinafter, chunks[2]) and took the first event of each one as the reference to compute the following quantities with respect to it: distances, slope angle of tangent, curvature, and curvature rate of change; finally, we compute the curvature and the curvature rate of change with respect to the origin. These features are fully described in Table 2.

For the purpose of continuous monitoring, these characteristics are structured in two types of temporally sequential chunks as follows: on the one hand, we defined time-based chunks lasting one, two and five seconds, which gather a variable number of events for each one. On the other hand, size-based chunks were defined which contain exactly 100, 200, and 500 events, respectively. Non-overlapped chunks were used in both cases, which provided six different scenarios that were tested independently.

3.2 Feature preprocessing

Once the features have been extracted, they must be appropriately treated before they are analyzed or used to train any machine learning technique. To this purpose, the following operations were performed:

- Firstly, outlying events were detected and removed for each user, so that they cannot affect the analysis of the data. Such occurrences can be due to a lack of concentration, distractions, or even correspond to an unrepresentative mood of the user.
- Then, the mean value was computed for the features contained within each chunk, which allows us to obtain representative information about it. This preprocessing step was skipped during the preparation of the data for the deep learning techniques (see Sect. 4) because they are able to work with chunk-based features that were configured as a hyper-parameter.
- Finally, all the features were scaled to the interval $[0, 1]$ following a minimum-maximum approach.

[1] The Constellation Mouse Movements dataset can be found in: https://lia2.udc.es/web/form_submission/constellation_dataset.php.

[2] These sequences of movements within a session that we refer to as “chunks” can be also found in the literature as “windows”. We decided to use the term “chunk” to avoid confusion with the concept of “sliding window” introduced in Sect. 5.

Table 1 Reference features from [14]

Feature	Definition
Horizontal velocity	$hv_i = \frac{x_i - x_{i-1}}{t_i - t_{i-1}}$
Vertical velocity	$vv_i = \frac{y_i - y_{i-1}}{t_i - t_{i-1}}$
Tangential velocity	$tv_i = \sqrt{hv_i^2 + vv_i^2}$
Tangential acceleration	$ta_i = \frac{tv_i - tv_{i-1}}{t_i - t_{i-1}}$
Tangential jerk	$tj_i = \frac{ta_i - ta_{i-1}}{t_i - t_{i-1}}$
Distance from the origin	$l_i^0 = \sqrt{x_i^2 + y_i^2}$
Slope angle of the tangent from the origin	$\theta_i^0 = \arctan\left(\frac{y_i}{x_i}\right)$
Curvature	$c_i = \frac{\theta_i^0 - \theta_{i-1}^0}{l_i^0 - l_{i-1}^0}$
Curvature rate of change	$\delta c_i = \frac{c_i - c_{i-1}}{l_i^0 - l_{i-1}^0}$

where x_i and y_i are the mouse positions and t_i the timestamp for a given record i within the session

3.3 Feature selection

Once the features have been treated, we proceed to analyze them in order to identify a number of relevant features. In this case, we focused on a recursive feature elimination method [24] based on Extra-Trees (ERT, [25]) that was also combined with a cross-validation procedure. The overall process was repeated many different times, and it demonstrated that all the 31 features were relevant enough in almost all the cases, so that they should be taken into account by the machine learning procedures that will be discussed in Sect. 4.

4 Methods used to create user profiles

In order to define a continuous authentication scheme, user's identity must be verified using the behavioral features described above, defining unique user profiles capable of detecting whether it is the user who is logged in or an impostor. To this purpose, different Artificial Intelligence techniques were explored, from classical methods such as Support Vector Machines (SVM, [21]) or Multi-Layer Perceptrons (MLP, [19, 20]) to Deep Learning (DL) techniques, such as Convolutional Neural Networks (CNN, [22]) or Long Short-Term Memory (LSTM, [23]) networks.

A user-based approach was followed in order to train and evaluate each of these techniques, and to find the model or profile more suitable for a particular user. Two samples of data are defined for each user: the own user records and a random selection of other users records. This allows us to guarantee that both samples are balanced and to avoid overfitting/underfitting. As a reminder, all the 31 behavioral features were used, and both datasets (Constellation and Balabit) were tested using time-based chunks (one, two, and five seconds) and size-based chunks (100, 200, and 500 events), so that twelve different full experiments were conducted as it is detailed below.

Each of the proposed techniques requires different hyper-parameters to be appropriately set, which are summarized in Table 3. In order to determine the best configuration for each one, a grid-search procedure was followed. It was combined with a cross-validation procedure to guarantee the generalization of the models, where each model was trained up to 10 times with a random selection of train/test samples (90% training, 10% test). In addition, the entire process was repeated a hundred times so that the sample of non-user data could be rebuilt with different records to avoid any bias and properly test the performance and generalization of these models.

For the deep learning models, the architecture design has been included as an additional variable, where not only the hyper-parameters of each type of neural network are configured, but also the number of layers for each type, and even the combination of different types in the same architecture (e.g., [26, 27]). In this case, tests have been carried out with convolutional and recurrent neural networks, using the Long Short-Term Memory (LSTM) architecture in the second case. The idea behind this configuration where both networks are combined is, on the one hand, to exploit the temporal locality of the data through convolutional layers, which are in charge of extracting increasingly higher-level characteristics at each step; and, on the other hand, identify patterns over time through recurrent neural networks.

It should also be noted that by hyper-parameterizing the number of layers for each type of networks, not only the capability of these networks to work together has been tested, but also the independent operation of each one, since the number of layers can be set to zero. As a result of this approach, after running the grid-search procedure, a single deep learning model is obtained as the combination of the two types mentioned above (see Fig. 1). The final parameters can be seen in Table 3.

Regarding the data used for experimentation, both datasets were used to perform the training and to measure their performance through a cross-validation procedure (see Sect. 6.1). In addition, the Constellation dataset was also used to conduct a rejection test, whose results are discussed in Sect. 6.2, since it requires a set of non-legitimate users that are not part of the system (outsiders), whereas the Balabit dataset was not used due to the limited user data available.

