Person Authentication using Visual Representations of Keyboard Typing Dynamics

Ladislav Peška, Patrik Veselý, Tomáš Skopal
Faculty of Mathematics and Physics
Charles University
Prague, Czech Republic
ladislav.peska/patrik.vesely/tomas.skopal@matfyz.cuni.cz

Krisztian Buza
Department of Mathematics-Informatics
Sapientia Hungarian University of Transylvania
Targu Mures, Romania
buza@biointelligence.hu

***Abstract*—In this paper, we focus on the problem of user's authentication through typing dynamics patterns. We specifically focus on small-sized problems, where it is difficult to fully train corresponding machine (deep) learning algorithms from scratch. Instead, we propose a different approach based on the visualization of the typing patterns and subsequent usage of pre-trained feature extractors from the computer vision domain. We evaluated the approach on a publicly-available dataset and results indicate that this is a viable solution capable to improve over several baselines. Moreover, the proposed visual representation of the data contributes to the explainability of AI.**

***Index Terms*—Typing Dynamics, Visual Representations**

I. Introduction

Person (user) authentication may be necessary in various domains ranging from online services over daily situations (banking, traveling) to forensic investigations. Conventional techniques rely on passwords, presence of a physical object (identity card, passport, bank card, dynamic token generator, etc.) and biometrics such as fingerprints, iris-patterns, electroencephalography (EEG) or electrocardiography (ECG).

It was shown that the dynamics of keyboard typing (a.k.a. typing patterns or keystroke dynamics) is characteristic to particular users [1], [2] and most users are hardly able to mimic the dynamics of others [3]. Therefore, person authentication based on typing patterns is especially appealing in cases when the user is not necessarily interested to cooperate, such as testing the identity of students when taking exams in an online setting. Furthermore, in cases where high accuracy is required, such as online bank transactions involving significant amounts of money, person authentication based on typing patterns could be combined with conventional techniques, such as passwords and authentication codes sent to the user in SMS or e-mail.

Although the dynamics of typing is characteristic to users, even the same user cannot always type with the exactly same dynamics, see e.g. [2] for an illustration. In case of realistic applications with many users, it is extremely difficult, if not impossible, for human experts to identify patterns that can reliably distinguish users. Therefore, approaches based on machine learning are required.

This work was supported by Czech Science Foundation project GACR-22-21696S and by Charles University grant SVV-260588. Source codes are available from https://github.com/lpeska/KeystrokeDynamics2022

In the literature, person identification based on keystroke dynamics is often considered as a time series classification task, see e.g. [1], [2] and the references therein. Thus, in principle, various time series classifiers may be used, such as models based on dynamic time warping [4], or convolutional neural networks [5]. Although solutions based on deep learning became very popular in the last decade, we have to note that, in general, large amount of data is required to train such systems. In case of authentication systems, including systems based on keystroke dynamics, there may not be enough data to train a neural network. The problem of insufficient training data is, quite naturally, more severe for tasks with smaller overall volume of users (e.g., authentication of students for online examination) and also for users who have signed up recently (i.e., new user problem). Therefore, most popular solutions in the domain of keystroke dynamics are based on similarity search (based on dynamic time warping distance, DTW) and comparable techniques.

Nevertheless, an inherent difficulty of considering person identification as a time series classification task is that only a subset of the information, such as the duration of keystrokes, may be represented by time series in a straight forward way. However, it may be important to consider additional pieces of information as well, e.g., which keys were pressed.

In the last decade, computer vision techniques exhibited tremendous improvements mostly due to the application of deep learning techniques - see e.g. [6]. Also, in some occasions, pre-trained models of computer vision were successfully applied as feature extractors for artificial images representing initially non-visual data [7]–[9]. Therefore, we propose TYping VIsualiZER framework (*TYVIZER*) to utilize these advances in the domain of typing dynamics recognition. Specifically, *TYVIZER* generates artificial images to represent user's typing patterns, then applies selected feature extractors pre-trained on image domain to get embeddings of each image and finally applies some similarity based learning technique (e.g. k-nearest neighbors, kNN) for the desired prediction task.

The main advantage of our approach is that it does not need large annotated training datasets, which is crucial in many low-traffic applications. An additional advantage is that the generated visualizations may be interpretable by human experts, contributing towards the problem of explainability and

interpretability (of AI techniques) [10].

