Paper

# Application of Recurrent Neural Networks for User Verification based on Keystroke Dynamics

Paweł Kobojek and Khalid Saeed

*Faculty of Mathematics and Information Sciences, Warsaw University of Technology, Warsaw, Poland*

**Abstract—Keystroke dynamics is one of the biometrics techniques that can be used for the verification of a human being. This work briefly introduces the history of biometrics and the state of the art in keystroke dynamics. Moreover, it presents an algorithm for human verification based on these data. In order to achieve that, authors' training and test sets were prepared and a reference dataset was used. The described algorithm is a classifier based on recurrent neural networks (LSTM and GRU). High accuracy without false positive errors as well as high scalability in terms of user count were chosen as goals. Some attempts were made to mitigate natural problems of the algorithm (e.g. generating artificial data). Experiments were performed with different network architectures. Authors assumed that keystroke dynamics data have sequence nature, which influenced their choice of classifier. They have achieved satisfying results, especially when it comes to false positive free setting.**

***Keywords—biometrics, GRU networks, keystroke dynamics, LSTM networks, recurrent neural networks, user verification.***

## 1. Introduction

The problem of verification is most often solved by assigning some kind of a password, which should only be known to a given user and consists of finite sequence of characters. When the user provides this password, a party responsible for confirmation of an identity may tell whether the user is whom he claims, he is (based on an assumption that only the real user knows the password). However, such approach is not free from drawbacks. For example, there has to be some kind of a mechanism to handle a situation in which the user forgets his or her password. Moreover, traditional passwords can be broken with brute force method if only attacking person has enough time and computation power (and, of course, there are no other protections against it). Also, if the user stores the password somewhere else than in his or her own brain it has to be somehow secured as well. Alternative to this method is using a biometrics-based security.

Keystroke dynamics is a field within behavioral biometrics, which concerns humans typing patterns on a keyboard. It turns out that the way a user writes on a keyboard is one of his or her unique characteristics. Back in 1980s, the first work was done in order to develop an algorithm which could identify a user based on this trait [1]. Many experiments were performed which have shown it is a good indicator of identity [1]–[4].

In order to describe mathematically a typing pattern we first need to acquire specific data from the user. This data consists of a timestamp of the moment of pressing and/or leaving the button. Next, different measurements out of this can be computed, e.g. [5]:

- dwell time – time between moment of pressing and moment of leaving the button,
- flight time – time between pressing (or leaving) subsequent keys.

A user who types the text can make mistakes, which means that vectors representing different samples may differ in length.

In the next step, data is passed to some kind of a model, which task is to answer the question whether examined user is the one who he claims to be. This model may be anomaly detection system or classifier. Popular approach is to use algorithms based on database of samples. In this case, new sample is compared with those already in database in order to find similarity.

The algorithm consists of two parts: way of acquiring data along with features extraction and a model, which verifies/identifies the sample. Designing new solutions may affect both of these modules.

The accuracy may be influenced even by a way of acquiring data from a user as well as it nature. In the most basic approach, sample describing the user simply consists of timestamps mentioned earlier (from which dwell/flight time is computed). Besides this, it is sometimes useful to measure other values, e.g. eye motion. Humans often either follow their fingers with their eyes or look straight at the monitor. Taking this behavior into consideration may enhance classification accuracy. Mobile devices are supplied with additional sensors like gyroscope or accelerometer. Information from these sensors was proven useful [6]–[8]. [9] shows thoughts about authorization specific for mobile devices with focus on using biometrics techniques including keystroke dynamics. In addition to all these information, there is also meaningful signal in errors made by the user along with the way they correct them (e.g. by using *delete* vs. *backspace*).

For some applications, using only keystroke dynamics may not be accurate enough because of strict regulations. Even

in such situation, it can be used as a valuable support for traditional data. Such approaches increase security and combined accuracy may be high enough to be used even in healthcare [10]. Such methods may be extended by even more biometrics techniques, e.g. face recognition [11].

As it was stated before, keystroke dynamics data may also find applications when it comes to user identification. In this paper this problem is reduced to of multiclass classification, i.e. each user is represented by a class. In this case, we usually have limited user count. This work focuses on verification because in a problem it tries to solve the user is already identified by his or her email address. Identification problem was broadly described in [12] along with proposed algorithm.

### *1.1. State of the Art Algorithms*

Looking at the problem as an anomaly detection problem, statistical methods based on some kind of distance are often used. In standard approach, having some data set (let us treat every sample as a vector) we find its center, which is also a vector. This is a training phase. In testing phase on the other hand, the task is to tell that whether given vector (test sample) is an anomaly or not. In order to answer this question distance between center and test sample has to be computed. The distance may be classic Euclidean distance as well as something more sophisticated i.e. Manhattan distance. This simple algorithm can be further modified e.g. by applying distance norming. In *Filtered Manhattan* algorithm, after finding the center at first all samples, which are too far from it, are removed and then new center point is computed. Similar group of algorithms are those based on $k$-nearest neighbors idea. In this case, instead of designating a center point and comparing input with it, we find $k$ (in particular, $k = 1$) closest, in terms of defined distance, samples. In this case, usually an *anomaly score* as distance from their center is computed. Another interesting approach is using fuzzy sets. In such sets each object belongs (to some degree) to ranges. The anomaly score is then computed as an average lack of belonging. The approach, which is most similar to the idea presented in this work, is probably one-class SVM. However, such a classifier is trained only on positive class (in opposition to this work's algorithm).

More thorough description of those algorithms (with references to exhaustive descriptions) can be found in [13]. Results of [13] are benchmark for results achieved by the algorithm described in this paper.

When it comes to multiclass classification with keystroke dynamics, the multiple classifiers were tested: HMM, SVM, $k$-nearest neighbors, and neural networks [14]. Presented algorithm does not solve multiclass classification problem. Nevertheless, with slight modification it could be trained for such problems as well. On the other hand, algorithms mentioned in this paragraph could be used as binary classifiers and replace the proposed one.

