Free-Text Keystroke Dynamics for User Authentication

Jianwei Li, Han-Chih Chang, and Mark Stamp

Abstract In this research, we consider the problem of verifying user identity based on keystroke dynamics obtained from free-text. We employ a novel feature engineering method that generates image-like transition matrices. For this image-like feature, a convolution neural network (CNN) with cutout achieves the best results. A hybrid model consisting of a CNN and a recurrent neural network (RNN) is also shown to outperform previous research in this field.

1 Introduction

User authentication is a critically important task in cybersecurity. Password based authentication is widely used, as are various biometrics. Examples of popular biometrics include fingerprint, facial recognition, and iris scan. However, all of these authentication methods suffer from some problems. For example, passwords can often be guessed and are sometimes stolen, and most biometric systems require special hardware [14, 17, 23]. Moreover, research has shown, for example, that the accuracy of face and fingerprint recognition on the elderly is lower than for young people [13]. Thus, an authentication method that can resolve some of these issues is desirable.

Intuitively, it would seem to be difficult to mimic someone's typing behavior to a high degree of precision. Thus, patterns hidden in typing behavior in the form of keystroke dynamics might serve as a strong biometric. One advantage of a keystroke dynamics based authentication scheme is that it requires no specialized hardware. In addition, such a scheme can provide a non-intrusive means of continuous or ongoing authentication, which can be viewed as a form or intrusion detection. Coursera, an online learning website, currently employs typing characteristics as part of its login system [15].

J. Li · H.-C. Chang · M. Stamp (✉)
San Jose State University, San Jose, CA, USA
e-mail: jianwei.li@sjsu.edu; han-chih.chang@sjsu.edu; mark.stamp@sjsu.edu

M. Stamp et al. (eds.), *Artificial Intelligence for Cybersecurity*, Advances in Information Security 54, https://doi.org/10.1007/978-3-030-97087-1_15

Research into keystroke dynamics began about 20 years ago [16]. However, early results in this field were not impressive. Most of the existing research in keystroke dynamics has focused on fixed-text typing behavior, which is viewed as one-time authentication [2, 5, 12, 14, 23]. Compared with fixed-text keystroke dynamics, the free-text case presents some additional challenges. First, the number of useful features may differ among input sequences. Second, the optimal length of a keystroke sequence for analysis is a factor that must be considered—a longer sequence is slower to process and might include more noise, while a shorter sequence may lack sufficient distinguishing characteristics. Moreover, for free-text keystroke sequences, it is more challenging to extract an effective pattern, thus the robustness of any solution is a concern.

In this paper, we consider the free-text keystroke dynamics-based authentication problem. For this problem, we propose and analyze a unique feature engineering technique. Specifically, we organize features into an image-like transition matrix with multiple channels, where each row and column represents a key on the keyboard, with the depth corresponding to different categories of features. Then a convolutional neural network (CNN) model with cutout regularization is trained on this engineered feature. To better capture the sequential nature of the problem, we also consider a hybrid model using our CNN approach in combination with a gated recurrent unit (GRU) network. We evaluate these two models on open free-text keystroke datasets and compare the results with previous work. We carefully consider the effect of different lengths of keystroke sequences and other parameters on the performance of our models.

The contribution of this paper include the following:

- A new feature engineering method that organizes features as an image-like matrix for free-text keystroke dynamics-based authentication.
- An analysis of cutout regularization as a step in the image analysis process.
- A careful analysis of various hyperparameters, including the length of keystroke sequence in our models.

The remainder of this paper is organized as follows. Section 1 introduces the basic concept of keystroke dynamics-based authentication, and we outline our general approach to the problem. In Sect. 2, we discuss background topics, including the learning techniques employed and the datasets we have used. Section 2 also provides a discussion of relevant previous work. Section 3 describes the features that we use and, in particular, we discuss the feature engineering strategy that we employ to prepare the input data for our continuous classification models. Then, in Sect. 4, we elaborate on the architectures of the various models considered in this paper, and we discuss the hyperparameter tuning process. Section 5 includes our experiments and analysis of the results. Finally, Sect. 6 provides a conclusion and points to possible directions for future work.