Finally, note that Deep Learning techniques are able to handle chunk-based features through a hyper-parameter, namely, the number of timesteps. Consequently, raw

Table 2 Custom features

Feature	Definition
Absolute horizontal velocity	$abs\_hv_i = \|hv_i\|$
Horizontal velocity towards left	$hv_i^l = \begin{cases} \|hv_i\| & \text{if } hv_i < 0 \\ 0 & \text{otherwise} \end{cases}$
Horizontal velocity towards right	$hv_i^r = \begin{cases} \|hv_i\| & \text{if } hv_i > 0 \\ 0 & \text{otherwise} \end{cases}$
Absolute vertical velocity	$abs\_vv_i = \|vv_i\|$
Vertical velocity downwards	$vv_i^d = \begin{cases} \|vv_i\| & \text{if } vv_i < 0 \\ 0 & \text{otherwise} \end{cases}$
Vertical velocity upwards	$vv_i^u = \begin{cases} \|vv_i\| & \text{if } vv_i > 0 \\ 0 & \text{otherwise} \end{cases}$
Absolute tangential acceleration	$abs\_ta_i = \|ta_i\|$
Tangential deceleration	$ta_i^d = \begin{cases} \|ta_i\| & \text{if } ta_i < 0 \\ 0 & \text{otherwise} \end{cases}$
True tangential acceleration	$ta_i^t = \begin{cases} \|ta_i\| & \text{if } ta_i > 0 \\ 0 & \text{otherwise} \end{cases}$
Absolute tangential jerk	$abs\_tj_i = \|tj_i\|$
Positive tangential jerk	$tj_i^p = \begin{cases} \|tj_i\| & \text{if } tj_i < 0 \\ 0 & \text{otherwise} \end{cases}$
Negative tangential jerk	$tj_i^n = \begin{cases} \|tj_i\| & \text{if } tj_i > 0 \\ 0 & \text{otherwise} \end{cases}$
Distance from the chunk reference	$l_i^{cr} = \sqrt{(x_{cr} - x_i)^2 + (y_{cr} - y_i)^2}$
Slope angle of the tangent from the chunk reference	$\theta_i^{cr} = \arctan\left(\frac{y_i - y_{cr}}{x_i - x_{cr}}\right)$
Absolute curvature	$abs\_c_i = \|c_i\|$
Curvature within the chunk	$c_i^{cr} = \frac{\theta_i^{cr} - \theta_{i-1}^{cr}}{l_i^{cr} - l_{i-1}^{cr}}$
Absolute curvature within the chunk	$abs\_c_i^{cr} = \|c_i^{cr}\|$
Absolute curvature rate of change	$abs\_\delta c_i^0 = \|\delta c_i^0\|$
Curvature rate of change within the chunk	$\delta c_i^{cr} = \frac{c_i - c_{i-1}}{l_i^{cr} - l_{i-1}^{cr}}$
Absolute curvature rate of change within the chunk	$abs\_\delta c_i^{cr} = \|\delta c_i^{cr}\|$
Curvature from the origin	$c_i^0 = \frac{\theta_i^0 - \theta_{i-1}^0}{l_i^0}$
Curvature rate of change from the origin	$\delta c_i^0 = \frac{c_i^0 - c_{i-1}^0}{l_i^0}$

where x_i and y_i are the mouse positions and t_i is the timestamp for a given record i within the session

feature values can be used instead of average chunk values as in classical techniques. In contrast, Deep Learning techniques represent a high cost in computational resources, reason why the overall process to both find the hyperparameters and evaluate the models were cut to ten times, instead of a hundred times.

5 Continuous authentication scheme using a weighted sliding window

It must be stressed that none of the models proposed above can work by themselves outside an environment responsible for orchestrating the entire process: collect and process the mouse data to obtain the features, integrate the aforementioned models and build a continuous authentication system that, based on the predictions obtained for each chunk, identifies possible intrusions. Please note that the design and implementation of such a platform is beyond the scope of this work. This section intends to just show an example of continuous authentication using a sliding

Table 3 Hyper-parameters

Technique		Hyper-parameter	Best configuration
SVM		Kernel	Gaussian
		Regularization parameter (C)	10
		Kernel coefficient (γ)	10
MLP		Number of hidden layers	1
		Neurons in the hidden layer	20
		Learning rate (α)	0.001
		Activation function	Hyperbolic tangent
DL			
	CNN	Number of filters	64
		Kernel size	3
		Padding	"Same"
		Pool size	2
		Number of timesteps	100
		Number of layers	2
	LSTM	Units	32
		Dropout rate	0.35
		Number of timesteps	100
		Number of layers	2
	Fully conn. ANN	Neurons	50

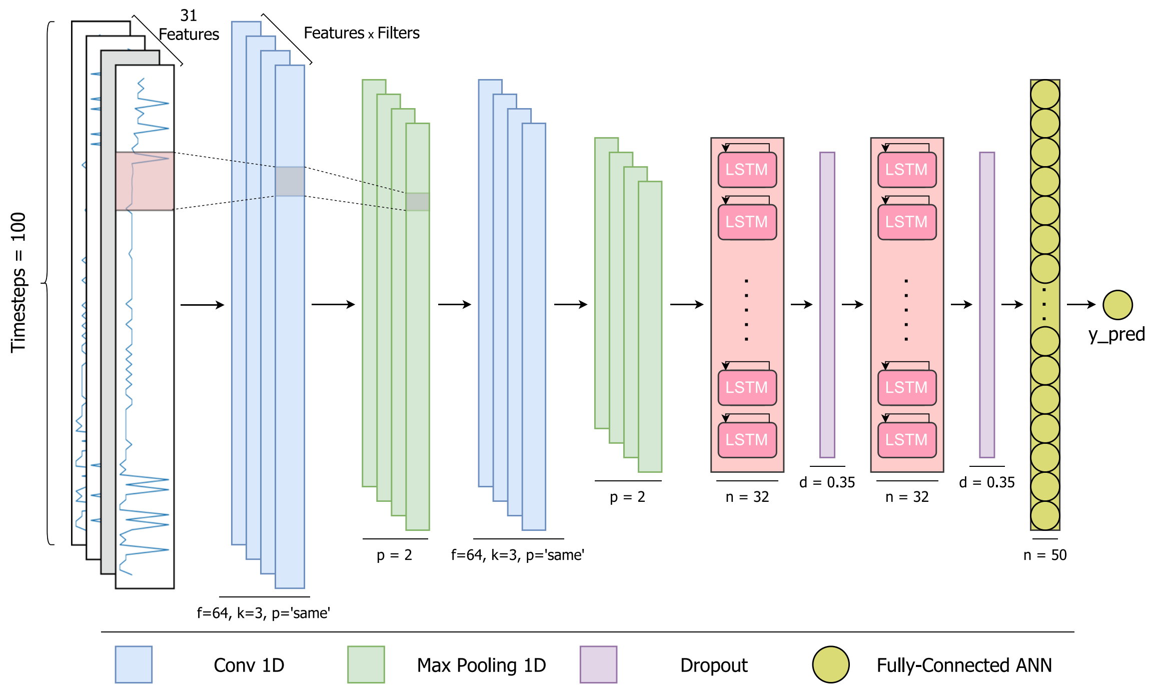

Fig. 1 Deep learning architecture overview. The model uses two 1D convolutional neural layers, where f is the number of filters, k is the size of the kernel, and p is the padding. A pooling layer is applied to the output of each layer, where p is the size of the kernel. Both types of layers use a stride of 1. n is the number of neurons in the fully-connected layer and the number of units in the 1D recurrent neural network. For dropout layers, d is the dropout rate