### *1.2. Algorithm Evaluation Methods*

An important thing to consider is the evaluation of proposed algorithms. Let us introduce the following terms:

- True Positive Rate (TPR) or hit-rate $\frac{TP}{TP+FN}$,
- False Positive Rate (FPR) $\frac{FP}{FP+TN}$, informs about the probability of accepting an impostor,

where: $TP$ – number of true positives, $TN$ – number of true negatives, $FP$ – number of false positives, $FN$ – number of false negatives.

Besides standard accuracy or error measure, when it comes to keystroke dynamics (and also in other fields of biometrics) two more measures are often used to evaluate algorithms:

- Equal Error Rate (EER) – value for a threshold in which FPR and miss rate1 – TPR are equal,
- Zero-miss rate – FPR value for which TPR = 1 (no false positive errors).

Both these values can be easily read from ROC curve. Figure 1 shows sample ROC curve along with mentioned points marked on it. The values can be read from $x$ axis of these points.

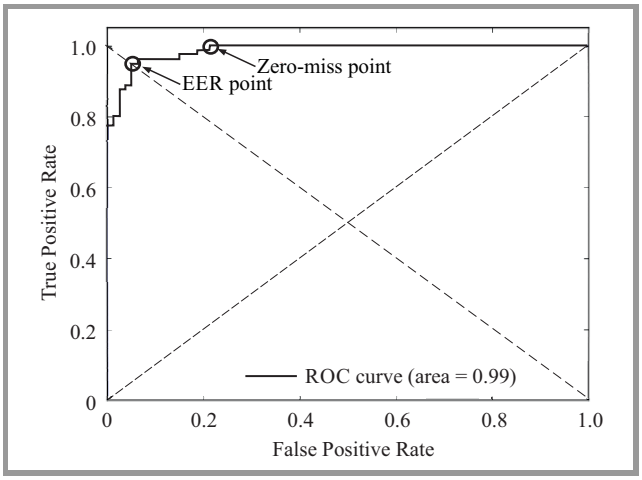


***Fig. 1.*** Sample ROC curve (from the results of this research) with EER and zero-miss points marked.

### *1.3. Problems with Algorithms and Authors' Proposition*

Some of the mentioned algorithms are based on assumption that we have some database of patterns for a user. In the moment when a new sample appears, we need to go through the whole database and find similarities ($k$-nearest neighbors is an example of this approach). Note that the keystroke dynamics is a behavioral feature, thus it changes with time more than physiological traits. When it comes to keystroke dynamics problem, maintaining a static database for a given user may end up with gradually decreasing accuracy. One of the solution, which comes to mind, is adding new samples. Unfortunately, the side effect of this approach

is the growing need of memory of such a system. This disadvantage, combined with a big number of users may result in memory consumption as the main drawback. When it comes to multiclass classification, there is a need to add new class with each new user.

Another problem of these algorithms is the fact, that they treat input as a vector. Intuitively it seems that numbers representing the sample from a human are more like a sequence, i.e. there is some relation between them. As usual in machine learning problems – it is hidden and unknown. Because mentioned algorithms do not treat data as a sequence, some information natural to them must be encoded artificially. As an example, let us say that the user has mistaken and then corrected the errors. In case of sequence, such information is directly encoded in its length, because errors and corrections require more keystrokes.

Algorithms, which are described and compared in [13], reach relatively low accuracy when it comes to the situation where the threshold was set to avoid false positive errors (zero-miss). The best presented algorithm in this setting (*k*-nearest neighbors with Mahalanobis distance) achieved 0.468 zero-miss rate. In a problem of access control, this would mean the situation in which probability of rejecting a genuine user is close to 0.5.

A problem for which the presented algorithm could be useful is creating a centralized system serving authentication based on a way the user types his or her email address. Thus, the email along with the biological characteristic of a human being would be the only ID in the Internet and necessity of using multiple long passwords would disappear. Services of such a system could be used by external services which could supply it with sufficient information, i.e. keystroke dynamics data plus email address and in return get the information whether the user is verified or not.

Having all this in mind, the presented algorithm is a subject of more constraints. First, it should authenticate potentially everyone in the Internet. Given the enormous number of Internet users (almost 3 billion in 2014 [15]), infinite scalability in terms of user count have to be assumed, which cannot be constrained by the algorithm.

Such a system could potentially be used to grant the access to many services using the same one identification way. The most important feature of such a system is definitely securing resources from unauthorized people. Ignoring this problem would result not only in not solving the problem in which a user has one password to many accounts and someone has accessed it, but could even make it worse. It seems better to reject a genuine user from time to time than to accept the attacking one. The designed algorithm should thus focus on minimizing (ideally eliminating) false positive errors, which means accepting wrong user. Eliminating such errors should be a goal even at the cost of big drop in accuracy.

The proposed algorithm was designed with all that features in mind. Thus, the most important goals are scalability in terms of user count and high accuracy without false positive errors.

# 2. The Algorithm

## *2.1. General Idea and Motivation*

The standard approach in a keystroke dynamics based verification is using anomaly detectors. Presented approach is different. It uses a binary classifier (recurrent neural networks). Data from the genuine user are positive and from the other people – negative. A big disadvantage, which may appear in readers mind, is the requirement of negative data for training phase. Some thoughts about it along with ways of mitigating this issue were described in latter sections.

In order to choose good classifier it is worth to consider the nature of a problem. First property of the examined data is they do not seem to be a vector describing some physical phenomenon or object (like images, where every element of a vector contains information about specific pixel). As it was stated before, it is assumed that data has a sequence nature. It is worth noting though, that there are no strict proofs of that. However, for some people it seems intuitive, because of (among other reasons) keyboard arrangement. This assumption has influenced the choice of a classifier.

## *2.2. Recurrent Neural Networks*

Due to assumed sequence nature of input data authors have decided to use recurrent neural networks. These networks naturally operate on sequences. Plain recurrent neural networks are very simple (compared to other neural network architectures) models. They differ from feed forward networks in the way of processing input – here it is processed in a step-by-step manner. At step $t$ the network receives $x_t$ as input and having knowledge about state from last step $h_{t-1}$ it computes its output according to the formulas (1) and (2). $W_{hh}$, $W_{xh}$ and $W_{hy}$ are matrices of network parameters.

$$h_t = tanh(W_{hh} \cdot h_{t-1} + W_{xh} \cdot x_t) \tag{1}$$

$$y = W_{hy} \cdot h_t \tag{2}$$

Unfortunately, in its simplest form, recurrent neural networks are very hard to train due to the problem known as exploding or vanishing gradient [16], [17]. However, there are modified architectures of recurrent neural networks, which solve this problem.