Table 1 Use cases for keystroke dynamics-based systems

Text	Scenario	Precision	Recall	Input length
Fixed	One-time authentication	High	High	Short
Free	Intrusive detection	Low	High	Long
Either	Identification	High	Low	Either

2 Background

Authentication is the process that allows a machine to verify the identity of a user. By the nature of the problem, authentication is a classification task. Keystroke dynamics is one of many techniques that have been considered for authentication. One advantage of keystroke dynamics is that such an approach requires no special hardware.

Precision and recall are two metrics used to evaluate classification models. Precision is the fraction of true positive instances among those classified as positive, while recall is the fraction of true positive instances that are correctly classified as such. Table 1 lists some examples of use cases, along with the general degree of precision and recall that typically must be attained in a useful system. Depending on the scenario, too many false positives (i.e., low precision) can render an IDS impractical, but an IDS must detect intrusions (high recall) or it has clearly failed to perform adequately. On the other hand, in the identification problem, we must be confident that our identification is correct (high precision), even if we fail to identify subjects in a number of cases (low recall).

We note in passing that even if the precision and recall are both high, practical usage scenarios for keystroke dynamics based systems may be limited by the length of the keystroke sequence required for analysis. In cases where a short keystroke sequence suffices, the technique will be more widely applicable.

For a usage scenario, consider password-protected user accounts. Keystroke dynamics would provide a second line of defense in such an authentication system. In a two-factor authentication system, an attacker would need to also accurately mimic a users tying habits. Note that the second “factor” (i.e., keystroke dynamics) is transparent from a user’s perspective—the keystroke-related biometric information is collected passively, and requires no additional actions from a user beyond typing his or her password.

Even in cases where the length of the keystroke sequence must be relatively long in order to achieve the necessary accuracy, keystroke dynamics systems could still be useful. For example, suppose that a user needs to reset their password for a high-security application, such as an online bank account. Most such systems require the user to answer a “secret” security question or multiple such questions. It can be difficult for users to remember the answers to security questions, and the answers themselves (e.g., “mother’s maiden name”) are often not secret. Replacing these question with a keystroke dynamics system would free the user from the need to remember answers, as the user would simply need to type a sufficient number of characters in the user's usual typing mode.

From the use-case point of view, keystroke dynamics-based systems can be classified into those for which long input sequences are acceptable, and those for which short input sequences are essential. We can also classify keystroke dynamics systems according to whether they are based on fixed-text or free-text. In this paper, we only consider free-text.

2.1 Related Work

Previously, most work in keystroke dynamics was based on fixed-text, but recently more attention has been paid to free-text keystroke analysis. There are two commonly used free-text keystroke datasets, which we refer to as the Buffalo dataset [20] and the Clarkson II dataset [22]. We discuss these datasets in more detail in Sect. 2.2. Yan et al. [20] introduced the Buffalo dataset, which they use to evaluate a Gaussian mixture model (GMM) proposed by Hayreddin et al. [4]. The best EER obtained is 0.01. Their experiments are limited to keystroke data generated using the same keyboard. In our research, we evaluate our models on the entire Buffalo dataset, which includes different keyboards.

Pilsung et al. [9] divide the keyboard into three areas, left, right, and space, which correspond to the keys that are typically typed by the left hand (L), right hand (R), and thumbs (S), respectively. In this way, the time-based features extracted from different adjacent keystroke pairs fall into eight categories, which are denoted as L-L, L-R, R-R, R-L, R-S, S-R, L-S, S-L. Then they compute average time-based histogram over each group and concatenate these values to form a feature vector. In this way, the free-text keystroke sequence is embedded into a vector of fixed length eight, which can then be used in different detection models. However, their method fails to preserve most of the sequential information that is available in keystrokes.

To improve the performance of authentication systems based on free-text, Junhong et al. [10] propose a novel user-adaptive feature extraction method to capture unique typing pattern behind keystroke sequences. The method consists of ranking time-based features, and splitting all of these features into eight categories based on the rank order. Similar to the method proposed in [9], they calculate the average time-based feature of each category as a single feature value and concatenate these features to form a vector. Their experiments show that the method significantly improves performance, as compared with the method in [9]. However, they are still discarding a significant amount of the information available in the raw keystroke dynamics data.

Eduard et al. [17] explore the use of multi-layer perceptrons (MLP) for keystroke based authentication. Their model considers time-based information between different keys separately, and does not aggregate information from the entire keystroke sequence. The performance appears to be relatively poor.