window procedure, where the assessments are based on an average of the predictions given by the models described above. Note that each individual prediction is the result of processing a single chunk, as described in Sect. 4. The entire process is explained in the next paragraphs and illustrated by Fig. 2, which shows a scenario where the legitimate user is using the computer for a certain period of time, and then an impostor takes control over the session.

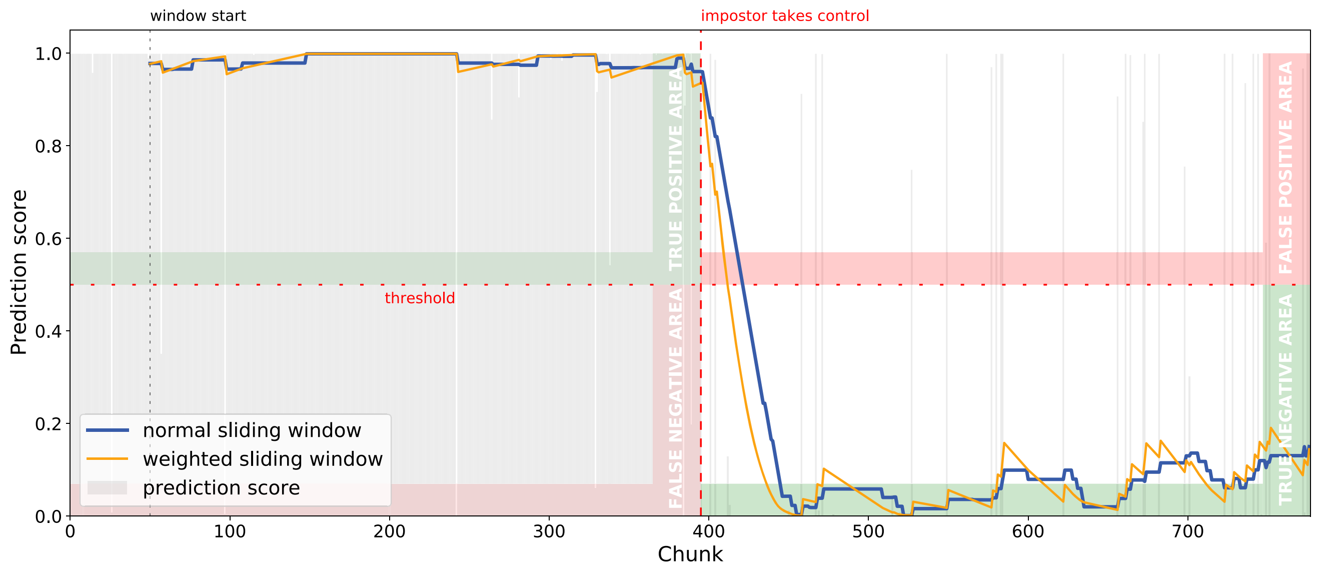

Fig. 2 Example of continuous authentication using the Deep Learning model for user Nash (Constellation dataset). While the first 395 iterations correspond to the user's own test data, the rest of the iterations belong to user Altair. An authentication threshold of 0.5 and a window size of 50 were used

The prediction of whether the user is the legitimate one or an impostor is shown through a set of gray bars, which take continuous values between 0 and 1. Such bars represent the confidence that each one of the chunks provides about the legitimacy of the user. The continuous authentication procedure uses a sliding window that computes the mean of the predictions in two different ways:

(a) A classical arithmetic mean ($\bar{x}$), where each prediction that falls within the window interval is weighted equally:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} p_i, \tag{1}$$

where n stands for the size of the sliding window, and p_i represents the prediction for the ith chunk.

(b) A weighted mean ($\overline{x_w}$), in which the latest predictions in the interval take a larger weight value:

$$\overline{x_w} = \frac{\sum_{i=1}^{n} p_i w_i}{\sum_{i=1}^{n} w_i}, \tag{2}$$

where w_i represents the weight for the ith prediction. Such a weight can be determined as $w_i = i/m$, being m the mitigating factor that is used to control the effect of weighting, and $i/m \in \mathbb{N}$. This way, the lower is m, the more weight is applied to the most recent chunks of the window.

The results for both classical and weighted sliding windows are represented with blue and orange lines, respectively. A value for the kth window represents the average value (weighted or not) of the predictions between the kth and $(k-n)$th chunks. As the comparison between both types of means shows, the weighted one has a faster response to new predictions, helping to detect an impostor earlier when it takes control of the device, but being also more sensitive to false negatives.

In addition to the different types of weighting, the size of the sliding window has also been tested. In this case, the larger the size of the window, the more "memory" it has about the user's behavior. Note that although with smaller windows reactivity is reinforced, it is also more prone to a false negative output.

The last parameter to configure is the authentication threshold, which is used to detect a possible identity theft when the value of the sliding window is below such a threshold. In Fig. 2, the red dotted horizontal line represents the threshold, and the vertical dotted line represents the moment when the impostor user takes control over the session. Using these two lines, the graph can be divided into four zones, which represent a kind of confusion matrix: the upper left part would represent the area of the true positives, where it is the legitimate user who is using the device and the continuous authentication system predicts so; the lower right area corresponds to the situation where the impostor takes control and the system is able to detect that it is not the legitimate user who is in control (true negative); the upper right area represents the worst possible case, when the impostor is in control and the authentication method determines that it is the legitimate user (false positive); and, finally, when the value of the sliding window falls in the lower left area, the system would be identifying the legitimate user as an impostor (false negative), and it would be asking again for the primary authentication credentials. Despite being an undesirable situation, it is obviously preferable that the system forces the user to authenticate again using the primary method in exceptional moments, even if it is the legitimate user (false negative), than a false positive situation where the impostor can use the device without being detected.