## *2.3. LSTM*

Long Short-Term Memory (LSTM) networks along with training algorithm were proposed in 1997 in the paper [16] in order to solve mentioned problem of vanishing gradient. They are successfully used in many fields, especially when data is sequential, e.g. natural language processing, speech recognition, machine translation, image captioning [18] or even bioinformatics [19].

Core idea behind LSTM network is inclusion of a so-called cell state. It is a vector, which simply stores information, thus it is a kind of memory. This vector is passed through computation steps – modified or not. At each step the network can write or remove some information to/from the

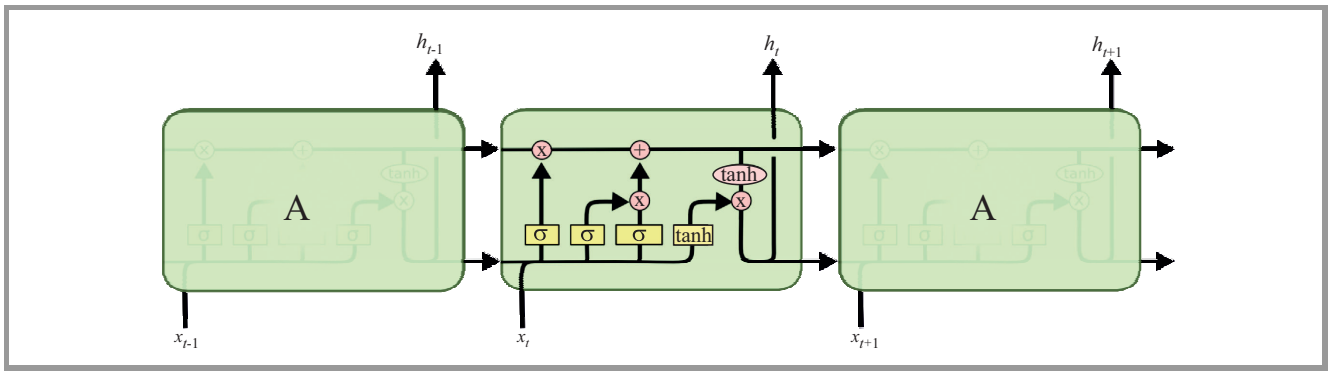


***Fig. 2.*** LSTM computation in time.

memory. It is done using so-called gates. At each computation step the input and the cell state from previous step first go to the forget gate. The way it operates is very simple – it is a plain sigmoidal layer known from neural networks. To be more precise, its output is computed with formula:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3}$$

where:

- $\sigma$ – sigmoid function $\left(\sigma(x) = \frac{1}{1+e^{-x}}\right)$,
- $W_f, b_f$ - weight matrix and bias of forget gate,
- $h_{t-1}$ - output from previous step,
- $x_t$ - input in current step.

The vector resulting from this gate tells how much information should be forgotten and how much should be remembered. Degree of this "forgetting" is controlled by the value of sigmoid function which is in range $[0,1]$: 0 means forget everything, 1 means remember everything.

Next is the input gate. Input data along with the output from the previous step are used twice in this gate: in the sigmoid layer (similar to forget gate) and in another layer with hyperbolic tangent as activation. Results of these layers are going to be used in order to create a vector, which is then added to the cell state. This step is described by formulas:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{4}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C). \tag{5}$$

After computing these 3 values they can be used to update the cell state. This computation is shown by equation:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t, \tag{6}$$

where: $C_t$ is the cell state at the moment $t$. $f_t$ is computed from Eq. (3), $i_t$ from Eq. (4), $\tilde{C}_t$ from Eq. (5). Current memory value is first multiplied by an output from forget gate which potentially erases some information and then new information is added. The final LSTM result from the current step is computed not only from the input and the state but also from the cell state. This is described by following formulas:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{7}$$

$$h_t = o_t \cdot tanh(C_t). \tag{8}$$

Final result of this computation is some real value from range $[0,1]$ if the network is last layer of the model. If it is inner layer then it returns the whole sequence containing all values of $h$ computed in "for" loop. If network is the last layer then its output is compared with the threshold, which determines the final class.

Figure 2 shows how the mentioned computations are performed in time [20].

### *2.4. GRU*

Gated Recurrent Units (GRUs) were introduced in 2014 [21]. They are a similar to LSTM. What is different is that instead of two gates – forget and input gate, GRUs have only one – update gate. Another difference and simplification lies in fact, that GRUs do not have separate memory (cell state). The memory is associated with the state from previous step. Network computation is described by formulas:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]), \tag{9}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]), \tag{10}$$

$$\tilde{h}_t = tanh(W \cdot [r_t \cdot h_{t-1}, x_t]), \tag{11}$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t. \tag{12}$$

Merged gates output is $z_t$ vector. It is used for both forgetting and remembering.

As in LSTM, output is a real number from range $[0,1]$ if network is last layer of a model. Otherwise, it returns sequence consisting of every $h$ values computed in "for" loop. The final model output is then compared with the threshold in order to determine the class.

Figure 3 presents diagram with GRU cell [20].