Bernardi et al. [3] propose a feature extraction model to capture user input patterns; additional related work by these authors can be found in [21]. In [3], the authors test the impact of different numbers of layers in various deep learning networks and compared the effectiveness of deep networks with classical machine learning methods. They attain a highest accuracy of 99.9% using an MLP with nine hidden layers. However, their architectures are limited to feed-forward fully-connected layers, and better results require a large number of hidden layers. Also, the dataset used in their research is different from that used in our research, and thus the results are not directly comparable.

Kobojek et al. [11] uses an RNN-based model for classification based on keystroke data. They make use of keystroke sequential data. They achieve a best EER that is relatively high at 13.6%.

Influenced by the work in [11], Xiaofeng et al. [14] divide continuous keystroke dynamics sequences into keystroke subsequences of a fixed length and extracts time-based features from each subsequence. These features are then organized into a fixed-length sequence, and the resulting data is fed to a complex model consisting of a combined CNN and RNN. They consider an overlapping sliding window, and the they use a majority vote system to further improve the accuracy. They best EERs of 2.67 and 6.61% over a pair of open free-text keystroke datasets. In our research, we propose a new architecture that is inspired by the model in [14].

2.2 Datasets

In this paper, we evaluate various models based on two open-source free-text keystroke dynamics datasets. The two datasets we consider are from Clarkson University [22] and SUNY Buffalo [20]. Next, we discuss these datasets.

2.2.1 Buffalo Keystroke Dataset

The Buffalo free-text keystroke dataset was collected by researchers at SUN Buffalo from 148 research subjects. In this dataset, the subjects were asked to finish two typing tasks in a laboratory. For the first task, participants transcribed Steve Jobs' Stanford commencement speech, which was split into three parts. The second task consisted of responses to several free-text questions. The interval between the two sessions was 28 days. Additionally, only 75 of the subjects completed both typing tasks with the same keyboard, while the remaining 73 subjects typed using three different keyboards across three sessions.

The Buffalo dataset includes relatively limited information. Specifically, the key that was pressed, along with timestamps for the key-down time and the key-up events. The average number of keystrokes in the three sessions exceeded 17,000 for each subject. Additionally, some of the participants used keyboards with different key layouts to input text information. This dataset also provides gender information for each subject.

2.2.2 Clarkson II Keystroke Dataset

The Clarkson II keystroke dataset is a popular free-text keystroke dynamics dataset that was collected by researchers at Clarkson University. This dataset includes keystroke timing information for 101 subjects in a completely uncontrolled and natural setting, with the date having been collected over a period of 2.5 years. Compared with other datasets which are controlled to some degree, the participants contribute their data with different computers, different keyboards, different browsers, different software, and even different tasks (e.g., gaming, email, etc.). Models that perform well on this dataset should also perform well in a real-world scenario.

Unfortunately, the Clarkson II dataset only provides very limited features—specifically, the timestamps of key-down and key-up events. The average number of keystrokes for each research subject is about 125,000. However, the number of keystroke events is far from uniform, with some users having contributing only a small number of keystrokes. Therefore, we set a threshold of 20,000 keystrokes, which gives us only 80 subjects.

2.3 Deep Leaning Algorithms

In this research, we apply deep learning methods to the free-text keystroke datasets discussed above. Our best-performing architecture is a novel combination of neural network based techniques. In this section, we briefly discuss the learning techniques that we have employed.

2.3.1 Multilayer Perceptron

Multilayer perceptrons (MLP) [18] are a class of supervised learning algorithms with at least one hidden layer. Any MLP consists of a collection of interconnected artificial neurons, which are loosely modeled after the neurons in the human brain. Nonlinearity is provided by the choice of the activation function in each layer. MLP is related to the classic machine learning technique of support vector machines (SVM).

2.3.2 Convolutional Neural Network

Convolutional neural networks (CNN) [1] are a special class of neural networks that make use of convolutional kernels to efficiently deal with local structure. CNNs are often ideal for applications where local structure dominates, such as image analysis. CNNs with multiple convolutional layers are able to extract the semantic information at different resolutions and have proven to be extremely powerful in computer vision tasks.

