IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 29, 2021, accepted September 2, 2021, date of publication September 7, 2021,
date of current version September 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3111098

SapiAgent: A Bot Based on Deep Learning to Generate Human-Like Mouse Trajectories

MARGIT ANTAL, KRISZTIAN BUZA, AND NORBERT FEJER
Department of Mathematics-Informatics, Faculty of Technical and Human Sciences, Sapientia Hungarian University of Transylvania, 540485 Targu Mures, Romania

Corresponding author: Margit Antal (manyi@ms.sapientia.ro)

This work was supported in part by the Sapientia Hungariae Foundation-Collegium Talentum project and MTA Domus Hungarica Scientiarum et Artium.

ABSTRACT The growing interest in bot detection can be attributed to the fact that fraudulent actions performed by bots cause surprisingly high economical damage. State-of-the-art bots aim at mimicking as many as possible aspects of human behavior, ranging from response times and typing dynamics to human-like phrasing and mouse trajectories. In order to support research on bot detection, in this paper, we propose an approach to generate human-like mouse trajectories, called *SapiAgent*. To implement SapiAgent, we employ deep autoencoders and a novel training algorithm. We performed experiments on our publicly available *SapiMouse* dataset which contains human mouse trajectories collected from 120 subjects. The results show that SapiAgent is able to generate more realistic mouse trajectories compared with Bézier curves and conventional autoencoders.

INDEX TERMS Autoencoder, bot agent, bot detection, mouse dynamics, mouse trajectory.

I. INTRODUCTION

Distinguishing between humans and artificial intelligence (AI), a.k.a. *bot detection*, is dating back to 1950 when Alan Turing developed the famous test named after him [1]. What was a curiosity in the fifties, has now grown into a serious problem in the field of application security: meanwhile, a surprisingly high fraction of world wide web traffic is generated by professional bots developed by skilled engineers using artificial intelligence [2].

Bot detection methods can be grouped into active and passive algorithms. Active methods or CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) use online tasks such as distorted character recognition or object detection in images which are supposed to be hard for bots. These tasks have several drawbacks. First of all, bots using AI may become very good at solving such challenges. Additionally, the aforementioned CAPTCHA tasks may be difficult for human users as well. Consequently, passive detectors are being used more frequently as they are more comfortable for humans, because these detectors analyze user interactions with the device. Regarding passive detectors, including passive bot detection based on mouse dynamics, we point out that it can be used continuously as opposed to active detection which is usually performed only once, at the authentication step.

Google reCAPTCHA systems, besides other sources of information, use mouse dynamics for bot detection.[1] In this paper, we investigate how safe it is to perform bot detection based on mouse trajectories. In particular, we propose an approach to generate human-like mouse trajectories with neural networks. As this method might potentially be used by a bot, we call this approach *SapiAgent*. Our empirical results show that SapiAgent is able to generate realistic mouse trajectories. Therefore, as its most prominent application, SapiAgent has a high potential to support research on bot detection by providing a challenging detection task.

Our main contributions can be summarized as follows:

1) We present an analysis of our new SapiMouse dataset [3], containing mouse dynamics data from 120 users. When collecting the data, each user interacted with the system in two sessions, a longer 3-minute session followed by a shorter 1-minute session. In both sessions, the users had to solve simple tasks involving mouse movements. The dataset is publicly available.[2]

The associate editor coordinating the review of this manuscript and approving it for publication was Diana Gratiela Berbecaru.

[1] https://www.google.com/recaptcha/about/
[2] https://ms.sapientia.ro/~manyi/sapimouse/sapimouse.html

2) We propose SapiAgent, a method to mimic human mouse trajectories. SapiAgent is based on an autoencoder trained in a novel way encouraging the generation of realistic trajectories.
3) Our experiments demonstrate that most common anomaly detectors fail to reliably identify the trajectories generated by SapiAgent as outliers. Our results show that SapiAgent is able to generate more realistic mouse trajectories compared with the well-known approach based on Bézier curves and conventional autoencoders.
4) Our system is publicly available,[3] therefore it can be extended with other methods.

The remainder of this paper is organized as follows. In Sect. II we review the most important works related to mouse movement analysis with special focus on works based on bot detection and bot agents aiming at mimicking human behaviour. Sect. III. describes the problem of human-like mouse movement generation in detail, our new SapiMouse dataset and the methods used throughout the study, including our approach, SapiAgent. This is followed by the experimental evaluation in Sect. IV. The last section concludes the paper and presents possible directions of future work.

II. RELATED WORK

In this section, we review most relevant works in two domains: (a) mouse trajectory analysis, and (b) bot detection.

A. MOUSE TRAJECTORY ANALYSIS

In the last decades, user recognition based on mouse trajectories became a prominent field of research. Although mouse dynamics data contain less user-specific information than a signature, combined with other biometrics, such as keystroke dynamics [4], [5], mouse trajectories may allow for user recognition in various applications, e.g. in cases when the set of possible users is limited. In simple settings, user recognition is based on a single mouse trajectory [6]–[9], [10]–[12] or movements performed during a time interval of fixed length [13]. Other studies concluded that not all users can be identified with high precision solely based on a single mouse trajectory [10], [14].

Usually, features are extracted from a single mouse trajectory, then machine learning models are used for user recognition. In this setting, trajectories are compared based on the extracted features. In contrast, Qin *et al.* [15] proposed a novel progress-adjusted dynamic time warping algorithm to compare mouse trajectories. In recent works using deep neural networks, mouse trajectories were divided into blocks of fixed length. Promising results of user authentication were presented by Chong *et al.* [16] and Antal and Fejer [17]. In contrast, Tan *et al.* [18] presented different adversarial attack strategies against behavioral mouse dynamics.

Due to its emerging applications, mouse cursor movement analysis is becoming increasingly popular in various domains, including web search [19]–[22]. For example, Lagun *et al.* concluded that mouse movements constitute a rich source of information and can be used to improve search ranking [19]. Mouse cursor movements have also been used to infer user attention on search engine result pages (SERP) [20]. In a recent study, the same authors proposed efficient representations of mouse movements such as time series, heatmaps, and trajectory-based images in order to predict user attention to SERP advertisements [21].

User's demographic information such as age or gender may be inferred from mouse trajectories with reasonable accuracy, but an adversarial noise method may be used to protect the users' privacy [23].

In another recent study, Banholzer *et al.* concluded that computer mouse can be used to infer work stress [24]. This finding was reported based on a dataset collected from 71 persons using their computers for usual daily work.

Seelye *et al.* studied whether the mouse movement patterns can be used to predict mild cognitive impairment (MCI) [25]. They found that people affected by MCI tend to make fewer and more variable mouse movements.

Mouse movement analysis seems to be a valuable source of information in online learning environments as well. De Mooij *et al.* integrated mouse tracking in MathGarden, a large-scale online learning environment. Their findings suggest that mouse movement analysis may contribute to the diagnosis of systematic difficulties of the students [26].

B. BOT DETECTION

Bot detection is becoming more and more important because bots engage in an increasing number of manipulative activities. To address this problem, various approaches have been developed. For example, Suchacka *et al.* proposed a real time method for web bot detection [2], while Pozzana and Ferrara studied the behavior of bots in social media and devised an algorithm based on machine learning to detect bots on Twitter [27], whereas Pao *et al.* [28] proposed a trajectory-based bot detection approach for online gaming scenarios.

The most essential aspect of the implementation of bots is mimicking human behavior which is closely related to the synthesis of realistic trajectory data. Therefore, data augmentation is inherently related to bot detection. In case of data augmentation, new data are obtained by modifying existing data. Synthetic data generation for 2D gesture recognition was investigated by Taranta *el al.* [29]. Realistic gesture variations were created by selecting random points of human gestures trajectories and scaling the distances between these points.

Gianvecchio *et al.* [30] were the first who used mouse dynamics features to detect game bots. Chu *et al.* [31] adapted this approach for bot detection in blogs. These systems are built on the difference in behavior patterns between human users and bots, regarding their mouse and keystroke activities. Blog bots were detected with high accuracy using a stream of mouse trajectories and keystrokes.

[3]https://github.com/margitantal68/sapiagent

Generative adversarial networks (GANs) [32] are increasingly used for imitating shapes and images. Variants of GANs include conditional GANs [33] and Wasserstein GANs [34]. Acien *et al.* [35] proposed to detect bots based on a single mouse trajectory. The authors proposed two methods to generate human-like mouse trajectory data: an approach based on heuristic functions and a data-driven method based on GANs. Two versions of the bot detection problem were formulated: (i) an anomaly detection problem, and (ii) a binary classification problem. In the former case, One-Class Support Vector Machine (OCSVM) was used as an anomaly detector. In the latter case, several classification algorithms, SVM, k-Nearest Neighbors, Random Forest, Multi-Layer Perceptron, recurrent neural networks, such as Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) were employed.

In contrast to the aforementioned works, our SapiAgent is able to generate trajectories between arbitrary points. This feature allows our agent to have a much larger field of usage. In principle, it can also be used to generate gestures or trajectories of other types.