To sum up, main contributions of this paper are as follows:

- Proposed *TYVIZER* framework for user authentication through their typing dynamics. The framework operates in three steps: generating artificial visualizations of typing patterns, obtaining their descriptive embeddings and applying similarity-based learning techniques to deliver the desired prediction task.
- Visualizations of typing dynamics that may serve both as an input of the downstream learning pipeline, but also as interpretable representation of the underlying data.
- Evaluation w.r.t. user identification task on a publicly available dataset.

II. Related Work

Keystroke dynamics for person identification is a popular research topic. Recent approaches range from modified distance metrics [11] over time-frequency analysis [12] to one-class naive Bayes [13]. Please refer to [14] for a more complete overview. As the aforementioned examples show, a wide variety of methods have been developed for the classification of keystroke dynamics data.

Feature extraction from images is a well-studied task in computer vision [15]. Most of the recent approaches for feature extraction are based on state-of-the-art neural network models, e.g. [16]. Nonetheless, various shallow models can serve as image feature extractors as well, e.g. based on depicted colors [17], textures, or more complex low-level semantic descriptors [18]. We applied a range of both deep and shallow feature extractors to learn representations of the keystroke dynamics data and to identify the user based on those learned features.

There are some related works aiming to transform initially non-visual data to a visual representation and subsequently utilize pre-trained or fine-tuned models of computer vision to extract features from these visual representations [8], [9], [19], [20]. This was even envisioned as a possible direction for an universal data representation [7]. While artificial visual representations were tested on numerous problems, with the exception of [21], we are not aware of any related work applying them in the domain of keystroke dynamics. Nonetheless, the goal in [21] was to combine facial images with keystroke visualization. Specifically, only the duration of typing events were represented, not the particular pressed keys. In contrast, we also visualize additional information of keystrokes that can not be represented by univariate time-series in a straightforward way.

Obviously, one of the main challenges in the proposed framework is how to construct the artificial visualizations. Perhaps the simplest variant was described by Naz et al. [8] for univariate time-series such as ECG. Authors directly encoded the sequence of values as grey-scale colors and plot them along one axis of the resulting image. In order to adjust the varying length, time-series were trimmed to a fixed size and to adjust for square-shaped images required by the most of neural networks, the visualizations were chunked and stacked along the y-axis in a lines-first fashion. We based the proposed visualization on this generic approach, but extended it considerably. Specifically, the typing dynamics domain contains following additional challenges as compared to simple univariate time-series.

- Individual keystrokes have varying duration, which should be reflected in the resulting visualization.
- Pressing of different keys should be distinguished in visualizations as well.
- Multiple keys can be pressed at the same time (e.g., pressing a *SHIFT* key together with some alphabet key in order to capitalize it).
- Blank intervals between individual keystrokes may also provide relevant information and should be visualized.

We are not aware of any single related approach combining all of these challenges, but there are several related works that inspired the proposed visualization approach. First, transforming the length of a keystroke to the length of colorized objects representing each keystroke is rather straightforward extension of the original approach. One can for instance construct arbitrarily granular multivariate time-series, where for each time point and each key, the series will indicate, whether it is currently pressed or not. As such, gaps between individual keystrokes may be represented as blank spaces of appropriate length. However, representing pressed keys is more challenging. In some related works, authors directly encoded individual channels of a multivariate time-series as different components of RGB color model [19]. In our use-case, these channels might correspond to individual pressed keys. However, the volume of color channels is much smaller than the set of keys on a standard keyboard, therefore a substantial data reduction would have to be performed. Instead, we modified the idea from [22], where spatio-temporal coordinates of human skeleton joints were visually encoded for the purpose of gait recognition. Authors encoded coordinates of a 3D cube as corresponding values of red, green and blue channels of RGB color model. Then, for each time point, each joint was represented as a color corresponding to its current location in the 3D cube. We used a similar idea of overlaying RGB colors over a standard (2D) keyboard layout. As such, spatially adjacent keys would be rendered in a similar color.

III. *TYVIZER* framework

The proposed framework is comprised from three main components: typing patterns visualization, feature extraction and similarity-based classification. These components are plugged as a pipeline in that order.

A. Typing Patterns Visualization

Our typing pattern visualization is inspired by the work of Naz et al. [8]. At the conceptual level, we approached typing patterns as multivariate time-series, where each channel corresponds to a single key and represents binary information whether the key was pressed at each timepoint.[1] In the

[1]Note, however, that in implementation we utilized a more compact event-based representation from which the multivariate time-series can be derived.