2.3.3 Recurrent Neural Network

Recurrent neural networks (RNN) [8] are used to deal with sequential or time-series data. For example, sequential information is essential for the analysis of text and speech. Plain "vanilla" RNNs suffer from vanishing gradients and related pathologies. To overcome these issues, highly specialized RNN architectures have been developed, including long short-term memory (LSTM) [8] and gated recurrent units (GRU) [6]. In practice, LSTMs and GRUs are among the most successful architectures yet developed. In this research, we focus on GRUs, which are faster to train than LSTMs, and perform well in our application.

2.3.4 Cutout

Fully connected neural networks often employ dropouts [19] to reduce overfitting problems. While dropouts work well for models with fully-connected layers, the technique is not suitable for CNNs. Instead, we use cutout regularization [7] with our CNN models. Cutouts are essentially the image-based equivalent of dropouts—we cut out part of the image when training, which forces the CNN to learn from other parts of the image. In addition to helping with overfitting, a model that is able to handle images with such occlusions is likely to be more robust.

3 Feature Engineering

As mentioned in Sect. 2, we consider two open source keystroke datasets. Both the Buffalo and Clarkson II datasets are free-text, and only provide fairly limited information. Therefore, we will have to consider feature engineering as a critical part of our experiments. In this section we consider different categories of features and various types of feature engineering.

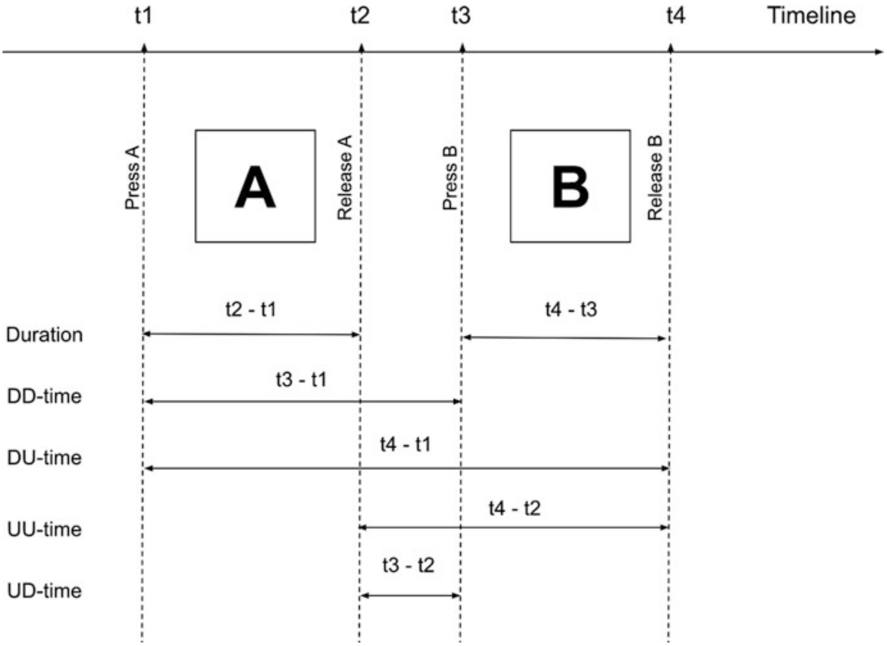

Fig. 1 Five time-based features

3.1 Features

With the development of mobile devices, modern keyboards are no longer limited to physical keyboards, but also include most virtual devices that allow user input. Ideally, we would like to consider patterns in user typing behavior with respect to time-based information and pressure-based features. However, pressure-based features are not directly available from the datasets used in this research. In the future, datasets obtained using mobile devices could include such information, which should enable stronger authentication and identification results.

Again, in this research we necessarily focus on time-based features, because that is what we have available in our keystroke datasets. The five time-based features that we consider are illustrated in Fig. 1.

Let `A` and `B` represent two consecutive input keys, with press and release representing a key-down and key-up event, respectively. The five time-based features are duration, down-down time (DD-time), up-down time (UD-time), up-up time (UU-time), and down-up time (DU-time). Duration is the time that the user holds a key in the down position, while the other four features are clear from the figure. Note that for any two consecutive keystroke events, say, `A` and `B`, six features can be extracted, namely, duration-`A`, duration-`B`, DD-time, UD-time, UU-time, and DU-time.