This method was evaluated using both datasets, so that an outsider (external user not known by the system) was used in the Constellation dataset, and an impostor which is part of the system (another legitimate user different from the model owner) was used in the Balabit. The overall results are presented in Sect. 6.3.

6 Results

Once the models have been appropriately configured, their performance was evaluated through different widely-used metrics such as precision, recall, or the $F_1 - score$ [28]. Firstly, a cross-validation procedure was performed in order to obtain the base results that demonstrate that the proposed methods are able to authenticate the legitimate users. Then, a rejection test was carried out on the legitimate user models built in the previous stage, so that we can evaluate their capability of detecting and prevent session usurpation by an outsider user. In addition, an overview of the performance of the continuous authentication scheme described in Sect. 5 is also shown. Finally, since computing time is one of the main issues in a continuous authentication scheme like the one we are proposing, we have measured evaluation times for these models.

6.1 Cross-validation

As it was stated above, all the scenarios were independently tested for both datasets, using the different chunk configurations proposed as well as the methods described. This led us to a large amount of results that will be summarized in this section.

Firstly, time-based chunks were found to perform slightly worse than the fixed-size approach. This is due to the high variability on the number of records available per chunk: sometimes the user makes many moves within a short time frame, but in some cases, such as a waiting period in a loading window, the user does not make any movement for several seconds. Therefore, using time-based chunks seems not suitable for such situations, whereas fixed-size have proved to perform better and to be more stable.

For this reason, we decided to focus on the latest type of chunks (fixed-size). We found that they performed similarly for the different sizes that were taken into account (100, 200, 500). However, since our goal is to authenticate users in a continuous manner over the entire user sessions, using small-sized chunks will allow us to carry out the authentication process in a more frequent manner than using larger sizes, while reducing the computational cost of the entire process. Hence, we are presenting the results obtained for each one of the proposed techniques using fixed-size chunks of 100 events for both datasets.

Regarding authentication, the main objective is to minimize false positives; that is, to detect any intrusion into the system, which can be efficiently measured through False Positive Rate (FPR) or precision score. However, false negatives must also be taken into account because repeatedly rejecting a legitimate user can lead to a disruption of the users' activities and may be annoying. Consequently, False Negative Rate (FNR) or recall score can be used to minimize false negatives and, finally, both aspects can be evaluated at the same time through the $F_1 - score$ or Area Under the Receiver Operating Characteristic curve (AUC-ROC). Table 4 summarizes the performance of the proposed models through the average value for these metrics.

On the one hand, we started analyzing the Constellation dataset (see Table 4), where we obtained average scores above 0.85 for those metrics that are intended to be maximized, while for the FPR and FNR, which are intended to be minimized, we obtained average values below 0.14, and 0.10 respectively. This means, in general terms, that the proposed methods are capable of authenticating the users achieving a good performance, especially for the Deep Learning models. To further illustrate this behavior, Fig. 3

Table 4 Average performance for the evaluated metrics range during the cross-validation procedure

		Model		
		DL	MLP	SVM
Constellation dataset				
Metric	$F_1 - score$	0.9218	0.8976	0.8876
	Precision	0.9001	0.8902	0.8728
	Accuracy	0.9185	0.8964	0.8842
	Recall	0.9473	0.9073	0.9060
	AUC-ROC	0.9615	0.9494	0.9397
	FPR	0.1111	0.1146	0.1376
	FNR	0.0527	0.0927	0.0940
Balabit dataset				
Metric	$F_1 - score$	0.9691	0.9593	0.9577
	Precision	0.9594	0.9594	0.9543
	Accuracy	0.9679	0.9593	0.9574
	Recall	0.9795	0.9595	0.9615
	AUC-ROC	0.9900	0.9877	0.9880
	FPR	0.0441	0.0409	0.0466
	FNR	0.0205	0.0405	0.0385

shows the average $F_1 - score$ for each user. Although it presents some user-dependent variability, it always takes values above 0.80. Two different facts that could lead to certain degree of confusion among users can be pointed out: users were asked to complete exactly the same test cases, which required common actions such as replicate some shapes or clicking on specific items; and the number of available records can considerably vary from one user to another, mainly because of the pointing device used, but also due to their different behavior on mouse usage.

On the other hand, once our authentication scheme was successfully proven in the guided scenario which included some mandatory mouse actions, the next step was to analyze its performance in a non-guided scenario with totally free movements. To carry it out, we decided to use the Balabit dataset because it was obtained by an external system and, as we mentioned before, it was widely analyzed by other authors, allowing the comparison of the results obtained by our system to those others [15]. In general terms, the performance of our models was found to be particularly good for the Balabit dataset, obtaining scores above 0.95 for those metrics that are intended to be maximized, scores below 0.05 for the ones intended to be minimized (FPR and FNR), whereas the average $F_1 - score$ is always above 0.90 for all users. Figure 4 illustrates these results and demonstrates that if we compare them with those published in [15], a substantial improvement is found. In particular, columns "MM" in Table 4 of the mentioned paper [15] contain results for features based on mouse movements, providing average values for accuracy and AUC-ROC metrics of 0.79 and 0.86, respectively.

6.2 Rejection test

Besides the cross-validation test described above, a second validation step has been carried out. As it was mentioned above, the objective of this phase is to detect fraudulent use of the system by an external or outsider user, who is not part of the system. Therefore, we used data records from 7 users in the Constellation dataset that were not used during the training process of the methods, so that they can be considered as true outsider users.

Therefore, those users that were not considered during the training and cross-validation tests, will be now tested against the legitimate user models built in the previous stages (see Sect. 4). Since we are exclusively using novel user data to perform this test, the metrics used are limited to: true negative rate, which measures those cases where the impostor users are detected as so; and, false positive rate, which measures the acceptance of impostor users as legitimate ones.