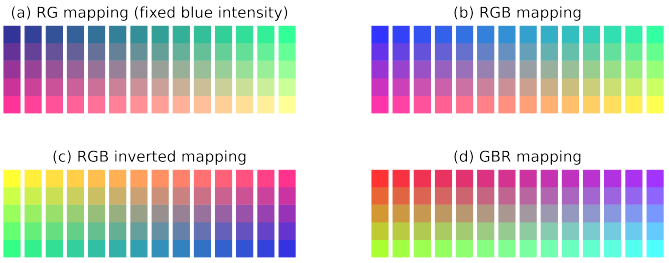

Fig. 1: Variants of color mappings for the keyboard layout.

visualization, x-axis represents time. Along the y-axis, Naz et al. [8] represented values of the (univariate) signal encoded as greyscale color blobs. In contrast, the signal is binary and multimodal in our case, so the same approach cannot be applied. Instead, we assigned colors to each channel (each key) and depict this color at each point corresponding to channel's positive value (i.e., forming rectangles of specified color, where its size corresponds to the key press duration). Nonetheless, we followed on [8] in two key aspects of the visualization. First, we fixed the scale of the time axis (in our case, the maximal represented value corresponds to the 99% quantile of the typing patterns duration). This was proposed to represent the time axis consistently along all typing patterns. Second, we break the time axis at several points (15 in our case) and stacked the resulting chunks along the y-axis. The main reason for this is that most feature extractors expect rectangular images on its input and such resizing would make the differences in keystroke duration virtually unrecognizable.

While assigning colors to each key, we adapted the idea of Elias et al. [22] and overlayed a color gradient over the 2D space corresponding to a standard keyboard layout. We experimented with the following four variants. In the first variant (denoted as *RG*), keyboard's y-axis is mapped to the red channel, keyboard's x-axis is mapped to the green channel, while the blue channel is kept fixed (Fig. 1a). In the second variant (*RGB*), the blue channel is also varied to amplify the differences between individual keys. Specifically, blue channel values linearly depends on both horizontal and vertical coordinates with the highest and lowest intensities in the top-left and bottom-right corners respectively (Fig. 1b). The remaining two variants are slight modifications of *RGB* approach, namely reversing the color slopes (*RGB inverted*, Fig. 1c) and shuffling the color channels (i.e., keyboard's y-axis is mapped to the green channel, keyboard's x-axis to the blue channel and diagonal to the red channel - see Fig. 1d).

We consider *RG* variant as the basic option, while *RGB* variant aims to explore whether more contrast can do better. With *RGB inverted* and *GBR* variants we want to explore the extent to which are feature extractors sensitive to particular color mappings. Apart from the colored variants, we also evaluated a baseline *monochromatic* variant, where all pressed keys have black color. Note that we only assigned colors to the basic alphanumeric keyboard keys and several control keys (shift, enter, backspace etc.). While this accounted for the vast majority of data, there were exceptions. In such cases, pressed keys with unknown coordinates were depicted as black similarly as in monochromatic variant.

So far we did not accounted for multiple keys being pressed simultaneously. While we can theoretically use some color arithmetics (e.g., depicting mean color of all pressed keys), this may introduce some unwanted similarities (i.e., mean of two key's colors equals a color of another key). Instead, we followed a different approach. Depicted rectangles' height is only 80% of the line height and they are partially transparent (alpha=0.8). If another key is pressed while previous was not released yet, the corresponding rectangle is moved up (20% of the line height), so rectangles corresponding to both previous and current key are at least partially visible. Some examples of the visualized typing patterns are depicted in Fig. 2.

B. Feature Extractors

In this paper, we consider three types of feature extractors: simple color models, extractors based on SIFT features [23] and several deep learning (DL) techniques. Each extractor receives an artificial image corresponding to one recorded typing pattern[2] and provides embedding (i.e., a feature vector) representing the image on its output. Let us now describe individual extractors used in this study.

Color-based extractors. Simple color-based extractors were commonly used in the pre-deep learning era for content-based image retrieval. Due to their usual simplicity, they worked well in some domains [17] and thanks to the hand-crafted nature these methods usually do not need any training data. As a representative of this class, we utilized *RGB Histogram* extractor. This method computes a pixel-wise histograms of the whole input image separately for each of R-G-B color channels. Then, all three histograms are concatenated and L_2-normalized. The size of the histogram is a hyperparameter of the model (32, 64 and 256 used in the experiments).