III. DATA AND METHODS

In this section, after the problem formulation, we present our data and their representation to train neural networks. Subsequently, we describe a solution based on the well-known Bézier-curves which we used as one of the baselines in our experiments. We close this section by developing our approach.

A. PROBLEM FORMULATION

In the literature, bot detection has been formulated both as a one-class classification problem [36] and as a binary classification problem [35], [37]. Due to the large variety of possible attacks, it is difficult to make realistic assumptions about attack types. Therefore, we decided to model the bot detection task as a one-class classification problem in case of which only human mouse trajectories are used to train the anomaly detector. For a new trajectory, the anomaly detector is expected to output a score that represents the deviation from normal (i.e. human) behavior. Based on this score, human trajectories may be distinguished from trajectories generated by a bot agent.

In this paper, we use wide-spread anomaly detectors for bot detection. With *agent*, we mean an approach that is able to generate trajectories mimicking human mouse trajectories. Our goal is to develop a new bot agent, SapiAgent, and examine how well state-of-the-art anomaly detectors are able to distinguish the data generated by SapiAgent from the real data associated with mouse movements performed by humans.

B. THE SapiMouse DATASET

The dataset contains mouse dynamics data for 120 subjects, out of which 92 are male and 28 are female. The age of the subjects ranged from 18 to 53 years. Participation was voluntary. Volunteers were students, professors and other members of the Sapientia Hungarian University of Transylvania. None of them was diagnosed with any disease that is known to influence the subject's mouse dynamics (such as Parkinson's or Alzheimer's Disease). The vast majority of the subjects (111 out of 120) were right-handed. A web-based game application was developed in which users had to move the mouse to different target coordinates on the screen. The game continuously logged the position of the mouse at a sampling frequency of approximately 60Hz during the game sessions. In each task of the game, the application displayed different geometric shapes (triangle, reversed triangle, square, circle) in random positions on the screen. The users had to move the mouse cursor to the geometric shape performing a left click, right click, double click or drag-and-drop depending on the shape. Raw data consisting of mouse cursor position (x, y), button type, event type (move, drag, pressed or released), and the corresponding timestamp were recorded.

Participants were instructed to use their own computer and were informed that their mouse activity would be logged. They were asked to complete two sessions that took for 3 minutes and 1 minute, respectively. These sessions are denoted by $S3$ and $S1$. In both sessions, the users had to solve as many of the aforementioned tasks of the game as they could. Both the data collection application[4] and the dataset are publicly available.

C. TRAJECTORY REPRESENTATION

Raw data were segmented into meaningful mouse actions. In this work, a mouse action is defined as a mouse trajectory consisting of a series of mouse events in which the mouse is moved between two points of the screen for a specific purpose. Specifically, in our dataset, each action corresponds to one of the user's tasks in the game as described in Section III-B.

As already mentioned, in both aforementioned sessions $S3$ and $S1$, the users had to solve similar tasks several times. Thus the actions obtained in $S3$ and $S1$ are from the same distribution. As the users had 3 minutes time in $S3$ instead of 1 minute in $S1$, they could solve more tasks in $S3$, resulting in roughly 3-times more actions in $S3$. Nevertheless, action trajectories in $S3$ are similar to the ones in $S1$ and vice versa, including their length.

Formally, a mouse action is defined as a sequence of $2D$ points $\{(x_0, y_0), (x_1, y_1), \ldots (x_n, y_n)\}$, which is referred to as trajectory from now on. Fig. 1 shows two typical mouse action trajectories. In addition to the trajectory, we also plotted the first-order differences according to the directions of the x and y axis (dx and dy) over time. As one can see, a typical mouse action is characterized by an acceleration phase at the beginning, followed by a deceleration phase, and a subsequent phase in which the velocity is constant.

[4]https://mousedynamicsdatalogger.netlify.app/

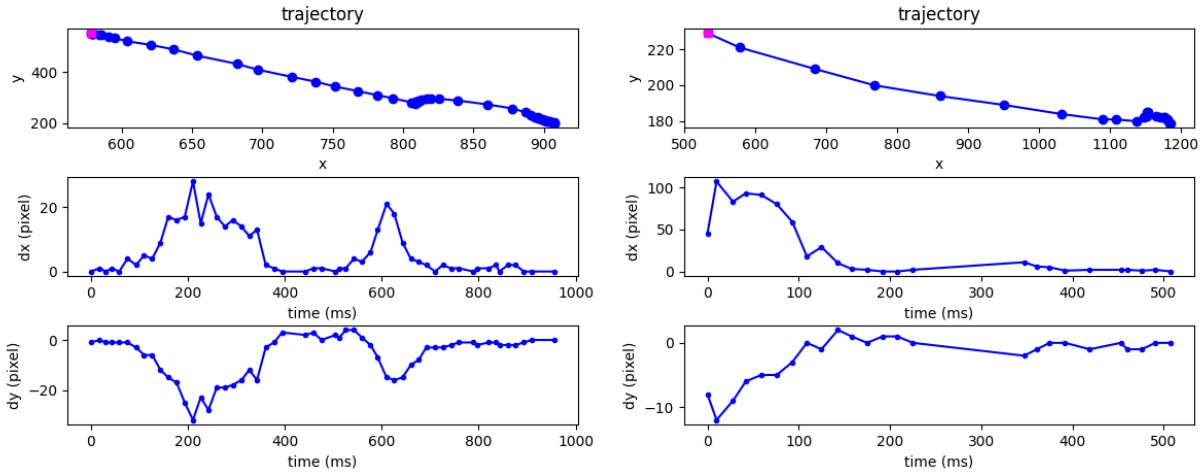

FIGURE 1. Human trajectories with their first order differences. The starting point is highlighted by magenta color.

TABLE 1. Descriptive statistics of the mouse actions in the *S*3 and *S*1 sessions of the SapiMouse dataset (length = number of points).

	3-min session ($S3$)	1-min session ($S1$)
total actions	16189	5205
min length	10.00	10.00
max length	363.00	239.00
mean length	36.17	36.72
std length	18.38	18.43

Short actions, containing less than 10 points were discarded. A mouse action contains 37 points on average and 99.5% of the actions contain less than 128 points. Therefore, mouse actions were represented with fixed-size arrays, the length of which was chosen to be 128. Instead of using the raw coordinates, we considered their first-order differences: $D = \{dx_1, dx_2, \ldots; dy_1, dy_2, \ldots\}$. We used zero padding for mouse actions shorter than 128 points. In case of those few actions (0.5% of all the actions) that contain more than 128 points, only the first 128 points were considered.

Table 1 shows the descriptive statistics of the actions in the SapiMouse dataset.

D. BÉZIER TRAJECTORIES

One of the most obvious ways to generate curves connecting specific points is the Bézier curve approach. A Bézier curve is a parametric curve named after Pierre Bézier, who used it for designing curves for cars at the Renault company. Bézier curves are generated using control points. The first and the last control point is the starting point and end point of the curve, while the intermediate control points generally do not belong to the curve, but the resulting curve approximates them. A Bézier curve is always bounded by the convex hull of its control points.

The pyclick[5] library was used to generate mouse trajectories based on Bézier curves. Trajectories were generated with the following strategy: (i) the start and end points and the length of each human trajectory were determined; (ii) for each human trajectory a corresponding Bézier curve was generated so that it passes through the start and end points, and its length equals the length of the original human trajectory. Two types of Bézier curves were generated: quadratic and cubic. Quadratic curves were generated by the *BezierCurve* class using a single intermediate control point. This control point was randomly generated inside the rectangle that has its diagonal between the two endpoints defined earlier. We consider these *baseline* curves. Cubic curves were generated by the *HumanCurve* class with default parameters. These parameters were defined to mimic human behavior. We call these cubic curves *humanlike*.

The left hand side of Fig. 2 shows a baseline Bézier trajectory with its first order differences and the right hand side shows the same for a humanlike Bézier trajectory. Both are compared to the original human trajectory they were generated from.

E. AUTOENCODERS

An autoencoder is a neural network that learns a low-dimensional representation of the input data. During the training process, an autoencoder learns two functions, an encoder which transforms the input sample into a latent code (representation), and a decoder which reconstructs the input from the latent code. As autoencoders are trained using unlabeled data, they belong to unsupervised learning methods. Traditionally, autoencoders are used for feature learning [38]. In such cases, the model is trained to reproduce the input on its output. Technically speaking, the same data (image, time series, etc.) serves as input and desired output when training the model. Therefore, we say that the autoencoder is trained *conventionally* if the same data is used as input and desired output.

We note that the notion of *autoencoder* may be used in a broad sense, e.g., *denoising autoencoders* [39] are trained to output clean data from noisy input. While essential properties of autoencoders, such as the encoder-decoder architecture, are kept in denoising autoencoders, when training these models, the input is not strictly the same as the desired output.