Table 5 shows the average results obtained for this test. The overall performance marginally decreases when compared to the cross-validation results presented in Sect. 6.1, although the DL methods continue to perform slightly better. However, this subtle worsening could be expected, since the data used in this test belong to new users that

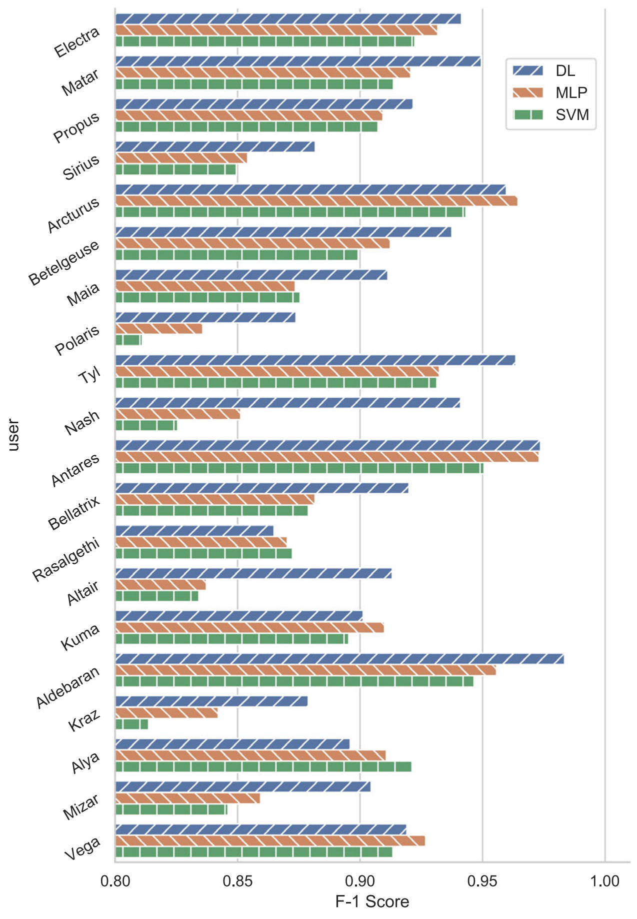

Fig. 3 Constellation user comparison in the cross-validation

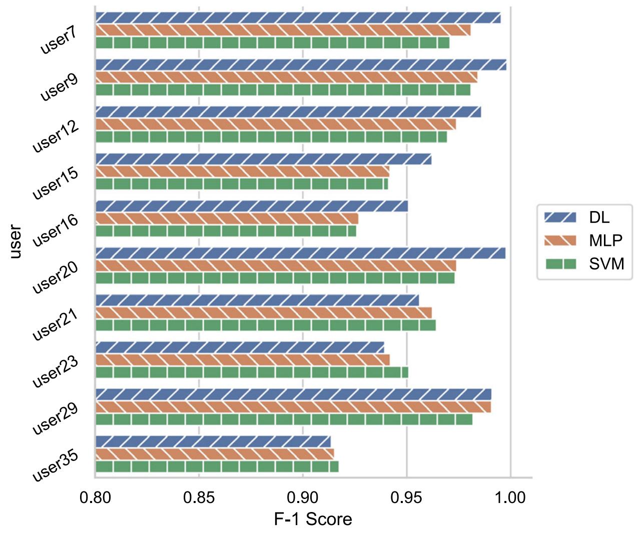

Fig. 4 Balabit user comparison in the cross-validation

Table 5 Average rate of true negatives (TNR) and false positives (FPR) for the rejection test conducted

		Model		
		DL	MLP	SVM
Metric	TNR	0.8664	0.8427	0.8362
	FPR	0.1336	0.1573	0.1638

were never presented to the models during the training phase. In addition, it must be also emphasized that in our dataset all the users were required to complete exactly the same actions and, therefore, some confusion was also expected to an extent.

6.3 Continuous authentication test

In this section, we discuss the results obtained during the experimentation using the sliding window procedure for continuous authentication described in Sect. 5, which are summarized in Figs. 5, 6 and 7.

On the one hand, Fig. 5 shows different comparisons of the continuous authentication procedure, by applying weighted windows with different sizes and thresholds to DL predictions. In summary, smaller windows increase the speed of the system's response to changeable values, allowing more sensitivity to erratic predictions. On the other hand, they also allow a faster detection of an impostor, so that they are more robust to false positives or false negatives. This happens at the cost of a slower response to detect a true negative case. The thresholds are used to determine when the procedure must intervene to avoid an intrusion: the larger value it takes, the higher security is. The downside is that a large value makes the system more likely to trigger the primary authentication mechanism to a legitimate user due to a wrong prediction (false negative), which can be annoying. However, when it takes small values, interruptions are minimized at the expense of less security. At this point, it must be taken into account that the value of a single window falling below the threshold is enough to ask for the primary authentication credentials. In this way, as it can be seen in the example, the impostor would be quickly detected even using a window size of 10 chunks and a threshold of 0.5.

On the other hand, Figs. 6 and 7 show the results for two different users of the Constellation dataset and the Balabit dataset, respectively. In both cases, a weighted window with the predictions of the proposed models (DL, MLP, and SVM) were used, as well as a window size of 25 and a threshold of 0.5. As it can be observed, although in both cases the values of the weighted windows are in the correct areas (true positive, and true negative) most of the time, a better performance is appreciated with the Balabit dataset. This was our expectation according to the results obtained in the cross-validation tests that were performed and reported previously.

It must be pointed out that, in order to implement an operational system, these parameters must be appropriately defined in order to minimize false negatives and, especially, false positives. In addition, it must be taken into account that they strongly depend on the similarity between the legitimate user and the impostor, as well as on the effectiveness of each user's models, so that their parameters may differ.

Finally, the continuous authentication test has also been tested using an outsider user as impostor. Most of the graphs between legitimate users and outsiders maintain a behavior similar to Fig. 6, with the DL model being the one that behaves best in general terms, as supported by the results shown in Sect. 6.2. In certain cases, the subtle worsening mentioned in the rejection test can be seen in Fig. 8. Despite this, a significant decrease in the values obtained by the sliding window is still observed when the outsider takes control, allowing for the detection of impostor users even in more complicated scenarios. In fact, this can be reinforced by increasing the detection threshold, so that the continuous authentication system is more sensitive to potential window prediction drops.

6.4 Performance time measurement

As it was previously stated, computing times are one of the main concerns for a continuous authentication system, in which near real-time is a requirement in order to become fully operational when integrated into a real environment. Therefore, in this section we report the execution times measured for each one of the proposed models when processing a different number of chunks to make a prediction about the legitimacy of the user.