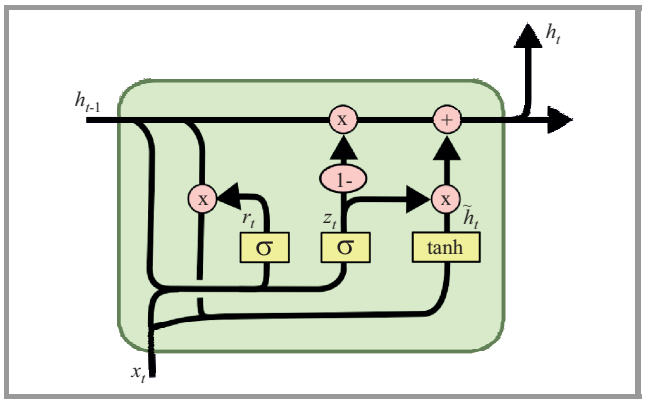


***Fig. 3.*** GRU diagram.

### *2.5. Training*

LSTM and GRU were trained by standard Back Propagation Through Time (BPTT) algorithm [22]. It is used to compute cost function derivative required for optimization algorithm, i.e. Adam optimizer in this case [23]. Networks were trained for 100 epochs, and cost is described by function:

$$C = -\frac{1}{n}\sum_{x}\left[y\ln a + (1-y)\ln(1-a)\right], \qquad (13)$$

where: $n$ – samples count, $x$ – single element, $y$ – expected label for $x$, $a$ – actual label for $x$.
Figure 4 shows an example loss over iterations graph.

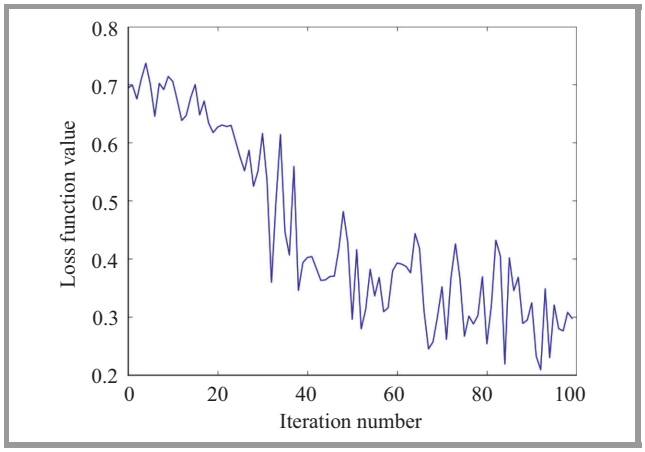


***Fig. 4.*** An example graph showing loss value over iterations.

### *2.6. Small Training Set and Lack of Negative Data*

As it was mentioned earlier, serious drawback of the presented algorithm is a need for negative data during the training phase. In real life applications, requiring user to type his address several times is already an inconvenience for him. Forcing other people to type, this address would be even harder. One of possible solutions could be using neural network as one-class classifier. In such a case, the model would be trained only on positive data, which is easier to acquire. Such methods are successfully applied for SVM classifiers [13] to solve verification problem. Unfortunately, the conducted experiments had not shown any good results with recurrent neural networks.
Another approach, which was tested, is generation of artificial negative data. From every positive sample authors got a negative by adding a random (with normal distribution, several values of standard deviation were tested) noise to it. The classifier was trained on positive and artificially created negative data and then evaluated only on real data.
Another problem is the size of a training set. Deep neural networks are models in which there are enormous number of parameters, which has to be adjusted during the training, thus they require big amount of data. Training for much iteration with small dataset tends to overfit. Unfortunately, in this case asking the user to type an address several hundred times would clearly be impractical. In this work, authors' dataset has only over a dozen samples for each user. Because of that, the authors had to apply regularization techniques in order to avoid overfitting.
It is worth mention that the challenges may not be a problem in some real applications. Large email services, e.g. Google Gmail, have (or might have) access to a large amount of data about address typing. Positive data could come from successful login or typing own email. Negative data on the other hand could be extracted from other people who type email of a given user in order to send him a message. In such a case, there would be no need to generate artificial data.

### *2.7. False Positive Errors Minimalization*

As it was mentioned earlier, one of the challenges for the designed algorithm is the minimalization of false positive errors.
A standard approach is to select the acceptance thresholds. By increasing its value, number of samples classified as positive should decrease. Hopefully, first to drop will be samples classified as positive with low likelihood, which are potentially false positive errors. The idea to eliminate such errors is then to increase the threshold until every false positive error is gone on training set.
Unfortunately, networks tend to classify with a very high likelihood. Thus selecting the threshold, which eliminates unwanted errors will definitely decrease total accuracy, because it has to be pretty high, so many genuine samples are rejected.
The question arouses – why LSTM and GRU models tend to return high numbers even if they mistake? These models are very sophisticated and are based on strong type of neural networks (so-called deep neural networks) and are used for high dimensional problems like image recognition [24]. Compared to such problems, the presented task has much smaller dimensionality, which is probably a reason why network overfits.
There are different methods of regularization, which help to mitigate the problem of overfitting [25]. One of them, used in this work is dropout.

### *2.8. Dropout*

Dropout was introduced by researchers from University of Toronto [26] as a regularization technique for deep neural networks. The idea behind it is to remove some random group of neurons along with connections during training phases. Since those random groups are different at each step, it prevents neurons from learning to copy other neurons, which in turn makes them better at approximating desired output. This is often compared to ensemble models, which is training several models and making them vote. Dropout is fully described in [26].

### *2.9. Tested Architectures*

Several neural network architecture were trained and tested as classifiers. Architecture which was satisfying on chosen test, dataset turned out to be too weak for benchmark dataset (see results in Section 4) so it had to be adjusted. The tested architectures are shown in Figs. 5–8, where:

- LSTM – single LSTM cell,
- GRU – single GRU cell,
- Dropout – adding regularization using dropout,
- LR – sigmoid layer,
- Embedding – mapping value to vector space.

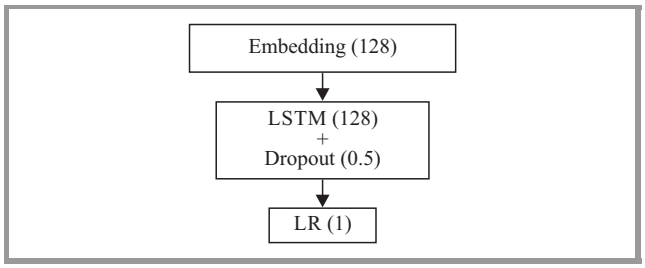


***Fig. 5.*** Network embedding and one LSTM layer.

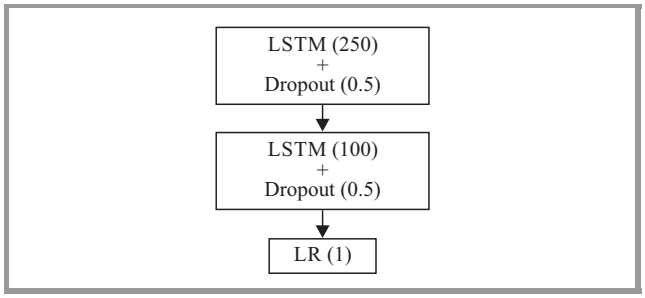


***Fig. 6.*** Network structure with two LSTM layers.

Moreover, networks in Figs. 5 and 6 were trained with and without dropout, which turned out to have major influence on results. Process of architecture selection was empirical, which means that many architectures have been tested and hyperparameters based on results were adjusted.