3.2 Length of Keystroke Sequence

As mentioned in Sect. 2, we can divide keystroke dynamics-based authentication into four categories depending on the length and consistency of the keystroke sequence. For our free-text keystroke datasets, the data consists of a long keystroke sequence of thousands of characters for each user. In previous research, such long sequences have been split into multiple subsequences, and we do the same here. Each subsequence is viewed as an independent keystroke sequence from the corresponding user. Previous research has shown that short keystroke subsequences decrease accuracy, while the longer keystroke subsequences may incorporate more noise. Therefore, we will need to experiment with different lengths of keystroke subsequence to determine an optimal value.

3.3 Keystroke Dynamics Image

In Sect. 2, we introduced the keystroke datasets used in this paper. As mentioned in the previous section, we divide the entire keystroke sequence into multiple subsequences, and in Sect. 3.1 we discussed the six types of timing features that are available. Thus, for a subsequence of length N, there are $6(N-1)$ features that can be determined from consecutive pairs of keystrokes, where repeated pairs are averaged and treated as a single pair. For example, for a subsequence of length 50, we obtain at most $6 \cdot 49 = 294$ features. We view each keystroke subsequence as an independent input sequence for the corresponding user. Next, we propose a new feature engineering structure to better organize these features.

The features UD-time, DD-time, DU-time, and UU-time are determined by consecutive keystroke events. Therefore, we organize these four features into a transition matrix with four channels, which can be viewed as four $N \times N$ matrices overlaid. This approach is inspired by RGB images, which have a depth of three, due to the R, G, and B channels.

Each row and each column in our four-channel $N \times N$ feature matrix corresponds to a key on the keyboard, and each channel corresponds to one kind of feature. Figure 2 illustrates how we have organized these features into transition matrices. For example, the value at row `i` and column `j` in the first channel of the matrix refer to the UD-time between any key presses of `i` followed by `j` within the current observation window.

The final feature is duration, which is organized as a diagonal matrix and added to the transition matrix as a fifth channel. Note that if a key or key-pair is pressed more than once, we use the average as the duration for that key or key-pair. In this channel, only diagonal locations have values because the duration feature is only relevant for one key at a time. The final result is that all of the features generated from keystroke subsequence are embedded in a transition matrix with five channels, which we refer to as the keystroke dynamics image (KDI).

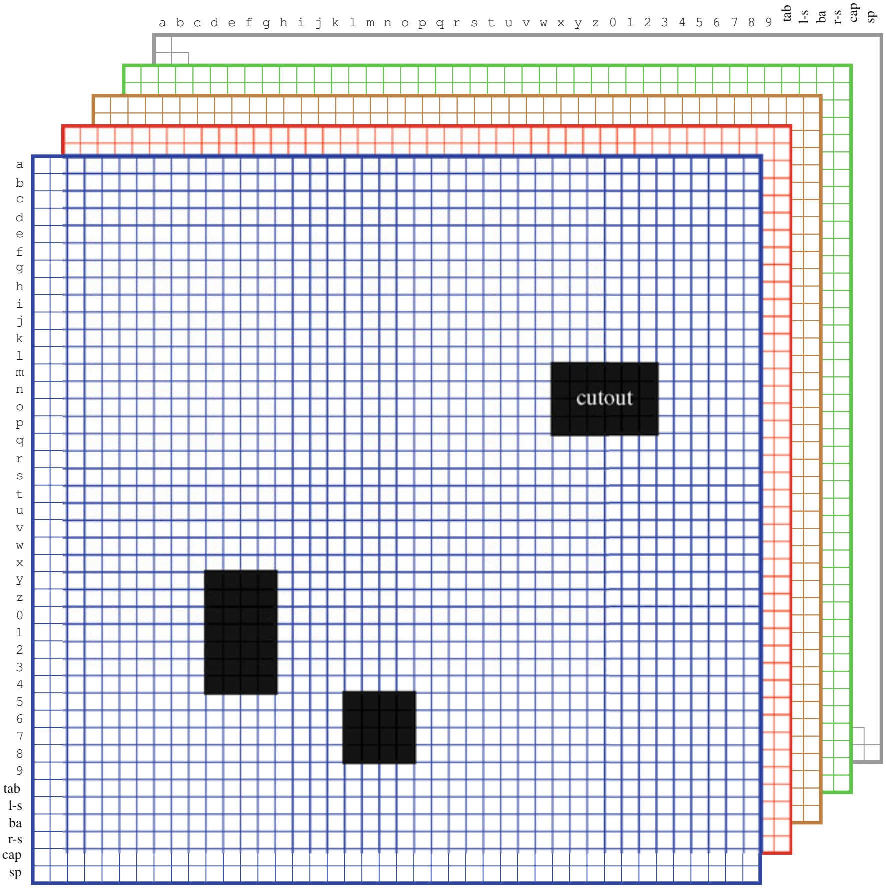
a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9 tab l-s ba r-s cap sp
a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9 tab l-s ba r-s cap sp
cutout