SIFT-based extractors. Scale-invariant feature transform (SIFT) [23] expand the color-based models with capability to extract also shapes and textures. Similarly as in the previous case, extracted features are pre-defined and thus no training is necessary. Nonetheless, as the volume of SIFT keypoints may vary across images, we utilize a post-processing as proposed in [18]. We denote this method as *VLAD* to remain consistent with the related work.

Deep learning based extractors. In the last decade, convolutional neural networks [24] (and more recently transformer-based architectures [25], [26]) have been widely used as state-of-the-art in many fields of computer vision and image retrieval. These deep networks need a lot of computation time and training data to learn, but such large datasets are available only for a handful of tasks. Therefore, transfer learning approach is often used. The method uses data from another domain to train a deep network and then use this pre-trained (optionally fine-tuned) network on the target domain.

[2]Note that for the performance improvements, mean image was subtracted from all visualizations before passing them to the extractor.

Fig. 2: Variants of visualized typing patterns for RG (top), RGB inverted (middle) and monochromatic color maps. For comparison, each column corresponds to the same typing pattern. The differences between patterns are clearly visible and understandable by human observer, therefore supporting explainability and scrutinability of the learning process.

As representatives of this class, we selected *EfficientNet* [24] (B2 size variant), Vision transformer (*ViT*; large variant) [25] and *ImageGPT* [26] (medium variant). All networks were pre-trained on ImageNet[3] w.r.t. image classification task. For EfficientNet and ViT we utilized activations of the last connected layer as feature vectors, but for ImageGPT, we used network's mid layer as suggested by its authors.

C. Typing Pattern Classification

While having the embeddings for each typing pattern visualization, one more step is necessary to perform person authentication or related tasks. In this paper, we focus on the problem of typing pattern classification, where individual users represent the target classes. In some sense, this can be considered as a worst-case scenario for person authentication, i.e., whether any of the existing users could successfully pretend to be someone else without being given away by their typing patterns. In order to perform typing pattern classification, we employed two similarity-based classifiers, namely k nearest neighbors (kNN) and its hubness-aware extension, hubness-weighted kNN (HWkNN).

Nearest neighbor classifiers are simple, intuitive and popular, there are theoretical results about their accuracy and error bounds [27]. However, nearest neighbors are affected by *bad hubs*. Roughly speaking, an instance is called a bad hub, if it appears surprisingly frequently as nearest neighbors of other instances, but its class label is different from the labels of those other instances. Bad hubs were shown to be responsible for a surprisingly large fraction of the total classification error. In order to reduce the detrimental effect of bad hubs, hubness-aware classifiers have been introduced, such as hubness-weighted k nearest neighbor or HWkNN for short [28]. Next, we review HWkNN.

We say that an instance x is a *bad neighbor* of another instance x^0 if (i) x is one of the k_H-nearest neighbors of x^0 and (ii) their class labels are different. In case of hubness-aware weighting [28], we first determine how frequently each instance x appears as a bad neighbor of other instances. This value is further standardized (denoted as $h_b(x)$) and used to determine the weight of each instance during the k-nearest neighbor classification.[4] Specifically, the weight of each training instance is $w(x) = e^{\,h_b(x)}$. We note that one additional advantage of the proposed approach is a seamless extension of the model with new target classes (i.e., new users) without the need of full model re-training.

In evaluation, we considered kNN with $k \in \{1, 3, 5, 7\}$, cosine distance and distance-weighted neighbors. For HWkNN, we also considered the number of neighbors $k \in \{1, 3, 5, 7\}$

[3]https://www.image-net.org/

[4]Note that the number of nearest neighbors in the final phase of kNN classification is not necessarily the same as the number of nearest neighbors used to determine the bad hubness score. The former is denoted by k, while the latter is denoted by k_H.

and cosine distance, while for hubness-aware weights we considered $k_H \in \{3, 5\}$.

IV. Evaluation

A. Dataset and Task

We used the dataset associated with the *Person Identification Challenge*[5]. As described in [29], keystroke dynamics data was collected in more than 500 typing sessions from 12 users. In each of the typing sessions, the users were asked to type the same short text. In particular, the users were asked to type the following text based on the English Wikipedia page about Neil Armstrong:

That's one small step for a man, one giant leap for mankind. Armstrong prepared his famous epigram on his own. In a post-flight press conference, he said that he decided on the words just prior to leaving the lunar module.