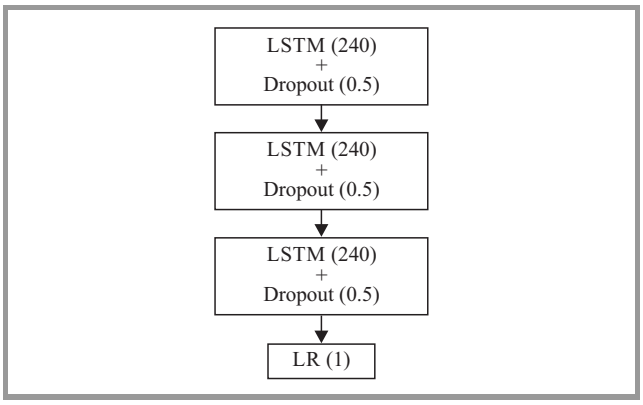


***Fig. 7.*** Network with three LSTM layers (it was tested only on benchmark set).

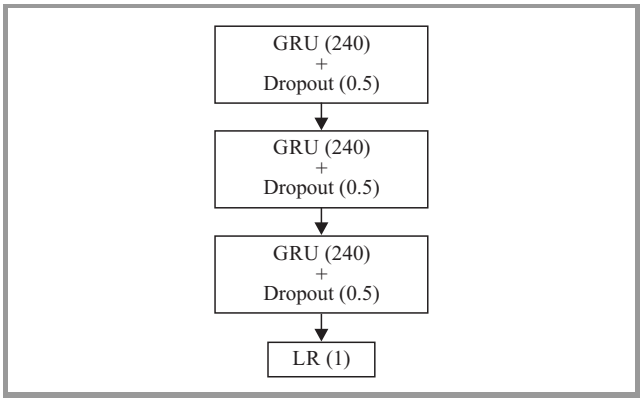


***Fig. 8.*** Network with three GRU layers (it was tested only on benchmark set).

## 3. Datasets

### *3.1. Authors' Dataset*

In order to conduct experiments we prepared own dataset. To achieve this task a website gathering keystroke data was used. The recording software was implemented as a student project by Albert Wolant. Every user was asked to type his address 5 times and type other addresses once. Nine students have taken part in this experiments, but due to low quality of some data (e.g. copy-paste method), samples batch from 3 people was rejected. Final dataset consisted of data from 6 people ranging from 12 to 20 samples each. It is worth mentioning that samples include information about mistakes made by typists.

### *3.2. Benchmark Dataset*

Because the dataset described in previous section has only few samples, in addition a benchmark dataset available on the website was used [27]. This site provides exhaustive description of this set and acquiring method. This dataset was used by its authors to compare anomaly detectors [13]. It consists of data gathered from 51 typists, each has typed the same phrase 400 times. Even though it was created

for anomaly detectors, it turned out to be very valuable for presented algorithm. Because users typed the same phrase (to be more precise – same password), it is possible to create a dataset for each user containing this user's samples as positive data and all other samples as negative. The problem here is that in such case there are 50 times as many negative as positive data. Classifier trained on such data will most likely tend to classify new samples as negative. It would also give false impression of high accuracy [28]. Because only small part of test data would be positive, so just classifying everything as negative gives high accuracy. With this problem in mind, authors decided to balance positive and negative data. For every user, only 400 random samples from other users were chosen as negative. Eventually, 51 sets were obtained (one for every user) containing 800 samples each: 400 hundred positive, 400 negative.
One drawback of this set is that it is cleared from mistakenly typed samples. Thus, there is no information about frequency of errors done by a user.
It should be clearly stated, that since this set was not used in its direct shape and that evaluation methods were different from those used by its authors results of this research cannot be directly compared to original results achieved by authors in [13].

# 4. Experiments and Results

For both (own and benchmark) datasets the different but not disjoint sets of models have been tested. In addition, a scenario in which negative data is artificial was also included in test process. In this case was tested and compared with *k*-nearest neighbors' classifier.
In order to achieve repeatability of experiments, they were all performed with the same random number generator seed.

## *4.1. Authors' Dataset*

Due to small size of this set, a non-standard evaluation has been employed (by "standard" we mean dividing set for training, validation and test sets). This is why leave-n-out-cross-validation with $n = 1$ [29] is used. In this validation with ($n = 1$) the one model for every dataset element was trained. This selected element acts as one element test set. The model is trained on the rest of the set. This means, having $n$ elements in a set, $n$ models should be trained. Then the model computation on this selected element is performed. The total accuracy is an average computed from all those results.

### 4.1.1. Model with one LSTM Cell and Embedding

For this model, the total accuracy for all users reached only 58%. Table 1 shows results for all users. Note that because of validation type, single sample is included multiple times here. FP-free thresholds cell shows score when acceptance thresholds were chosen to eliminate false positive errors on training set. Because total accuracy was low, we got rid of embedding layer in favor of another LSTM cells.