[5]https://pypi.org/project/pyclick/

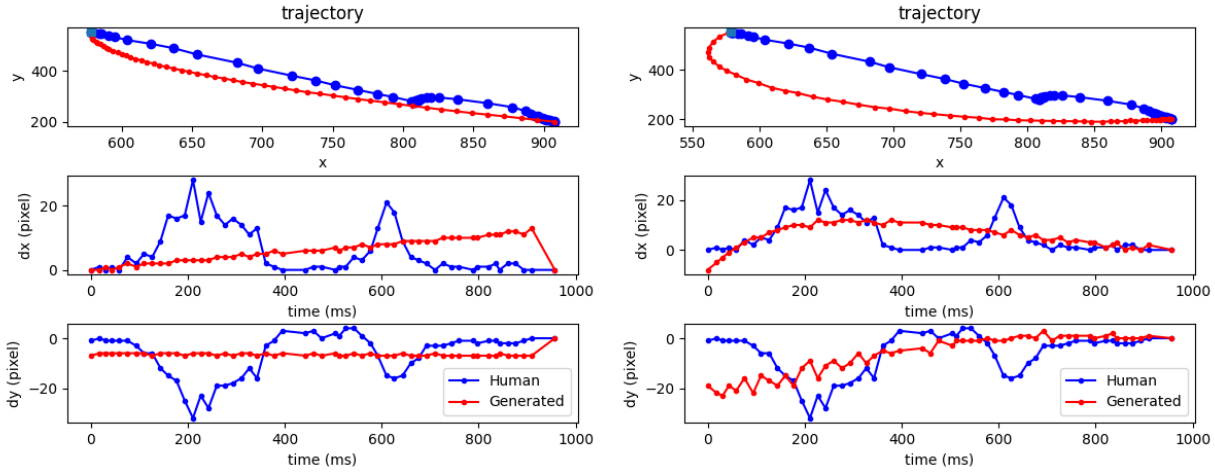

FIGURE 2. Human and its corresponding Bézier trajectories with their first order differences. Baseline (left) and humanlike (right).

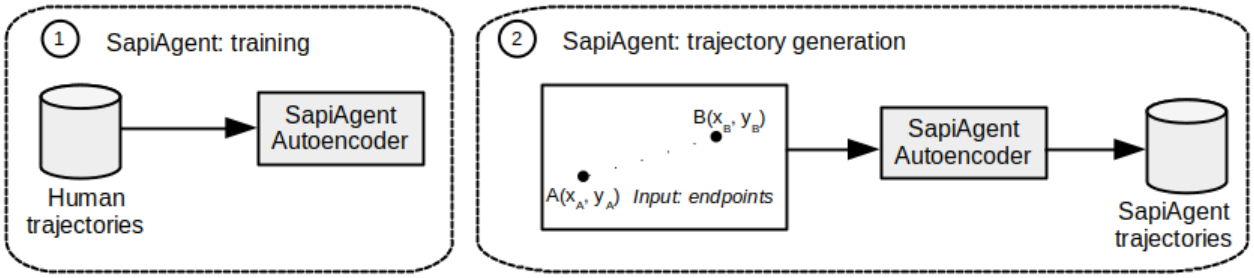

FIGURE 3. 1. Training the bot agent. 2. Generating mouse trajectories by the bot agent.

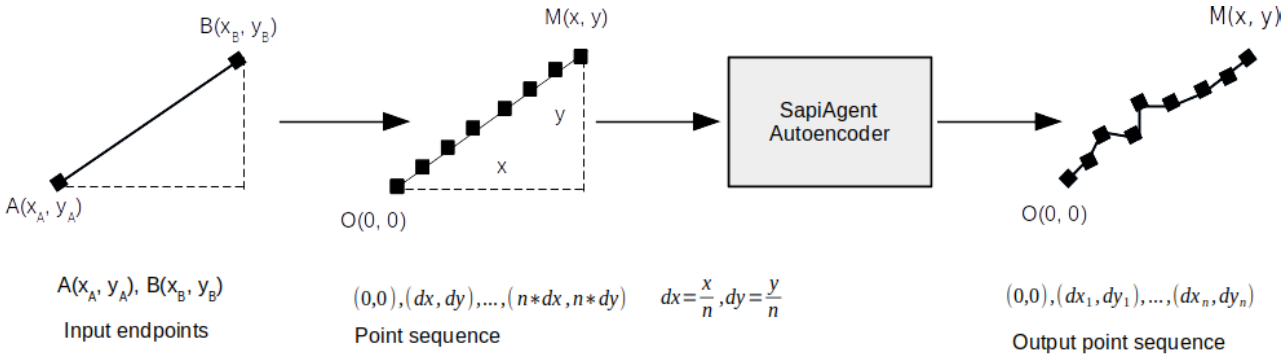

FIGURE 4. Autoencoder for generating human-like trajectory from equidistant points.

F. OUR APPROACH

The scheme of our approach is depicted in Fig. 3. As one can see, our bot agent, SapiAgent, is realized as an autoencoder. In the first step, we train SapiAgent to be capable of generating realistic trajectories. In the second step, SapiAgent is used to generate synthetic trajectories that mimic human trajectories. Next, we describe our approach in detail.

Let us suppose that we have two points $A = (x_A, y_A)$ and $B = (x_B, y_B)$, and the task is to connect these two points with a human-like trajectory as shown in Fig. 4. The segment AB can be translated to the origin obtaining the equivalent segment OM with the endpoints $O = (0, 0)$ and $M = (x, y)$, where $x = x_B - x_A$ and $y = y_B - y_A$. We divide the OM segment into n equal segments obtaining a sequence of points $(0, 0)$, (dx, dy), $\ldots(ndx, ndy)$, where $dx = \frac{x}{n}$ and $dy = \frac{y}{n}$. This sequence of points can be created easily for each pair of points on the screen. The task of the autoencoder is to learn the mapping between the equidistant sequence

$$(0, 0), (dx, dy), \ldots (ndx, ndy)$$

and real trajectories. Thus, when providing a new equidistant sequence, the autoencoder will be able to generate a realistic, human-like trajectory.

In the following, we describe the details of training the autoencoder and the process for generating synthetic trajectories.

We considered each human trajectory in the $S3$ session of the SapiMouse dataset. When training the autoencoder,

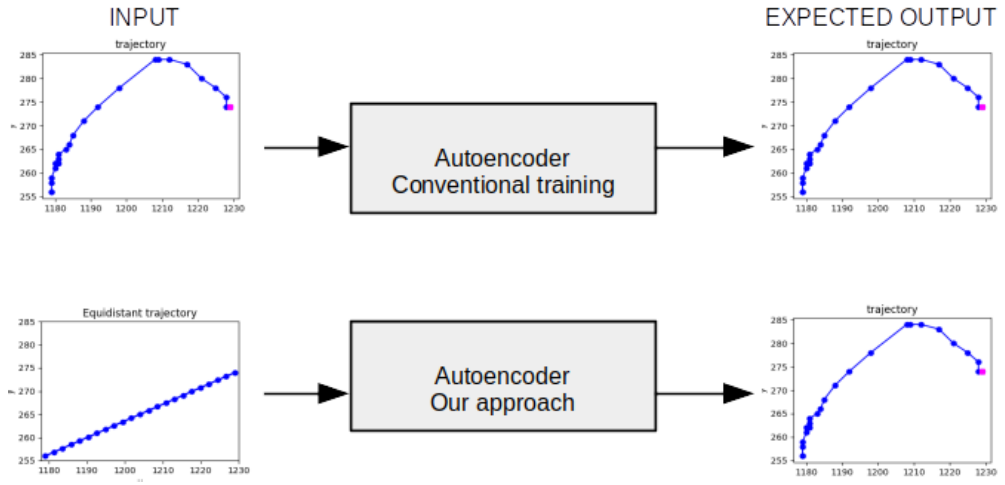

FIGURE 5. Conventionally trained autoencoder vs. our approach.

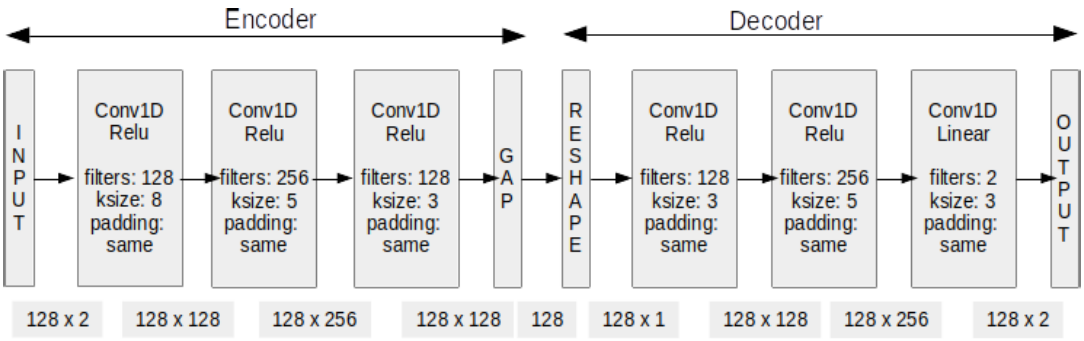

FIGURE 6. CNN-AE: Convolutional autoencoder architecture.