Table 6 shows the execution times measured for all three models that were used to perform the authentication when they have to process data in two scenarios: around an hour of user time; and, near three hours of user time. As it can be observed, the fastest method is the MLP, which can carry out the process in the order of milliseconds, demonstrating an outstanding performance with respect to the other methods. Despite the SVM and DL techniques are considerably slower than MLP, they can be perfectly used for a continuous authentication system, since all of them are capable to process all the data gathered in an hour of user time in less than a second. However, it must be pointed out that the number of users that the system has to analyze concurrently must be taken into account in order to

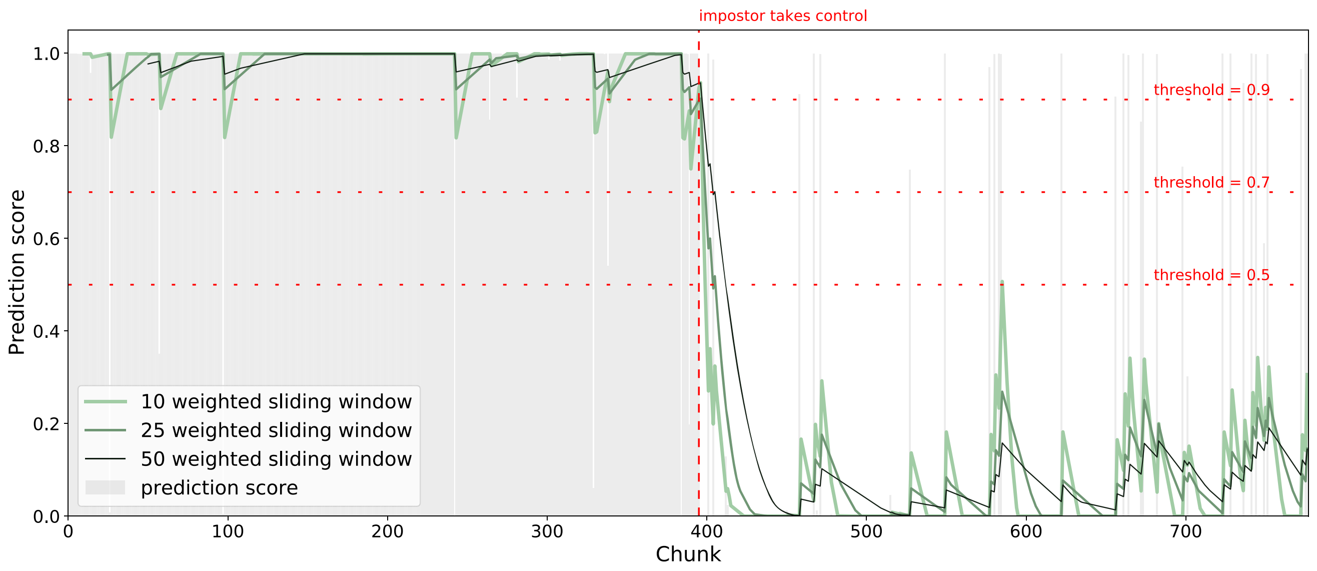

Fig. 5 Example of continuous authentication using different window sizes - It exemplifies the same case explained in Figure 2, but using only the weighted approach with window sizes of 10, 25, and 50; and three different thresholds of 0.5, 0.7, and 0.9

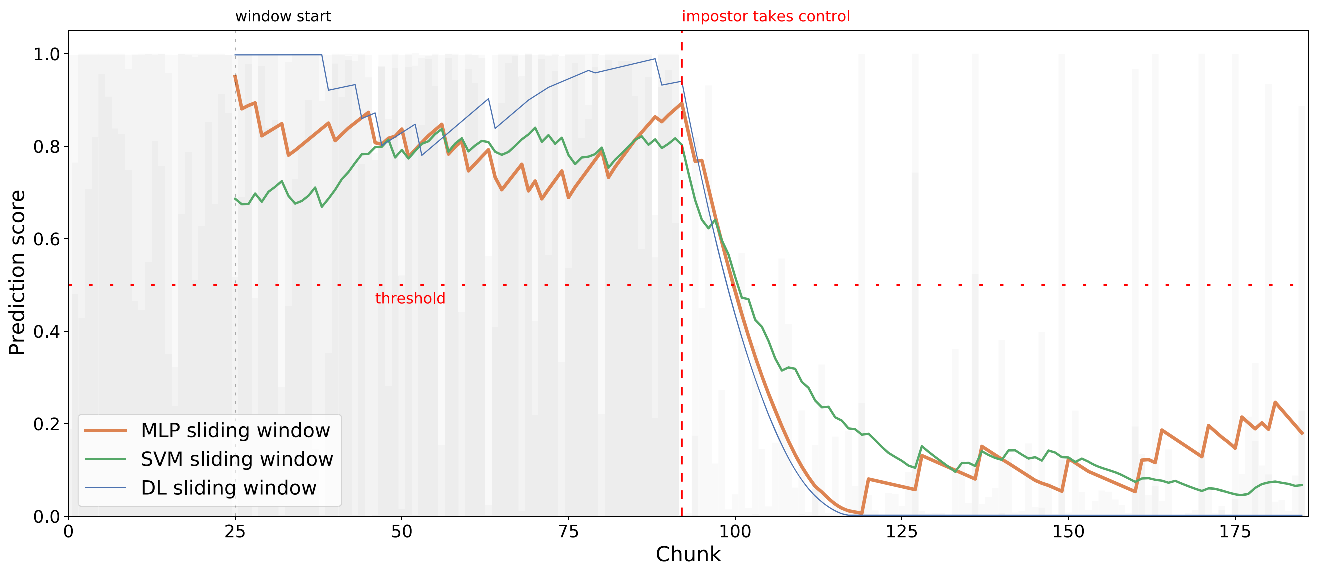

Fig. 6 Example of continuous authentication with the three proposed techniques applied to Constellation dataset and using the weighted approach with a size of 25 - The first 94 iterations correspond to the Kuma user (legitimate user), while the rest belong to Arcturus

determine which method should be used for a particular organization or platform.

7 Conclusions and future work

In this work, a full experiment was conducted in order to assess the capability of behavioral mouse movement features in order to authenticate the user by means of Artificial Intelligence techniques. Firstly, a non-intrusive mouse movement monitorization tool was developed and deployed to 27 users so that we could conduct the experiment in a guided scenario, where each user was asked to complete a number of tests that involved different mouse actions. Then, based on the information that was gathered, a wide variety of mouse movement features were extracted, studied, and processed in order to build unique user profiles using three different AI techniques: MLP, SVM, and DL.