Table 1
Results for all users

| | Accepted | Rejected |
|---|---|---|
| Genuine user | 359 | 138 |
| Impostor | 164 | 72 |
| | Threshold 0.5 | FP-free thresholds |
| Accuracy | 0.58 | 0.55 |

Table 2
Results for all users

| | Accepted | Rejected |
|---|---|---|
| Genuine user | 456 | 41 |
| Impostor | 75 | 161 |
| | Threshold 0.5 | FP-free threshold |
| Accuracy | 0.85 (0.88) | 0.52 (0.8) |

### 4.1.2. Model with two LSTM Cells

This model was tested in two versions – with and without dropout. Table 2 shows its accuracy. The numbers in parentheses relate to models with dropout.
The accuracy of LSTM model with 2 cells is satisfying. The dropout's influence on results is clear, especially if false positive free thresholds are used. By only adding dropout, the accuracy raised from 0.52 to 0.8. Unfortunately, the size of this dataset is small and the evaluation method had negative impact on results. Hence, it is hard to judge the algorithm quality by this data only. Despite this problem, those results hold some value, because in real life applications of verification based on keystroke dynamics usually only have small datasets are available.

## *4.2. Benchmark Set – Limited Data*

Benchmark dataset, as it was described earlier, contains more data (in terms of both user count and samples per user). It is thus more reliable when it comes to the algorithm evaluation. Samples in this dataset contain more than just dwell-time. However, because only dwell-time was recorded in author's dataset, first study was performed only including this measure.

### 4.2.1. Model with two LSTM Cells

This is the same model, which turned out to be good enough for our custom dataset. This time, only version with dropout was tested as it achieved better results. Results are presented in Tables 3 and 4.
Unfortunately, model with two LSTM cells, even though it performed well on small dataset, does not give satisfying

Table 3
Results on benchmark dataset for two LSTM cells

| | Threshold 0.5 | FP-free |
|---|---|---|
| Average accuracy | 0.759 | 0.59 |
| Maximum | 0.975 | 0.994 |
| Minimum | 0.4875 | 0.5 |
| Standard deviation | 0.101 | 0.1344 |

Table 4
Results on benchmark dataset for two LSTM cells

| | |
|---|---|
| EER | 0.227 (0.094) |
| Zero-miss rate | 0.764 (0.221) |

results on benchmark dataset. We have then made it more complex by adding one more LSTM cell as well as increasing the total number of neurons. This model has achieved results presented in Tables 5 and 6.

Table 5
Accuracy on benchmark dataset for model with three LSTM cells

| | Threshold 0.5 | FP-free |
|---|---|---|
| Average accuracy | 0.764 | 0.61 |
| Maximum | 0.9875 | 0.9875 |
| Minimum | 0.5187 | 0.5 |
| Standard deviation | 0.114 | 0.1399 |

Table 6
Benchmark dataset results for three model with three LSTM layers

| | |
|---|---|
| EER | 0.219 (0.106) |
| Zero-miss rate | 0.747 (0.221) |

#### 4.2.2. Model with Three LSTM Cells

In this case results are only slightly better than previous. It seems like simply increasing complexity of this model is not enough. Therefore, we decided to try again, swapping LSTM cells with GRU equivalents.

#### 4.2.3. Model with Three GRU Cells

Results of experiments with this model (Tables 7 and 8) are comparable with those achieved on own dataset. However, if we compare it with results achieved by author's dataset, proposed algorithm would be placed 8th in terms of EER and 6th when it comes to zero-miss rate. Especially zero-miss rate is high which we would like to minimize.

### *4.3. Benchmark Dataset – All Data*

As it was mentioned earlier, original dataset contains more than just dwell-time. Authors decided to try testing proposed algorithm using all data provided by the dataset. Achieved results are presented in Table 9. Only measures, which are easily comparable with algorithms presented in [13] are shown.

Table 7
Accuracy on the benchmark dataset for model with three GRU cells

| | Threshold 0.5 | FP-free |
|---|---|---|
| Average accuracy | 0.83 | 0.68 |
| Maximum | 0.9875 | 0.9875 |
| Minimum | 0.5 | 0.5 |
| Standard deviation | 0.099 | 0.1397 |

Table 8
Benchmark dataset results for three model with three GRU layers

| | |
|---|---|
| EER | 0.150 (0.087) |
| Zero-miss rate | 0.613 (0.260) |

Table 9
Models results on benchmark dataset with all data

| Model | EER | Zero-miss rate |
|---|---|---|
| LSTM 2 cells | **0.136 (0.176)** | 0.379 (0.314) |
| LSTM 3 cells | 0.165 (0.191) | **0.333 (0.282)** |
| GRU 3 cells | 0.224 (0.319) | 0.389 (0.325) |

The results are significantly better than those which only included dwell time. In addition, the best model here is the one built with 3 LSTM cells.

### *4.4. Artificially Generated Data*

One of the most important disadvantages of the algorithm is the need of negative data during training. In this work the method of artificial generation of negative data based on positive samples have been tested. Proposed algorithm was to *k*-nearest neighbor classifier. In total 417 different combinations of distance definition, number of neighbors and standard deviation of the normal distribution used for negative data generation was tested. It is worth noting that the algorithm is used as a classifier and not as an anomaly detector. The algorithm presented in this work has achieved results shown in Tables 10 and 11.

Table 10
Accuracy for data with artificial negative samples

| Model | Accuracy threshold 0.5 | Accuracy FP-free |
|---|---|---|
| LSTM 2 cells | 0.622 (0.094) | 0.562 (0.094) |
| LSTM 3 cells | 0.633 (0.106) | 0.561 (0.100) |
| GRU 3 cells | 0.629 (0.093) | 0.707 (0.101) |

Table 11
EER and zero-miss rate for data with artificial negative samples

| Model | EER | Zero-miss rate |
|---|---|---|
| LSTM 2 cells | 0.441 (0.336) | 0.598 (0.303) |
| LSTM 3 cells | 0.768 (0.219) | 0.592 (0.291) |
| GRU 3 cells | 0.527 (0.402) | 0.597 (0.334) |

For a comparison, best result of $k$-nearest neighbor was for $k = 1$ and dice distance and it was 58%.

A similar experiment was also conducted on author's dataset. In this case, the best $k$-nearest neighbor algorithm accuracy was 87% while neural networks achieved 100% accuracy. However, this cannot be used as an argument for high accuracy of the model, because experiments on the larger dataset have not confirmed such high accuracy. The question is, however, why artificial data has actually increased accuracy (using only real data 80% accuracy was achieved). The reason is probably that without artificial data, the dataset was imbalanced in terms of negative to positive samples ratio, which made the network to be more eager to answer with class, which was overrepresented in the training set. Since we generated one artificial sample for each positive, this gave us the perfectly balanced set.