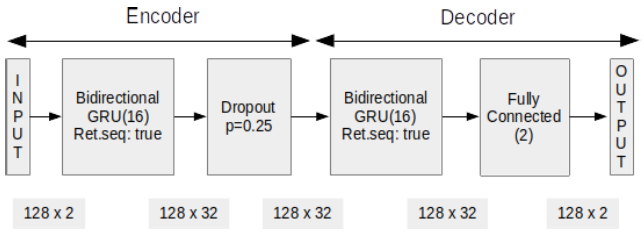

FIGURE 7. RNN-AE: Recurrent autoencoder architecture.

the corresponding equidistant sequence was used as input, and the desired output was the human trajectory. The reconstruction error is computed using a loss function between the actual output of the autoencoder and the human trajectory.

Let us consider a human mouse trajectory between two points. We represent this trajectory by its first order differences in x and y directions resulting in a two-dimensional sequence $D^{(h)}=\{dx_1^{(h)}, dx_2^{(h)}, \ldots, dx_n^{(h)}; \; dy_1^{(h)}, dy_2^{(h)}, \ldots, dy_n^{(h)}\}$. We create the input sequences for the autoencoder by dividing the segment between the endpoints of the trajectory into n equidistant sub-segments obtaining the following two-dimensional sequence $D^{(in)} = \{dx, 2dx, \ldots, ndx; \; dy, 2dy, \ldots, ndy\}$, where dx and dy are calculated as described above, see also Fig. 4. During training, the autoencoder generates an output $D^{(o)} = \{dx_1^{(o)}, dx_2^{(o)}, \ldots, dx_n^{(o)}; \; dy_1^{(o)}, dy_2^{(o)}, \ldots, dy_n^{(o)}\}$ for each input sequence. The reconstruction error is computed between the output two-dimensional sequence $D^{(o)}$ and $D^{(h)}$ by a loss function.

Fig. 5 shows the difference between conventional training and our approach. In the case of conventional training, both the input and the expected output are the same original trajectory. However, in our approach the input is the equidistant sequence between the endpoints of the original trajectory and the expected output is the original trajectory.

We note that, after training, our autoencoder is able to generate synthetic trajectories between two arbitrary points. However, we decided to generate mouse trajectories with the same endpoints as human ones, in order to closely mimic human trajectories.

In this paper, we consider two types of autoencoders: convolutional neural network autoencoder (CNN-AE) and recurrent neural network autoencoder (RNN-AE).

1) CONVOLUTIONAL NEURAL NETWORK AUTOENCODER (CNN-AE)

In their review, Fawaz *et al.* [40] compared various deep learning models for time series classification. They found that Fully Convolutional Neural Networks (FCN) performed best on multivariate time series. As mouse trajectories are multivariate time series, we build our CNN-AE approach on FCN.

FCN models contain pure convolutional layers without local pooling layers, so the length of the time series does not vary between convolutions. The original architecture proposed by Wang *et al.* [41] consists of three convolutional blocks. Each convolutional block contains a convolutional layer followed by ReLu activation. The first block contains 128 filters (kernel size: 8), the second block contains 256 filters (kernel size: 5), and the last block contains 128 filters (kernel size: 3). Each convolution preserves the length of the time series (stride: 1, zero padding). After the third convolutional block, global average pooling is used. We employ this construct in the encoding part of our model. The decoder is built in a similar fashion, but the order of layers is reversed and convolutions are replaced by transposed convolutions (Conv1DTransposed). No activation function was used in the last layer of the decoder, as our model has to generate sequences with positive and negative values. We trained this model (see Fig. 6) by the Adam optimizer with learning rate 10^{-4} for 100 epochs, and with a batch size of 32 samples.

Fig. 8 shows an original human trajectory and its corresponding trajectories generated by a CNN-AE. A trajectory generated by the conventionally trained CNN-AE is depicted in the left, while the trajectory generated for the same input by CNN-AE trained using our approach is shown in the right. As one can see, the model trained by our approach is able to generate a trajectory with the aforementioned three phases (acceleration, deceleration, constant velocity) characterising human trajectories, whereas these phases are not present in the trajectory generated by conventionally trained CNN-AE.

2) RECURRENT NEURAL NETWORK AUTOENCODER (RNN-AE)

Mouse movements are sequential in nature, therefore we also tested recurrent neural networks (RNNs) that are known to be effective in modeling time series. In particular, LSTM and GRU networks are known for their ability to remember long-term dependencies. Gates regulate the information flow, being responsible to select important information from time series. A two-way model is appropriate when it is necessary to learn both from previous and next observations.

Chung *et al.* compared LSTM and GRU recurrent units and found that GRU can outperform LSTM in terms of convergence and generalization [42]. These results are confirmed by recent studies in which bidirectional GRU-based models successfully predicted user attention [21], age and gender [23] from mouse trajectories. Consequently, we opted for bidirectional GRU.

In accordance with our trajectory representation, the input layer of our RNN-AE model consists of 128 values of a two-dimensional time series. The next layer is the forward-backward (bidirectional) recurrent block (Bi-GRU) with 16 output units configured to return sequences. This is followed by a dropout layer for regularization with a drop rate of 0.25. The decoder part consists of a similar Bi-GRU layer with 16 units followed by a fully connected layer to convert back to a two-dimensional time series. We used the same training parameters as for the CNN-AE model. Fig. 7 presents the architecture of our RNN-AE model.

Fig. 9 shows an original human trajectory and its corresponding trajectories generated by RNN-AE. In the left, a trajectory generated by a conventionally trained RNN-AE is shown. In contrast, a trajectory generated by the same RNN-AE trained using our approach is depicted in the right.

G. BOT DETECTION

The solution of a typical anomaly detection problem involves two steps. First, we model normal behavior. Based on this knowledge, we identify deviations in the second step.

In our case, training data contain only normal data (human mouse trajectories). During training, the anomaly detector learns a model of normal behaviour, in particular, how a human being produces mouse movements on the screen. An anomaly detector uses the model to produce anomaly scores. In the second step, the trained anomaly detector assigns a score to a given sample that represents a measure of deviation from normal behavior.

As we do not make assumptions about attacks, we train the anomaly detector using human mouse trajectories only. Subsequently, in order to evaluate the model, we use both human and synthetic mouse trajectories. We use wide-spread metrics, in particular, ROC (Receiver Operating Characteristics) curves, AUC (Area under the ROC curve), and EER (Equal Error Rate) to assess the performance of the model [43].

1) FEATURE EXTRACTION

In principle, trajectory data, as described in section III-C, could be used directly as input of anomaly detectors. However, anomaly detectors using appropriate features have been shown to be much stronger compared with using trajectory data directly. In particular, Gianvecchio *et al.* [30] observed that the efficiency of the mouse movement, defined as the ratio between the segment length and the traversed distance has a huge potential in bot detection (bots are more efficient than humans). Upon careful study of the dx and dy series of the trajectories, it can be seen that their smoothness is also different for human and bot movements. Usually, bots' movements are smoother than those of humans. Therefore, we computed the smoothness ($smth$) of dx and dy curves as follows:

$$smth(v_1, \ldots, v_n) = \frac{\sum_{i=2}^{n} |v_i - v_{i-1}|}{n-1}$$

where $v_1, \ldots, v_n$ corresponds to either $dx_1, \ldots, dx_n$ or $dy_1, \ldots, dy_n$ depending on whether the smoothness of dx or dy is calculated.

As histogram-based features perform well for many types of data [44], we decided to compute the histogram of dx and dy. In both cases, we used 5 equal bins. We used the values associated with $2 \times 5 = 10$ bins as features.

Additionally, basic statistical features such as minimum, maximum, mean, and standard deviations were also computed for dx and dy series, resulting in 8 statistical

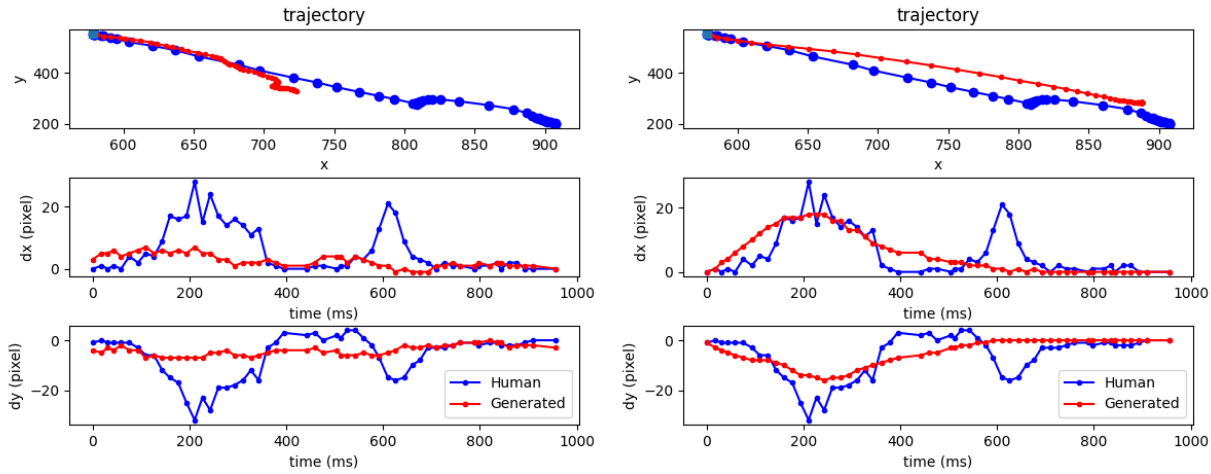

FIGURE 8. Human-like trajectories generated by CNN-AE with conventional training (left) and our approach (right).