A JavaScript application was used to register keyboard events such as pressing and releasing of each key. We note that due to typing errors and corrections, the length of keystroke sequences vary. Therefore, this dataset is more realistic than many others. Moreover, due to the relatively long typing sessions, experiments on this data simulate online exams more realistically than other datasets containing short typing sessions, in which users typed a password or their own name.

B. Evaluation Protocol

For the purpose of evaluation, we kept the same protocol and data splits as described on the website of the Person Identification Challenge. Specifically, for all users, we kept the first 5 typing patterns as train set, while the remaining typing patterns were split into validation and test sets at random. The validation set was utilized to select best hyperparameters of each model, while we report on the results w.r.t. test set. As the proposed framework comprises from three steps, we will refer to its individual variants as a triplet of visualization variant, feature extractor and the classifier respectively. [6] Our approach was compared with several baselines based on dynamic time-warping (DTW) distance, namely 1NN, kNN, ECkNN and SUCCESS. 1NN stands for the simple 1-nearest neighbor classifier (w.r.t. DTW distance), while kNN and ECkNN denote classification through nearest neighbor regression approach, comparing all applicable pairs of classes, as described in [29]. SUCCESS refers to semi-supervised classification of time series based on the cluster-and-label paradigm [30].

C. Results

Figure 3 depicts overall results of the evaluation. To our surprise, more advanced feature extractors were significantly outperformed (p-values $< 5.2 \cdot 10^{-10}$ w.r.t. one-sided Fisher exact test) by a simple RGB histogram method. Also, *RG*+RGB histogram, *RGB*+RGB histogram and *GBR*+RGB histogram variants significantly outperformed all baselines (p-value <= 0.028). As for the other feature extractors, also ImageGPT and VLAD+HWkNN performed comparably with the baselines. Nonetheless, given the higher complexity of both extractors as well as the fact that their feature space is more dimensional, RGB histogram can be considered as a clear winner among the feature extractors.

We also observed a certain level of variation in results of individual color mappings. Overall, *RGB inverted* mapping provided inferior results as compared to *RGB* and *GBR* (p-value <= 0.01). Also, *RGB* mapping significantly outperformed *RG* one (p-value = 0.005). Therefore, some thoughts should be given to the particular color schemes in the future work. However, most notably, using any of the visualization variants provided considerable improvements over a *monochromatic* baseline (p-value <= 0.009), showing that adding information about pressed keys may provide valuable insights. Both kNN and HWkNN classifiers performed comparably in general and also the best achieved results were almost identical (i.e., *RGB*+RGB histogram+kNN vs. *RG*+RGB histogram+HWkNN).

V. Discussion and Conclusions

Overall, we can conclude that the proposed approach is a viable solution for user's authentication through their typing dynamics patterns. One of the main advantages of the proposed solution is that no large datasets are necessary to train the corresponding models, which is especially convenient for small-scale tasks such as student's identification during online exams.

Let us also note that the proposed visualization approach is easily extendable via additional modalities. For instance, keypress strength can be visualized e.g. via height of the colored rectangle, or its alpha-channel.

Rather surprising for us was the performance of simple RGB histogram method. In some sense, it actually works as a bag of stroked keys and due to the partial transparency of visualizations, it can also adjust for overlapping keystrokes. So, there is a solid background behind this approach. Even though, we expected state-of-the-art DL techniques to outperform it. We plan to further investigate, whether this is due to excessively large dissimilarities between train and task domains and e.g., whether some modifications to the visualization approach could help. Also, while the feature extractors can work as fully pre-trained models, we would like to explore the option to fine-tune them even on such small datasets.

Another limitation of our current work lies in the visualization approach itself. While we evaluated several variants of keys coloring, there are several other hyperparameters that were held fixed in the current experiments (e.g., length of visualized patterns, number of line breaks, levels of transparency). Therefore, we plan to experiment with the visualization approach more thoroughly in the future.

[5] http://biointelligence.hu/typing-challenge

[6] I.e., *GBR*+ViT+kNN denotes GBR visualization variant, ViT feature extractor and kNN classifier. Note that while giving general statements on some of the pipeline steps, others are left blank.

References

[1] M. Antal and L. Z. Szabó, "An evaluation of one-class and two-class classification algorithms for keystroke dynamics authentication