The performance of these methods was analyzed following three different approaches. First, a classical cross-validation procedure was carried out to assess the authentication capability of the features and the techniques to authenticate users according to their mouse movement

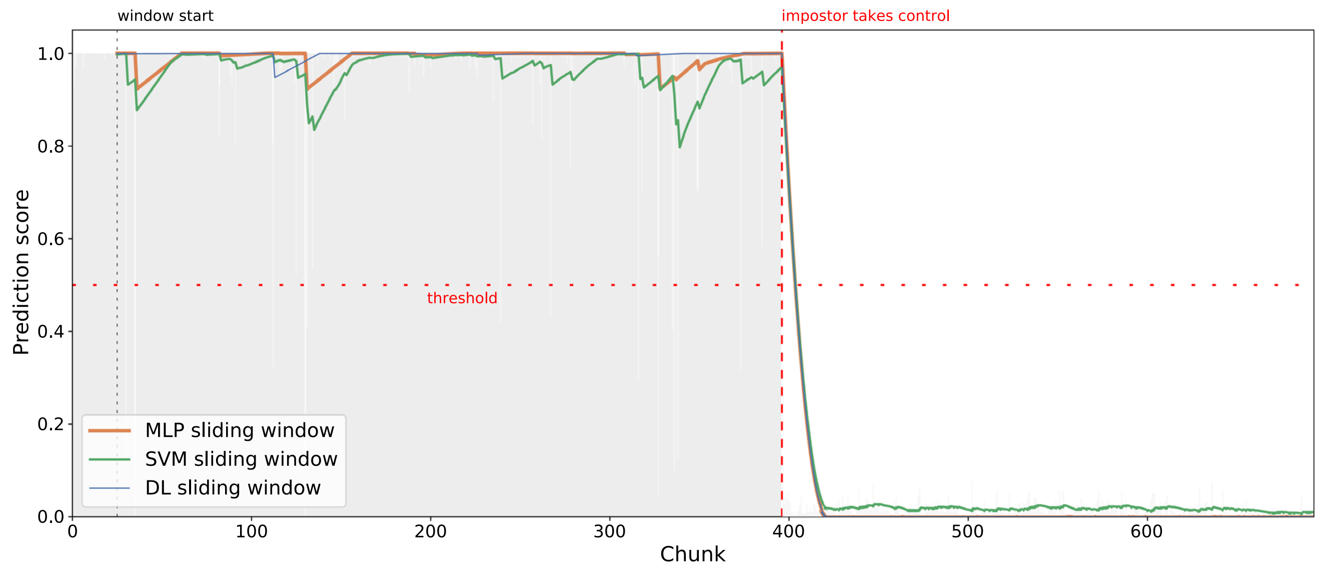

Fig. 7 Example of continuous authentication with the three proposed techniques applied to Balabit dataset and using the weighted approach with a size of 25 - The first 396 iterations correspond to the user7 (legitimate user), while the rest belong to user12

behavior. In general terms, all the techniques proved to be successful, but MLP and, especially, DL demonstrated to perform slightly better. Then, a rejection test was conducted in order to assess the capability of these methods to detect a non-legitimate user that is not part of the system (i.e. an outsider user), obtaining consistent results with respect to cross-validation. Finally, a continuous authentication test was also carried out in order to simulate a regular working environment where an impostor takes control over the session of a legitimate user. To this purpose, a sliding window procedure was proposed and

Table 6 Comparison of execution times for the models used

Method	Chunks	User time (h[a])	Execution time (s)
MLP	3600	1	0.003
	10,000	2.75	0.009
SVM	3600	1	0.995
	10,000	2.75	3.085
DL	3600	1	0.607
	10,000	2.75	1.601

[a]User times were estimated according to the average user time per chunk

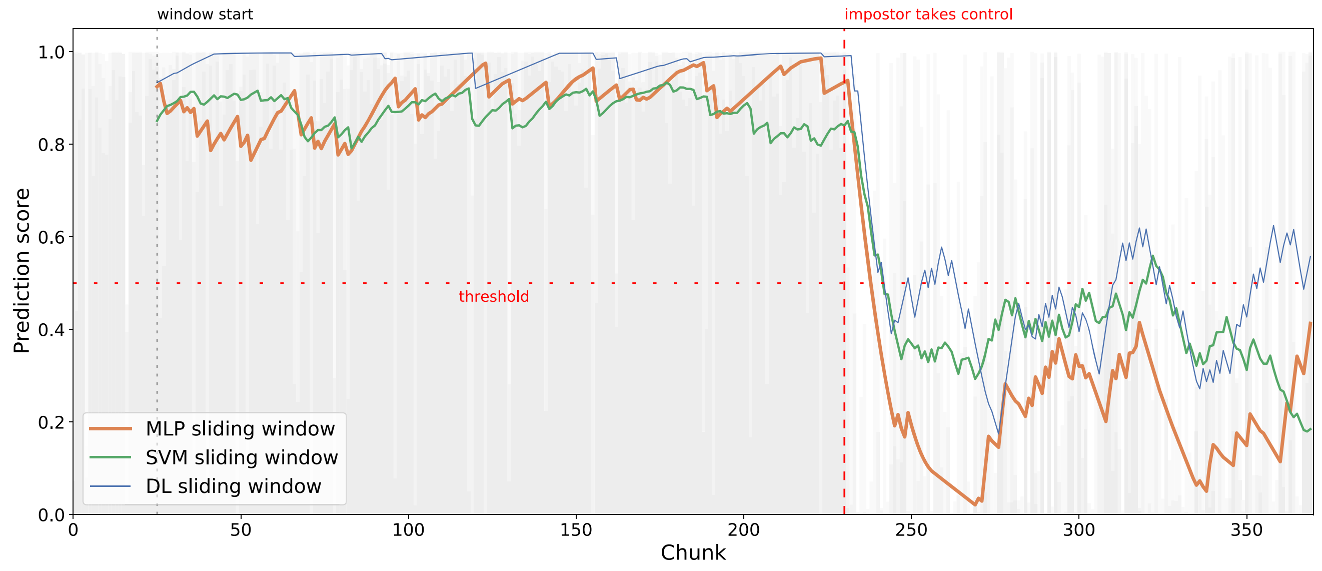

Fig. 8 Example of continuous authentication with the three proposed techniques applied to Constellation dataset and using an outsider user as impostor. The first 229 iterations correspond to the Tyl user, the rest of the iterations belong to user Zaniah. An authentication threshold of 0.5 and a window size of 25 were used

applied to carry out the authentication process, which proved to be effective on minimizing false negatives, while being able to identify false positives (impostor users) in a short period of time. Additionally, performance times were measured to verify that it is feasible to implement such a system in a real-time working environment.