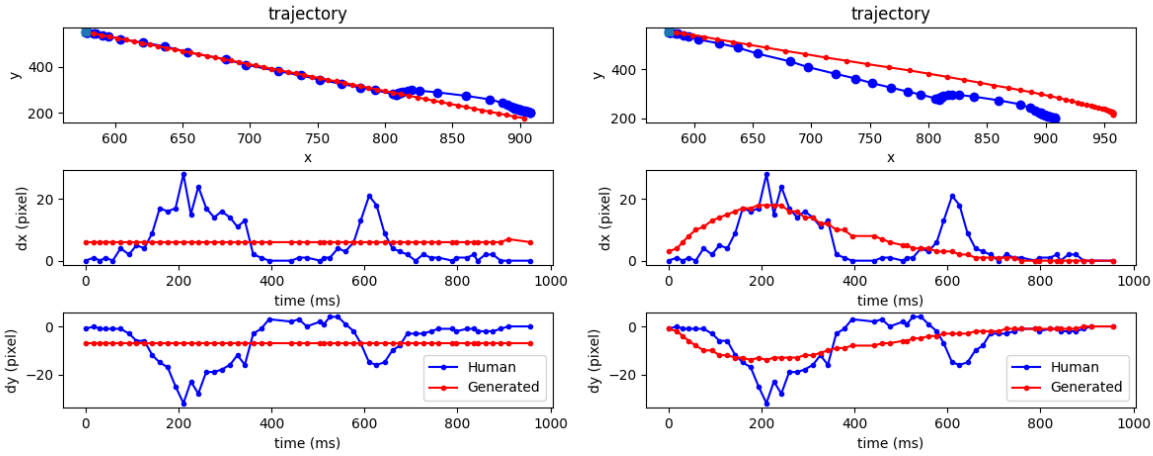

FIGURE 9. Human-like trajectories generated by RNN-AE with conventional training (left) and our approach (right).

features. Thus, the feature extraction resulted in 21 features in total.

2) ANOMALY DETECTORS

Due to its applications in intrusion detection and other domains, anomaly detection is a prominent research area. As a result, various approaches have been proposed for anomaly detection, such as distance-based techniques, detectors based on principal components and support vector machines, as well as ensemble approaches [45], [46]. Our goal was to employ the state-of-the-art anomaly detectors for our bot detection problem. Therefore, anomaly detectors were taken from PyOD [47], a Python toolkit containing various algorithms for anomaly detection in multivariate data. We performed initial experiments with all the available detectors. For simplicity, in Section IV we only report results of the strongest anomaly detectors, i.e., the ones that had the overall best performance in distinguishing human and synthetic trajectories, in particular: Principal Component Analysis (PCA), OCSVM, Local Outlier Factor (LOF), Clustering-Based Local Outlier factor (CBLOF), Isolation Forest (IForest), and Feature bagging.

PCA is a well-known method for dimensionality reduction. The covariance matrix of the data is decomposed to eigenvectors associated with eigenvalues, then the eigenvectors with the highest eigenvalues are retained. Outlier scores are calculated as a sum of euclidean distances to the eigenvectors [48]. OCSVMs are one of the state-of-the-art approaches for anomaly detection, due to their flexibility in fitting complex nonlinear boundaries between normal data and anomalies [49]. LOF [50] is a density based approach, that assigns a degree (called LOF) of being outlier to each data point. Outliers tend to have high LOF. CBLOF [51] defines a new outlier factor, which is calculated based on the size of the cluster the data point belongs to and the distance between the data point and its closest cluster. Isolation Forest [52] detects outliers based on a set of decision trees, being conceptually similar to Random Forest, a well-known robust classification algorithm. Feature Bagging [53] combines the results from applying the same detector using a randomly selected subset of features from the original feature set.

IV. EXPERIMENTS AND RESULTS

In this section, we assess anomaly detectors for their ability to distinguish real trajectories from synthetic ones generated

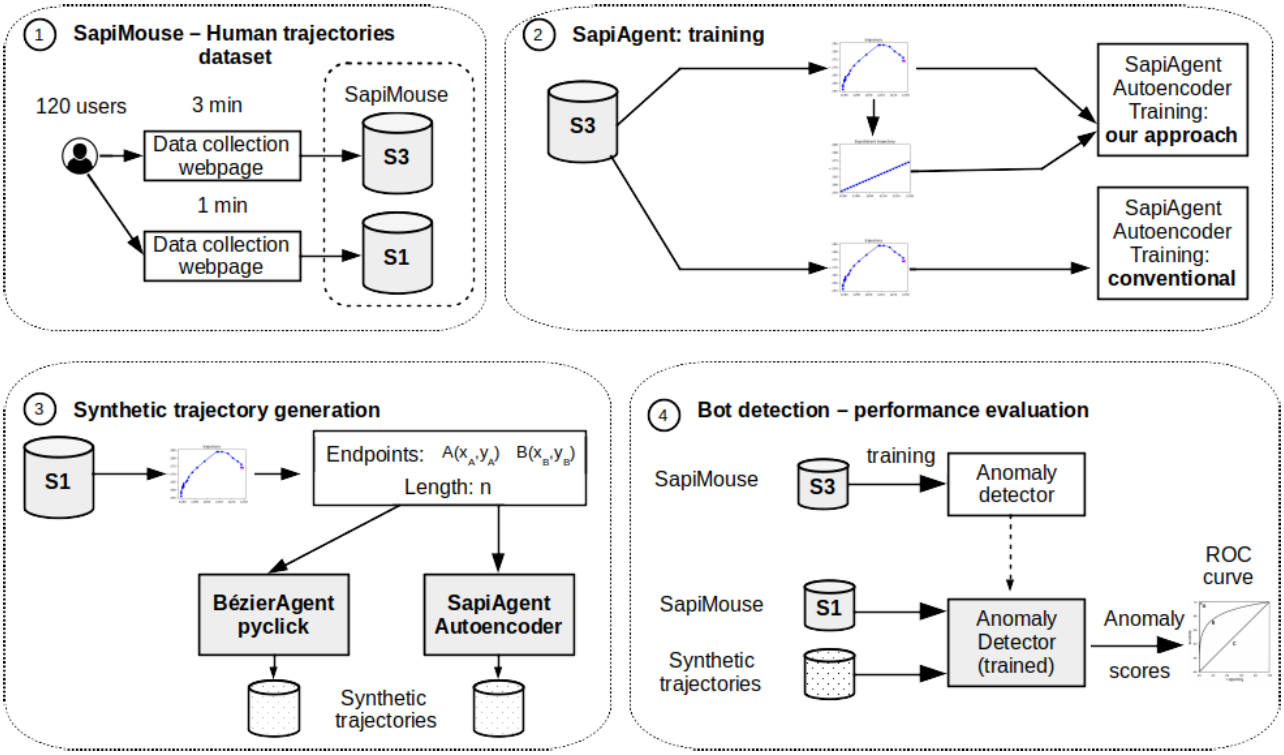

FIGURE 10. 1. SapiMouse dataset collection protocol. 2. Training the bot agent. 3. Generating synthetic trajectories 4. Bot detector performance evaluation.

by the various agents. Subsequently, we discuss the obtained results.

The entire methodology is depicted in Fig. 10. The first step shows the data collection protocol. Each user completed a 3-minute session followed by a 1-minute session, and the resulting mouse trajectories were stored in two separated subsets $S3$ and $S1$. In the next step, we trained SapiAgent. The third step shows the synthetic data generation process. The last step depicts the performance evaluation of our bot detector, implemented as an anomaly detector.

A. EXPERIMENTAL PROTOCOL

The goal of our experimental protocol is to simulate real-world applications of bot detection as closely as possible. In such applications, data from the past is used to train the system which is then applied to new data in the future. Therefore, we used data from session $S3$ to train the agents and the detectors as this data was obtained first. Session $S1$ took place *after* $S3$, thus we evaluated agents and detectors on data from $S1$.

As shown in the second step of Fig. 10, SapiAgent autoencoders were trained in two ways: conventionally (CNN-AE conv., RNN-AE conv.) and using our approach (CNN-AE our app., RNN-AE our app.) described in section III-F using session $S3$ of the dataset for training.