Unfortunately, presented method of generating artificial data turned out to be not very effective. The accuracy is significantly lower compared to training with only real (positive and negative) data. However, as was expected, recurrent neural networks performed generally better than $k$-nearest neighbor classifier.

# 5. Conclusions and Algorithm Evaluation

Compared to results from authors of the benchmark dataset, achieved best result (EER 0.136) would be on 7th place in terms of EER for total 14 places. It is equal to the one achieved by filtered Manhattan algorithm, yet its standard deviation is better: 0.083 compared to 0.176. The presented algorithm performed better than other neural networks tested by authors.

However, zero-miss rate is more interesting. The best result achieved by authors of [13] is 0.468. In this research the best result is 0.333, a lot better, however, those results cannot be directly compared, because of different nature of algorithms – this work shows binary classifier, authors of the mentioned paper tested anomaly detectors, different training method and different evaluation method. Despite that, presented algorithm performed well in terms of zero-miss rate and lets us recommended it as valuable when it comes to such a case. It is worth reminding, that high accuracy without false positive errors was one of the main objectives of the designed algorithm. It is worth noting, that the big leap in accuracy was caused by including additional data (both flight and dwell time). As it turned out, this had more influence than classifier architecture.

Another important feature, which was required from the algorithm, was scalability in terms of user count. Because for each user we train separate classifier, there is no problem with too many similar classes – each model is binary classifier trained for a given individual. Because neural networks are based on parametric models, they require the access to samples database only in the training phase. Thus, the increasing sample count for the user will not increase the size of the model when it comes to memory usage. Each model requires about 3.5 MB. This seems reasonable size (1 million users would require 3.5 TB of disk space). Therefore, the objective of unconstrained scalability is achieved.

We have stated the hypothesis that input data are sequences, and not just vectors and that a valuable signal comes from this information. Because recurrent neural networks are the natural choice for sequences processing, it could have direct impact on the accuracy. However, we cannot say with strong belief that this statement is more than just hypothesis. If our results were significantly better then others, it would be the strong evidence for it.

Unfortunately, the algorithm is not free from flaws. Most of them, however, were known at the beginning of the work. We have tried to mitigate the problem by generating artificial negative data. Results were admittedly better than $k$-nearest neighbors, yet they are noticeably worse than those achieved by the same model with access to real negative data. Perhaps, there is a method of generating better data, but further studies are needed here.

Certain drawback of the algorithm is how much time it requires to be fully trained. Neural networks are complicated models with a large number of parameters, so it requires time to adjust them. On a typical desktop 2.9 GHz Intel Core i5 CPU training and evaluating most sophisticated models took about 8 hours, which means about 10 minutes per user (there were 51 typists in benchmark dataset). Even if this seems quick, it is very long time compared to many anomaly detectors, which often only require one pass through the database. 10 minutes is a big issue for so-called continuous verification, i.e. constant monitoring of keyboard usage in order to detect impostors. However, training time directly depends on the dataset size. In this case, for each user we had 640 samples. Acquiring this number of samples (with assumption that exactly half of them are negative) requires time and has to be finished before training. Having said that, 10 minutes becomes less significant. Nevertheless, full training and evaluation requires 8 hours, which makes the hyperparameters adjustment a tougher task.

Paper [30] presents LSTM networks used as anomaly detectors. By incorporating this idea, we could use the same evaluation method as it was used in [13], which introduced benchmark dataset. This would allow us direct comparison. Moreover, it would solve the problem of negative data

requirement. Results achieved in the mentioned work give hope for increase of usability if those models for keystroke dynamics in the future.

## References


[1] R. Gaines, W. Lisowski, S. J. Press, and N. Shapiro, "Authentication by keystroke timing: some preliminary results", R-2526-NSF RAND Report, RAND Corporation, Santa Monica, CA, USA, May 1980.

[2] F. Monrose and A. D. Rubin, "Keystroke dynamics as a biometric for authentication", *Future Gener. Comp. Syst*., vol. 16, pp. 351–359, 2000.

[3] R. Joyce and G. Gupta, "Identity authorization based on keystroke latencies", *Commun. of the ACM*, vol. 33, no. 2, pp. 168–176, 1990.

[4] D. Mahar, R. Napier, M. Wagner, W. Laverty, R. Henderson, and M. Hiron, "Optimizing digraph-latency based biometric typist verification systems: Inter and intra typists differences in digraph latency distributions", *Int. J. Human-Comp. Stud*., vol. 43, no. 4, pp. 579–592, 1995.

[5] P. R. Dholi and K. P. Chaudhari, "Typing Pattern Recognition Using Keystroke Dynamics", in *Mobile Communication and Power Engineering*, Vi. V. Das and Y. Chaba, Eds. *Communications in Computer and Information Science*, vol. 296, pp. 275–280. Springer, 2012.

[6] G. Ho, "TapDynamics: Strengthening User Authentication on Mobile Phones with Keystroke Dynamics", Tech. Rep., Stanford University, San Francisco, CA, USA, 2014.

[7] Y. Deng and Y. Zhong, "Keystroke dynamics advances for mobile devices using deep neural network", in *Recent Advances in User Authentication Using Keystroke Dynamics Biometrics*, Y. Zhong and Y. Deng, Eds. Science Gate Publishing, 2015, vol. 2, pp. 59–70.

[8] C. Giuffrida, K. Majdanik, M. Conti, and H. Bos, "I sensed it was you: Authenticating mobile users with sensor-enhanced keystroke dynamics", in *Detection of Intrusions and Malware, and Vulnerability Assessment*, S. Dietrich, Ed. *LNCS*, vol. 8550, pp. 92–111. Springer, 2014 (doi: 10.07/978/3-319-08509-8_6).

[9] M. Rogowski, K. Saeed, M. Rybnik, M. Tabędzki, and M. Adamski, "User authentication for mobile devices", in *Computer Information Systems and Industrial Management*, K. Saeed, R. Chaki, A. Cortesi, and S. Wierzchoń, Eds. *LNCS*, vol. 8104, pp. 47–58. Springer, 2013.