The scope of this work did not cover the design and implementation of an entire authentication system, which is therefore proposed as future work. Such a system would integrate the authentication methods proposed in this work, as well as other future improvements, into a real-time platform that would be responsible for the entire authentication process, orchestrating all of the required components: data acquisition, data storage, feature extraction, feature preprocessing, authentication via AI-based user profiles, as well as a notification system that could trigger some action when an identity usurpation is detected. Due to the enormous volumes of data that must be handled, such a platform would require the use of a Big Data solution capable of processing such data in real-time or near real-time, such as a queue-based and streaming-based systems (e.g. Apache Kafka, Apache Spark or Eclipse Mosquitto). Additionally, such user profiles must be updated over time, so that the models are periodically adapted to variations in the user's behavior (e.g. temporal or permanent injury) or to changes in physical devices (e.g. acquisition of a new pointing device), and some trigger mechanisms must be proposed and studied in order to automatize such an update process. This update of the modules does not require real-time processing, but it is computationally expensive due to the vast amounts of data that must be handled, requiring also a Big Data solution that provides distributed computing capabilities, such as Apache Spark or Apache Hadoop.

Acknowledgements This work made use of the infrastructures acquired with Grants provided by the State Research Agency (AEI) of the Spanish Government and the European Regional Development Fund (FEDER), through RTI2018-095076-B-C22, and PID2019-525 111388GB-I00. We acknowledge support from CIGUS-CITIC, funded by Xunta de Galicia and the European Union (FEDER Galicia 2014-2020 Program) through Grant ED431G 2019/01; research consolidation Grant ED431B 2021/36; Art.83 collaboration F19/03 with the enterprise Odeene S.L.; and scholarship from Xunta de Galicia and the European Union (European Social Fund - ESF) ED481A-2019/155.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Guttman B, Roback EA (1995) An introduction to computer security: the NIST handbook. Tech. Rep., Special Publication (NIST SP), National Institute of Standards & Technology, Gaithersburg, MD, USA
2. Stallings W, Brown L (2012) International computer security: principles and practice, 2nd edn. Pearson Education Limited, Boston
3. López-Vizcaíno M, et al. (2018) Network data unsupervised clustering to anomaly detection. Proceedings 2(18). https://doi.org/10.3390/proceedings2181173
4. López-Vizcaíno M, et al. (2019) IEEE (ed.) Network data flow clustering based on unsupervised learning. (ed. IEEE) 2019 IEEE 18th international symposium on network computing and applications (NCA). IEEE, Piscataway, NJ, USA, pp 1–5
5. Shirey RW (2007) Internet Security Glossary, Version 2. RFC 4949
6. Barkadehi MH et al (2018) Authentication systems: a literature review and classification. Telemat Inform 35(5):1491–1511. https://doi.org/10.1016/j.tele.2018.03.018
7. Ahmed AAE, Traore I (2005) IEEE (ed.) Anomaly intrusion detection based on biometrics. (ed. IEEE) Proceedings from the sixth annual IEEE SMC information assurance workshop. IEEE, Piscataway, NJ, USA, pp 452–453
8. Bhatnagar M et al (2013) A survey on behavioral biometric techniques: mouse vs keyboard dynamics. Int J Comput Appl 975:8887
9. Ahmed AAE, Traore I (2007) A new biometric technology based on mouse dynamics. IEEE Trans Depend Secure Comput 4(3):165–179. https://doi.org/10.1109/TDSC.2007.70207
10. Zheng N, et al. (2011) ACM (ed.) An efficient user verification system via mouse movements. (ed. ACM) Proceedings of the 18th ACM conference on computer and communications security. Association for Computing Machinery, New York, NY, USA, pp 139–150
11. Gamboa H, et al. (2003) An identity authentication system based on human computer interaction behaviour. ICEIS Press, 46–55
12. Ometov A et al (2018) Multi-factor authentication: a survey. Cryptography. https://doi.org/10.3390/cryptography2010001
13. Pusara M, Brodley CE (2004) ACM (ed.) User re-authentication via mouse movements. (ed. ACM) Proceedings of the 2004 ACM workshop on visualization and data mining for computer security, 1–8. Association for Computing Machinery, New York, NY, USA
14. Sayed B et al (2013) Biometric authentication using mouse gesture dynamics. IEEE Syst J 7(2):262–274. https://doi.org/10.1109/JSYST.2012.2221932

15. M. Antal & E. Egyed-Zsigmond. Intrusion detection using mouse dynamics. IET Biometrics 8 (5), 285–294 (2019). DOI: 10.1049/iet-bmt.2018.5126
16. Mekruksavanich S, Jitpattanakul A (2021) Deep learning approaches for continuous authentication based on activity patterns using mobile sensing. Sensors. https://doi.org/10.3390/s21227519
17. Havrylovych M, Hovorushchenko T, et al. (eds) (2020) Comparative analysis of using recurrent autoencoders for user biometric verification with wearable accelerometer. In: Hovorushchenko T, et al. (eds) Proceedings of the 9th international conference "information control systems & technologies". Information control systems & technologies
18. Damopoulos D, Portokalidis G (2019) Hands-free one-time and continuous authentication using glass wearable devices. J Inf Secur Appl 46:138–150
19. Werbos PJ (1974) Beyond regression: new tools for prediction and analysis in the behavioral sciences. Ph.D. thesis, Harvard University
20. Rumelhart DE, McClelland JL (eds) (1986) Parallel distributed processing: explorations in the microstructure of cognition: foundations. MIT Press, Cambridge
21. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297. https://doi.org/10.1007/BF00994018
22. Fukushima K, Miyake S (1982) Neocognitron: a self-organizing neural network model for a mechanism of visual pattern recognition. Springer, Berlin, pp 267–285
23. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–80. https://doi.org/10.1162/neco.1997.9.8.1735
24. Guyon I et al (2002) Gene selection for cancer classification using support vector machines. Mach Learn 46(1–3):389–422. https://doi.org/10.1023/A:1012487302797
25. Geurts P et al (2006) Extremely randomized trees. Mach Learn 63:3–42. https://doi.org/10.1007/s10994-006-6226-1
26. Van VD, et al. (2017) ACM (ed.) Combining convolution and recursive neural networks for sentiment analysis. (ed. ACM) Proceedings of the eighth international symposium on information and communication technology. Association for Computing Machinery, New York, NY, USA, pp 151–158
27. Islam Z et al (2020) A combined deep CNN-LSTM network for the detection of novel coronavirus (COVID-19) using X-ray images. Inform Med Unlocked 20:100412. https://doi.org/10.1016/j.imu.2020.100412
28. Tharwat A (2021) Classification assessment methods. Appl Comput Inform 17(1):168–192. https://doi.org/10.1016/j.aci.2018.08.003