Both SapiAgent and Bézier curves aim at mimicking trajectories of the $S1$ session. In other words: $S1$ serves as test data. Since $S1$ contains 5205 mouse trajectories, all agents generate the same amount of data. In particular, we consider the following agents: Bézier agents (see Section III-D) baseline and humanlike variants and four variants of SapiAgent using CNN and RNN autoencoders. The synthetic datasets were created based on the endpoints and trajectory length of the $S1$ subset of the SapiMouse dataset, as shown in the third step of Fig. 10.

The anomaly detectors are trained on human mouse trajectories of session $S3$. As shown in the fourth step of Fig. 10, training was followed by the evaluation step, when anomaly scores were calculated for both human ($S1$ session) and synthetic trajectories.

In order to simulate real-world applications, in which the mouse dynamics may be captured for a reasonably long time (e.g. one minute), the anomaly detectors make their decision based on a batch of 10 trajectories if not specified otherwise. All the trajectories in a batch belong to the same class, i.e., either all of them are synthetic or human trajectories. The anomaly scores of the trajectories of the same batch are averaged and this is considered as the anomaly score of the batch. The performance of anomaly detectors is reported in terms of AUC and EER for each agent separately.

B. PERFORMANCE OF THE ANOMALY DETECTORS

Our results are summarized in Table 2 and Table 3. Table 2 shows the performance of detectors for the examined agents in terms of AUC. In our case, lower AUC scores indicate better performance because our goal is to generate mouse trajectories that are difficult to be distinguished from human mouse trajectories. Table 3 shows the performance of detectors in

TABLE 2. Bot detection performances in terms of AUC for different anomaly detectors. The lower the AUC, the more successful is the agent in generating trajectories that are difficult to distinguish from true trajectories. Therefore, low values of AUC indicate better performance. The symbols ● and ○ show the results of statistical significance tests (binomial test at p-value 0.001) as follows: the two symbols following Bézier agent indicate whether the two version of our approach (i.e., "SapiAgent – our approach" with CNN-AE and RNN-AE) are significantly better than Bézier agent (●) or not (○). The symbol following "SapiAgent – conventional training" indicate whether the corresponding model trained with our approach is significantly better (●) or not (○).

	Bézier agent		SapiAgent – conventional training		SapiAgent – our approach	
Detector	Baseline	Humanlike	CNN-AE	RNN-AE	CNN-AE	RNN-AE
PCA	0.96 ●/●	0.75 ●/●	0.88 ●	0.99 ●	0.67	0.68
OCSVM	0.96 ●/●	0.74 ●/●	0.85 ●	0.99 ●	0.69	0.72
LOF	0.99 ●/●	0.92 ●/●	0.98 ●	1.00 ●	0.73	0.76
CBLOF	0.94 ●/●	0.78 ●/●	0.83 ●	0.98 ●	0.63	0.66
IForest	0.82 ●/●	0.62 ●/●	0.56 ○	0.86 ●	0.55	0.56
Feature bagging	0.98 ●/●	0.91 ●/●	0.95 ●	1.00 ●	0.70	0.79

TABLE 3. Bot detection performances in terms of EER for different anomaly detectors. The higher the error (EER), the more successful is the agent in generating trajectories that are difficult to distinguish from true trajectories. Therefore, high values of EER indicate better performance.

	Bézier agent		SapiAgent – conventional training		SapiAgent – our approach	
Detector	Baseline	Humanlike	CNN-AE	RNN-AE	CNN-AE	RNN-AE
PCA	0.11	0.31	0.19	0.04	0.37	0.37
OCSVM	0.12	0.31	0.23	0.04	0.35	0.34
LOF	0.05	0.16	0.07	0.00	0.32	0.30
CBLOF	0.13	0.29	0.23	0.05	0.40	0.38
IForest	0.25	0.41	0.46	0.21	0.47	0.46
Feature bagging	0.06	0.17	0.10	0.00	0.35	0.28

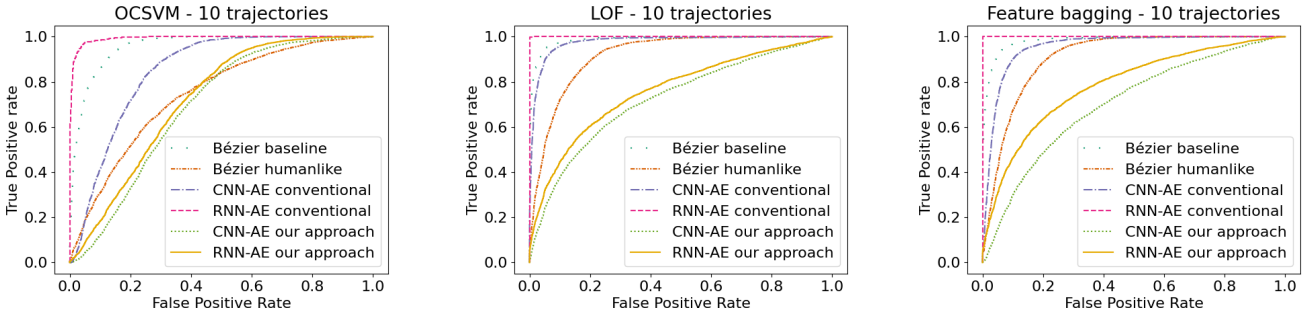

FIGURE 11. ROC curves of the examined agents using 10 trajectories for bot detection. OCSVM (left), LOF (middle), Feature bagging (right).

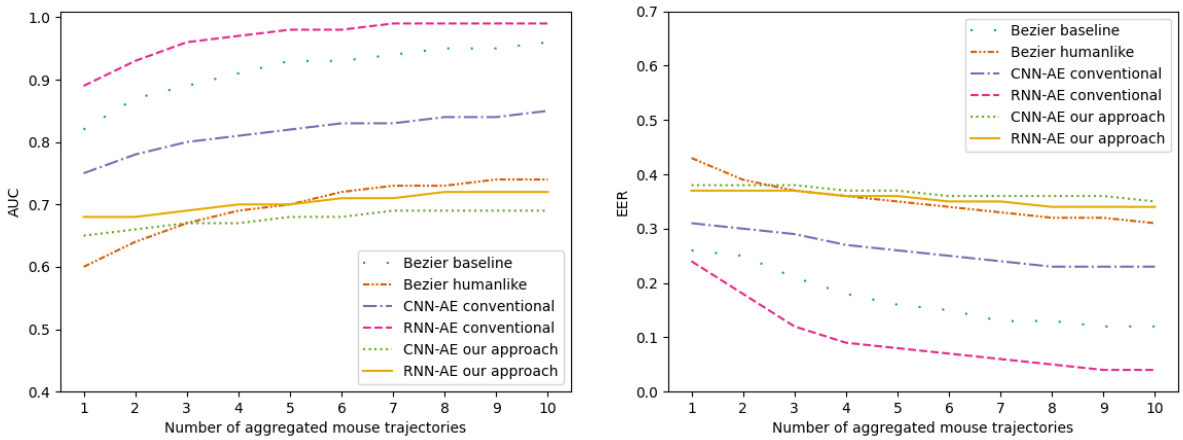

FIGURE 12. Performance of OCSVM detector in terms of AUC (left) and EER (right) for different number of aggregated trajectories.

terms of EER. The higher the EER, the more successful is the agent in generating trajectories that are similar to human trajectories. Therefore, higher EER indicates better performance.

Considering Table 2 and Table 3, it can be seen that SapiAgents trained with our approach clearly outperform both Bézier agents. Furthermore, SapiAgent trained with our approach is systematically better than the same agent with conventional training, indicating that the presented training strategy is indeed beneficial in order to construct models that are able to synthesize realistic trajectories. Last, but not least, we consider the two versions of our approach: we point out that SapiAgent with CNN-AE is systematically better than SapiAgent with RNN-AE both in terms of AUC and EER.

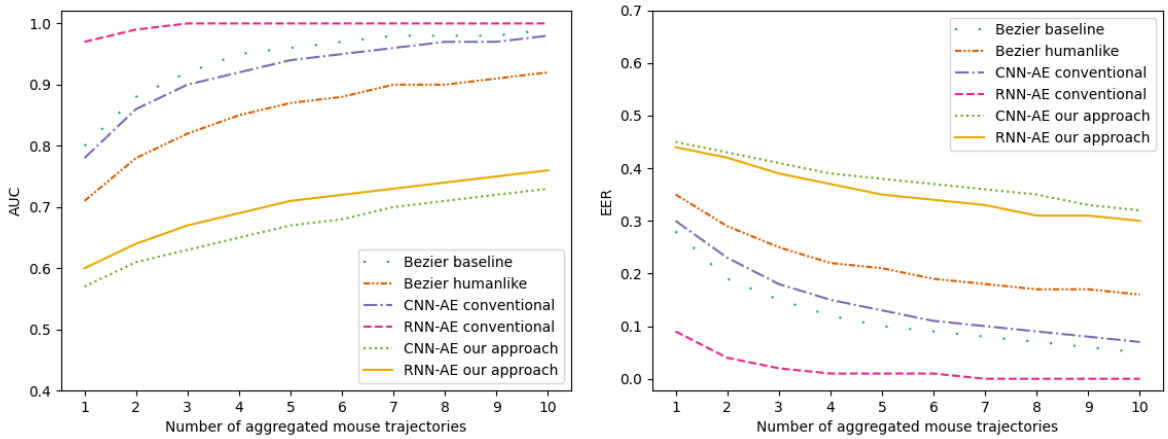

FIGURE 13. Performance of LOF detector in terms of AUC (left) and EER (right) for different numbers of aggregated trajectories.