Fig. 2 Keystroke dynamics image for free-text

To prevent the transition matrix from being too sparse, we only consider time-based features for the 42 most common keystrokes. These 42 keys include the 26 English characters (A-Z), the 10 Arabic numerals (0-9), and six meta keys (space, back, left-shift, right-shift, tab, and capital). Therefore, the shape of the transition matrix is 5 × 42 × 42, with the five channels as discussed above.

3.4 *Keystroke Dynamics Sequence*

Above, we provided details on the time-based image-like feature that we construct, which we refer to as the KDI. In this section, we discuss the application of an

RNN-based neural network to the KDI. Our goal is to use this feature to better take advantage of the inherently sequential nature of the keystroke dynamics data.

A keystroke in a keystroke sequence can be viewed as a word in a sentence. For our two free-text keystroke datasets, the keystroke sequence is different for each input and each user. For this data, we consider various encodings of each keystroke and use this encoding information in the embedding vector. Specifically, we experiment with index encoding and one-hot encoding. The resulting embedding vectors are used to construct a keystroke dynamics sequence, which we abbreviate as KDS. These KDS vectors will be used in our RNN-based neural networks.

3.5 Cutout Regularization

As mentioned in Sect. 2.3.4, we employ a cutout regularization to prevent overfitting in our CNN. By artificially adding occlusions to our image-like data, the network is forced to pay attention to all parts of the image, instead of over-emphasizing some specific parts. We apply cutouts to our novel KDI data structure, which is discussed in Sect. 3.3, and the KDS, which was mentioned in Sect. 3.4. The dark blocks in Fig. 2 illustrate cutouts.

4 Architecture

In this section, we discuss the classification models in more detail. We also discuss hyperparameters tuning for the models considered.

4.1 Multiclass vs Binary Classification

The Buffalo and Clarkson II keystroke datasets are based on 101, and 148 subjects, respectively. Regardless of the dataset, our goal is to verify a user's identity based on features derived from keystroke sequences. While this is a classification problem, we can consider it as either a multiclass problem or multiple binary classification problems. In a practical application, the number of users could be orders of magnitude higher than in either of our datasets. To train a multiclass model on a large number of users would be extremely costly, and each time a new user joins, the entire model would have to be retrained. This is clearly impractical.

To train and test our models, we require positive and negative samples for each user. All the data available for a specific user will be considered as positive samples, while an equivalent number of negative samples are selected at random (and proportionally) from other users' samples. In practice, the number of non-target

Table 2 Best Hyperparamters of deep learning models

Parameter	Search space
Training epochs	100, **200**, 500, 1000
Initial learning rate	0.1, **0.01**, **0.001**, 0.0001
Optimizer	**Adam**, SGD, SGD with Momentum
Learning schedule	**StepLR** (**0.1**, 0.3, 0.5), Plateau
Experiments	**50**

users may be very large. In that case, we could draw negative samples from a a fixed number of non-target users.

4.2 Hyperparameter Tuning

For the deep learning methods used in our experiments, we employ a grid search to find the best parameters for the initial learning rate, optimizer, number of epochs, and learning rate schedule. The values shown in Table 2 were tested, and those in boldface were found to generate the best result. To allow for a direct comparison of our different models, we use these same hyperparameters for all of our deep learning models. Note that a learning rate of 0.01 generates the best results for CNN, MLP, LSTM, GRU, while a learning rate of 0.001 generates best result in our RNN experiments.

4.3 Implementations

For our keystroke dynamics experiments, we evaluate two kinds of models. Specifically, a CNN is applied to the our novel KDI image-like features, while a hybrid model that combines CNN and GRU is applied to the KDS features. The KDI is presented in Sect. 3.3, while the free-KDS is described in Sect. 3.4.