[10] T. Bhattasali and K. Saeed, "Two Factor Remote Authentication in Healthcare", in *Proc. 3rd Int. Conf. Advan. in Comput., Commun. & Inform. ICACCI 2014*, Delhi, India, 2014, pp. 380–386.

[11] T. Bhattasali, K. Saeed, N. Chaki, and R. Chaki, "Bio-authentication for layered remote health monitor framework", *J. Medical Inform. & Technol.*, vol. 23, no. 1, pp. 131–139, 2014.

[12] M. Rybnik, P. Panasiuk, K. Saeed, and M. Rogowski, "Advances in the keystroke dynamics: the practical impact of database quality", in *Computer Information Systems and Industrial Management*, A. Cortesi, N. Chaki, K. Saeed, and S. Wierzchoń, Eds. *LNCS*, vol. 7564, pp. 203–214. Springer, 2012 (doi: 101007/978-3-642-33260-9_17).

[13] K. S. Killourhy and R. A. Maxion, "Comparing anomaly detectors for keystroke dynamics", in *Proc. 39th Annual IEEE/IFIP Int. Conf. Dependable Syst. & Netw. DSN 2009*, Lisbon, Portugal, 2009, pp. 125–134.

[14] Y. Deng and Y. Zhong, "Keystroke Dynamics User Authentication Based on Gaussian Mixture Model and Deep Belief Nets", *ISRN Sig. Process*., vol. 2013, article ID 565183, 2013 (doi: 10.1155/2013/565183).

[15] "ICT Facts and Figures" 2005, 2010, 2014, Telecommunication Development Bureau, International Telecommunication Union (ITU), 24 May 2015.

[16] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Computat*., vol. 9, no. 8, pp. 1735–1780, 1997.

[17] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult", *IEEE Trans. on Neural Netw*., vol. 5, no. 2, pp. 157–166, 1994.

[18] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent Neural Network Regularization", in *Proc. Int. Conf. on Learn. Representat. ICLR 2015* San Diego, CA, USA, 2015 [Online]. Available: https://arxiv.org/pdf/1409.2329.pdf

[19] S. K. Sønderby, C. K. Sønderby, H. Nielsen, and O. Winther, "Convolutional LSTM Networks for subcellular localization of proteins", in *Algorithms for Computational Biology*, A.-H. Dediu, F. Hernández-Quiroz, C. Martín-Vide, and D. A. Rosenblueth, Eds. *LNCS*, vol. 9199, pp. 68–80. Springer, 2015.

[20] C. Olah, "Understanding LSTM Networks", Aug. 27, 2015 [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (accessed on Aug. 9, 2016)

[21] K. Cho, F. Bougares, H. Schwenk, D. Bahdanau, and Y. Bengio, "Learning phrase representations using RNN Encoder-decoder for statistical machine translation", in *Proc. of the Conf. on Empir. Methods in Natural Language Process. EMNLP 2014*, Doha, Qatar, 2014, pp. 1724–1734.

[22] P. J. Werbos, "Backpropagation through time: What it does and how to do it", *Proc. of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1998.

[23] D. P. Kingma and J. L. Ba, "Adam: A Method for stochastic optimization", in *Proc. Int. Conf. on Learn. Representat. ICLR 2015* San Diego, CA, USA, 2015 [Online]. Available: https://arxiv.org/pdf/1412.6980v8.pdf

[24] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition", *IEEE Trans. Pattern Anal. & Mach. Intellig.*, vol. 31, no. 5, pp. 855–868, 2009.

[25] A. Ng, "Feature selection, L1 vs. L2 regularization, and rotational invariance", in *Proc. 21st Int. Conf. on Machine Learn. ICML'04*, Banff, Canada, 2004.

[26] N. Srivastava *et al.*, "Dropout: A simple way to prevent neural networks from overfitting", *J. of Mach. Learn. Res.*, vol. 15, no. 1, 1929–1958, 2014.

[27] K. Killourhy and R. Maxion, "Keystroke Dynamics – Benchmark Data Set", Carnegie Mellon University, Pittsburgh, PA, USA [Online]. Available: http://www.cs.cmu.edu/∼keystroke/ (accessed on May 27, 2016).

[28] H. He and E. A. Garcia, "Learning from imbalanced data", *IEEE Trans. on Knowl. & Data Engin.*, vol. 21, no. 9, pp. 1263–1284, 2009.

[29] J. Schneider, "Cross Validation", Carnegie Mellon University, Pittsburgh, PA, USA [Online]. Available: https://www.cs.cmu.edu/∼schneide/tut5/node42.html (accessed on May 27, 2016).

[30] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series", in *Proc. of European Symp. on Artif. Neural Netw., Computat. Intellig. and Machine Learning ESANN 2015*, Bruges, Belgium, 2015.



**Paweł Kobojek** received the M.Sc.degree in the field of Computer Science from Warsaw University of Technology in 2016. His scientific interests are mainly focused on Bioinformatics and Artificial Intelligence and its applications in biometrics and language processing.

E-mail: PawelKobojek@gmail.com
Faculty of Mathematics and Information Sciences
Warsaw University of Technology
Koszykowa st 75
00-662 Warsaw, Poland

**Khalid Saeed** is a B.Sc., M.Sc., Ph.D. and D.Sc. degrees holder. He is a Computer Science full professor (Biometrics, Image Analysis and Processing) at Białystok University of Technology. He also works with the Faculty of Mathematics and Information Sciences, Warsaw University of Technology. He was with AGH Krakow in 2008–2014. He has published more than 200 publications and edited 27 books, journals and conference proceedings, 11 text and reference books. He received 19 academic awards. Mr. Khalid Saeed is a member of more than 15 editorial boards of international journals and conferences. He is an IEEE Senior Member and has been selected as IEEE Distinguished Speaker for 2011–2017. He is the editor-in-chief of International Journal of Biometrics with Inderscience Publishers.

E-mail: k.saeed@mini.pw.edu.pl
Faculty of Mathematics and Information Sciences
Warsaw University of Technology
Koszykowa st 75
00-662 Warsaw, Poland