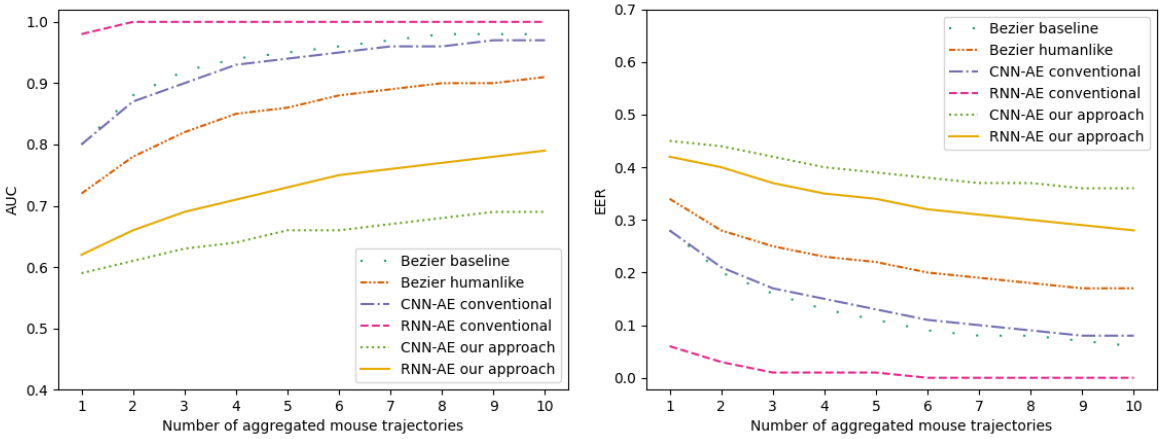

FIGURE 14. Performance of Feature bagging detector in terms of AUC (left) and EER (right) for different number of aggregated trajectories.

In order to assess whether the observed differences are statistically significant, we used Salzberg's binomial test [54]. We compared SapiAgent trained by our approach with each of the other agents, i.e., Bézier agents and conventionally trained SapiAgents in context of each examined detector. In order to perform the aforementioned binomial test, we need to calculate in how many cases are the two agents *different*, and in how many out of these cases is our approach the *winner*. In cases when the prediction of the detector was different for the two trajectories generated by the two compared agents,[6] we considered our approach the *winner* if the detector was unable to correctly identify whether the trajectory generated by our approach is synthetic or human. The results are shown by the symbols • and ◦ in Table 2 indicating whether our approach significantly outperforms Bézier agent and SapiAgent with conventional training. As one can see, our approach is almost always significantly better than its competitors.

Fig. 11 shows ROC curves for the examined agents for the case of using 10 trajectories for bot detection. The more realistic trajectories are generated by the agent, the weaker the performance of the anomaly detector.

[6]Note that each synthetic trajectory is generated based on a real trajectory, therefore, there is a natural correspondence between the synthetic trajectories generated by the two different agents.

C. EFFECT OF THE NUMBER OF TRAJECTORIES USED FOR DETECTION

The goal of this experiment is to show how the detection performance changes when multiple trajectories are available and thus the anomaly score may be estimated from more data. For simplicity, we only present results with the three best-performing anomaly detectors: OCSVM, LOF, and Feature bagging.

Fig. 12 shows the performance of the OCSVM detector for various number of trajectories. In the left, the AUC values are depicted as function of the number of trajectories ranging from one to ten. In the right, a similar performance curve is shown in terms of EER. It is clear that the detector distinguishes human trajectories from synthetic ones more accurately if more trajectories are available. In case if the detector's decision is based on a single trajectory only, humanlike Bézier agent is the most successful in ''misleading'' the detector. However, if several trajectories are available, the trajectories generated by the humanlike Bézier agent may be simpler to distinguish from real human

trajectories compared with the trajectories generated by our approach. Therefore, in real applications where the decision is based on several trajectories, our SapiAgent is stronger than the humanlike Bézier agent.

Performance curves for the LOF detector are shown in Fig. 13. Similar trends can be observed with this detector regarding the AUC performances. In this case, SapiAgents (our approach) perform much better than Bézier agents.

Fig. 14 depicts performance curves for the Feature bagging detector. Also in this case, SapiAgents (our approach) are the best.

D. TIME PERFORMANCE

Experiments were performed on a computer with 3.7 GHz Intel Core i9 processor having an NVIDIA, GeForce RTX 2080 SUPER graphical card (Ubuntu 20.04 OS). The software was implemented in Python 3.8.3 using the Keras library.

We measured the time needed to generate synthetic trajectories for SapiAgent and Bézier agents. SapiAgent needs less than a millisecond to generate a trajectory, regardless of the type of the autoencoder (CNN or RNN) and training method (conventional or our approach). In contrast, the Bézier agent needs more than 20 milliseconds to generate a humanlike trajectory. According to our observations, even the simple baseline Bézier agent was slightly slower than SapiAgent, as it took more than one millisecond to generate a simple (non-humanlike) Bézier trajectory.

V. CONCLUSION

In this paper, we introduced SapiAgent, a novel approach to generate synthetic mouse trajectories. SapiAgent employs deep autoencoders with a novel training strategy. We performed experiments in the context of bot detection on our new SapiMouse dataset which contains human mouse trajectories collected from 120 subjects.

Bot detection was formulated as an anomaly detection problem where the detector learns human behavior. Experimental results show that SapiAgent trained in a novel way generates more realistic mouse trajectories compared with conventional autoencoders and Bézier curves. In principle, SapiAgent may be used for generating other types of synthetic data such as two and three-dimensional gestures, therefore we envision its application in various other domains as well. As for future work at the technical level, we aim at investigating other deep neural network models in SapiAgent.

ACKNOWLEDGMENT

The authors would like to thank the editor and anonymous reviewers for their constructive comments.

REFERENCES

[1] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 49, pp. 433–460, Oct. 1950.
[2] G. Suchacka, A. Cabri, S. Rovetta, and F. Masulli, "Efficient on-the-fly web bot detection," *Knowl.-Based Syst.*, vol. 223, Jul. 2021, Art. no. 107074.
[3] M. Antal, N. Fejer, and K. Buza, "SapiMouse: Mouse dynamics-based user authentication using deep feature learning," in *Proc. IEEE 15th Int. Symp. Appl. Comput. Intell. Informat. (SACI)*, May 2021, pp. 61–66.
[4] M. Antal, L. Z. Szabó, and I. László, "Keystroke dynamics on Android platform," *Procedia Technol.*, vol. 19, pp. 820–826, Jan. 2015.
[5] K. Buza, "Person identification based on keystroke dynamics: Demo and open challenge," *CAiSE Forum*, vol. 4, pp. 161–168, Oct. 2016.
[6] H. Gamboa and A. Fred, "A behavioral biometric system based on human-computer interaction," *Proc. SPIE*, vol. 5404, pp. 381–392, Aug. 2004.
[7] C. Shen, Z. Cai, and X. Guan, "Continuous authentication for mouse dynamics: A pattern-growth approach," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, Boston, MA, USA, Jun. 2012, pp. 1–12.
[8] C. Feher, Y. Elovici, R. Moskovitch, L. Rokach, and A. Schclar, "User identity verification via mouse dynamics," *Inf. Sci.*, vol. 201, pp. 19–36, Oct. 2012.
[9] N. Zheng, A. Paloski, and H. Wang, "An efficient user verification system using angle-based mouse movement biometrics," *ACM Trans. Inf. Syst. Secur.*, vol. 18, no. 3, pp. 1–27, Apr. 2016.
[10] M. Antal and E. Egyed-Zsigmond, "Intrusion detection using mouse dynamics," *IET Biometrics*, vol. 8, no. 5, pp. 285–294, Sep. 2019.
[11] S. Almalki, N. Assery, and K. Roy, "An empirical evaluation of online continuous authentication and anomaly detection using mouse clickstream data analysis," *Appl. Sci.*, vol. 11, no. 13, p. 6083, Jun. 2021.
[12] M. Yildirim and E. Anarim, "Mitigating insider threat by profiling users based on mouse usage pattern: Ensemble learning and frequency domain analysis," *Int. J. Inf. Secur.*, vol. 1, pp. 1–3, May 2021.
[13] A. A. E. Ahmed and I. Traore, "A new biometric technology based on mouse dynamics," *IEEE Trans. Dependable Secure Comput.*, vol. 4, no. 3, pp. 165–179, Jul./Sep. 2007.
[14] J. J. Matthiesen and U. Brefeld, "Assessing user behavior by mouse movements," in *Proc. 22nd Int. Conf. Hum.-Comput. Interact.*, Copenhagen, Denmark, 2020, pp. 68–75.
[15] D. Qin, S. Fu, G. Amariucai, D. Qiao, and Y. Guan, "MAUSPAD: Mouse-based authentication using segmentation-based, progress-adjusted DTW," in *Proc. IEEE 19th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Guangzhou, China, Dec. 2020, pp. 425–433.
[16] P. Chong, Y. Elovici, and A. Binder, "User authentication based on mouse dynamics using deep neural networks: A comprehensive study," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1086–1101, 2020.
[17] M. Antal and N. Fejér, "Mouse dynamics based user recognition using deep learning," *Acta Univ. Sapientiae, Inf.*, vol. 12, no. 1, pp. 39–50, Jul. 2020.
[18] Y. X. Marcus Tan, A. Iacovazzi, I. Homoliak, Y. Elovici, and A. Binder, "Adversarial attacks on remote user authentication using behavioural mouse dynamics," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–10.
[19] D. Lagun, M. Ageev, Q. Guo, and E. Agichtein, "Discovering common motifs in cursor movement data for improving web search," in *Proc. 7th ACM Int. Conf. Web Search Data Mining*, New York, NY, USA, Feb. 2014, pp. 183–192.
[20] I. Arapakis and L. A. Leiva, "Predicting user engagement with direct displays using mouse cursor information," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Pisa, Italy, Jul. 2016, pp. 599–608.
[21] I. Arapakis and L. A. Leiva, "Learning efficient representations of mouse movements to predict user attention," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Virtual, China, Jul. 2020, pp. 1309–1318.
[22] Y. Liu, Y. Chen, J. Tang, J. Sun, M. Zhang, S. Ma, and X. Zhu, "Different users, different opinions: Predicting search satisfaction with mouse movement information," in *Proc. 38th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, Santiago, Chile, 2015, pp. 493–502.
[23] L. A. Leiva, I. Arapakis, and C. Iordanou, "My mouse, my rules: Privacy issues of behavioral user profiling via mouse tracking," in *Proc. Conf. Hum. Inf. Interact. Retr.*, Mar. 2021, pp. 51–61.
[24] N. Banholzer, S. Feuerriegel, E. Fleisch, G. F. Bauer, and T. Kowatsch, "Computer mouse movements as an indicator of work stress: Longitudinal observational field study," *J. Med. Internet Res.*, vol. 23, no. 4, e27121, pp. 1–11, 2021.
[25] A. Seelye, S. Hagler, N. Mattek, D. B. Howieson, K. Wild, H. H. Dodge, and J. A. Kaye, "Computer mouse movement patterns: A potential marker of mild cognitive impairment," *Alzheimer's Dementia, Diagnosis, Assessment Disease Monitor.*, vol. 1, no. 4, pp. 472–480, Dec. 2015.

[26] S. M. M. Mooij, M. E. J. Raijmakers, I. Dumontheil, N. Z. Kirkham, and H. L. J. Maas, "Error detection through mouse movement in an online adaptive learning environment," *J. Comput. Assist. Learn.*, vol. 37, no. 1, pp. 242–252, Feb. 2021.
[27] I. Pozzana and E. Ferrara, "Measuring bot and human behavioral dynamics," *Frontiers Phys.*, vol. 8, pp. 1–11, Apr. 2020.
[28] H.-K. Pao, K.-T. Chen, and H.-C. Chang, "Game bot detection via avatar trajectory analysis," *IEEE Trans. Comput. Intell. AI Games*, vol. 2, no. 3, pp. 162–175, Sep. 2010.
[29] E. M. Taranta, M. Maghoumi, C. R. Pittman, and J. J. LaViola, Jr., "A rapid prototyping approach to synthetic data generation for improved 2D gesture recognition," in *Proc. 29th Annu. Symp. User Interface Softw. Technol.*, Tokyo, Japan, 2016, pp. 873–885.
[30] S. Gianvecchio, Z. Wu, M. Xie, and H. Wang, "Battle of botcraft: Fighting bots in online games with human observational proofs," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, Chicago, IL, USA, 2009, pp. 256–268.
[31] Z. Chu, S. Gianvecchio, A. Koehl, H. Wang, and S. Jajodia, "Log or block: Detecting blog bots through behavioral biometrics," *Comput. Netw.*, vol. 57, no. 3, pp. 634–646, Feb. 2013.
[32] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[33] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*. [Online]. Available: http://arxiv.org/abs/1411.1784
[34] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.
[35] A. Acien, A. Morales, J. Fierrez, and R. Vera-Rodriguez, "BeCAPTCHA-mouse: Synthetic mouse trajectories and improved bot detection," 2021, *arXiv:2005.00890*. [Online]. Available: https://arxiv.org/abs/2005.00890
[36] J. Rodríguez-Ruiz, J. I. Mata-Sánchez, R. Monroy, O. Loyola-González, and A. López-Cuevas, "A one-class classification approach for bot detection on Twitter," *Comput. Secur.*, vol. 91, Apr. 2020, Art. no. 101715.
[37] S. Kudugunta and E. Ferrara, "Deep neural networks for bot detection," *Inf. Sci.*, vol. 467, pp. 312–322, Oct. 2018.
[38] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[39] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, New York, NY, USA, 2008, pp. 1096–1103.
[40] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Müller, "Deep learning for time series classification: A review," *Data Mining Knowl. Discovery*, vol. 33, no. 4, pp. 917–963, Jul. 2019.
[41] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Anchorage, Alaska, May 2017, pp. 1578–1585.
[42] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*. [Online]. Available: http://arxiv.org/abs/1412.3555
[43] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, no. 27, pp. 861–874, Oct. 2006.
[44] N. Sae-Bae and N. Memon, "Online signature verification on mobile devices," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 6, pp. 933–947, Jun. 2014.
[45] J. B. Camiña, M. A. Medina-Pérez, R. Monroy, O. Loyola-González, L. A. P. Villanueva, and L. C. G. Gurrola, "Bagging-RandomMiner: A one-class classifier for file access-based masquerade detection," *Mach. Vis. Appl.*, vol. 30, no. 5, pp. 959–974, Jul. 2019.
[46] M. E. Villa-Pérez, M. Á. Álvarez-Carmona, O. Loyola-González, M. A. Medina-Pérez, J. C. Velazco-Rossell, and K.-K.-R. Choo, "Semi-supervised anomaly detection algorithms: A comparative summary and future research directions," *Knowl.-Based Syst.*, vol. 218, Apr. 2021, Art. no. 106878.
[47] Y. Zhao, Z. Nasrullah, and Z. Li, "PyOD: A Python toolbox for scalable outlier detection," *J. Mach. Learn. Res.*, vol. 20, no. 96, pp. 1–7, Jan. 2019.
[48] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. W. Chang, "A novel anomaly detection scheme based on principal component classifier," Dept. Electr. Comput. Eng., Coral Gables, Miami Univ., Oxford, OH, USA, Tech. Rep., 2003.
[49] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.
[50] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, Jun. 2000.
[51] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recognit. Lett.*, vol. 24, nos. 9–10, pp. 1641–1650, 2003.
[52] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 413–422.
[53] A. Lazarevic and V. Kumar, "Feature bagging for outlier detection," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2005, pp. 157–166.
[54] S. L. Salzberg, "On comparing classifiers: Pitfalls to avoid and a recommended approach," *Data Mining Knowl. Discovery*, vol. 1, no. 3, pp. 317–328, 1997.

MARGIT ANTAL received the M.Sc. degree in computer science from Babes-Bolyai University, in 1991, and the Ph.D. degree in electronics and telecommunications from the Technical University of Cluj-Napoca, in 2006. Since 2016, she has been working as an Associate Professor with the Mathematics-Informatics Department, Sapientia Hungarian University of Transylvania. She published several research articles in international and national journals. Her research interests include biometrics, pattern recognition, and educational assessment. She serves as a Reviewer for *IET Biometrics* and IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY journals.

KRISZTIAN BUZA received the Diploma degree in computer science from Budapest University of Technology and Economics, in 2007, and the Ph.D. degree from the University of Hildesheim, in 2011. He is the coauthor of more than 50 publications, including the Best Paper of the IEEE Conference on Computational Science and Engineering, in 2010. His research interests include applied machine learning and data mining. He is a member of the program committee of international conferences, such as the Pacific-Asia Conference on Knowledge Discovery and Data Mining and the International Conference on Computational Collective Intelligence. He regularly serves as a Reviewer for renowned journals, such as *Neurocomputing*, *Knowledge-Based Systems*, and *Knowledge and Information Systems*.

NORBERT FEJER received the B.Sc. degree in computer science from the Sapientia Hungarian University of Transylvania, in 2020. He is currently pursuing the M.Sc. degree with a focus on software engineering, while also working at an international company in the field of building information modeling. He is the coauthor of several scientific papers and an award recipient at various student conferences. He is both a hardware and software aficionado. His research interests include time series analysis, deep learning, and GPU accelerated programming.