4.3.1 CNN

The architecture of our CNN is shown in Fig. 16 in the Appendix. The input of this model is the KDI, and hence we view the transition matrix as an image. Here, a "stage" includes two `conv2d` layers and a `maxpooling` layer, not counting the activation function.

In each stage, there are two convolutional layers and a maxpooling layer. Moreover, a `relu` function is employed after each convolutional layer. Following these two stages, there are three fully connected layers, and a dropout layer is added to prevent overfitting. Finally, a sigmoid function is used to compute the final probability of a positive sample.

4.3.2 CNN-RNN

The architecture of our CNN-RNN is illustrated in Fig. 17 in the Appendix. The input to this model is the KDS mentioned in Sect. 3.4. Note that 32 convolutional kernels shift in the keystroke sequence direction, and thus a sequence matrix with embedding size 32 is generated. This resulting output matrix is fed into a 2-layers GRU network, which is followed by a fully connected layer. Since this is a binary classification model, a sigmoid function is used to compute the probability of a positive sample.

5 Experiment and Result

In this section, we provide experimental results for our free-text binary classification experiments. The results of the various models considered are analyzed and compared. Note that in all of our experiments, we apply 5-folds cross validation and average the performance for each user.

5.1 Metrics

We adopt two metrics to evaluate our results. The first metric is accuracy, which is simply the number of correct classifications divided by the total number of classifications. More formally, accuracy is calculated as

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

where TP and TN are true positives and true negatives, while FP and FN are false positives and false negatives.

There are two kinds of classification errors, namely, false positives and false negatives. There is an inherent trade-off between the false positive rate (FPR) and the false negative rate (FNR), in the sense that by changing the threshold that we use for classification, we can lower one but the other will rise. For a metric that is threshold-independent, we compute the equal error rate (EER) which, as the name suggests, is the value for which the FPR and FNR are equal. The EER is obtained by considering thresholds in the range of [0, 1] to find the point where the FPR and FNR are in balance. Figure 3 illustrates a technique for determining the EER.

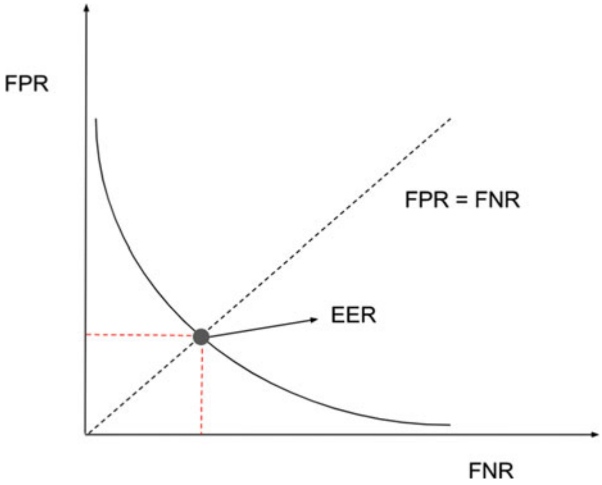

Fig. 3 Equal error rate

5.2 Result of Free-Text Experiments

For our free-text experiments, we focus on the effect of the lengths of keystroke sequences, kernel sizes for the CNN, encoding methods for the keystroke sequence data, and different hyperparameters of the RNN. Additionally, we explore the performance of models with and without cutout regularization.

5.2.1 Length of Keystroke Subsequence

First, we experiment with different lengths of keystroke subsequences. Specifically, we consider lengths of 50, 75, and 100 keystrokes. The results of these experiment are given in Figs. 4 and 5 for the Buffalo and Clarkson II datasets, respectively. From these results, we observe that when the length of a keystroke sequence has minimal impact on the accuracy or EER.

From these results, we observe that when the length of a keystroke sequence is relatively short, there is insufficient information to support strong authentication and, conversely, when the sequence is too long, the additional noise degrades the accuracy. Moreover, the results shows that the CNN-based model is more robust when the length of the keystroke sequence changes, which can be explained by the KDI mitigating the noise inherent in longer sequences. To accelerate the training process, we adopt the length 100 for the keystroke subsequences in all subsequent experiments.

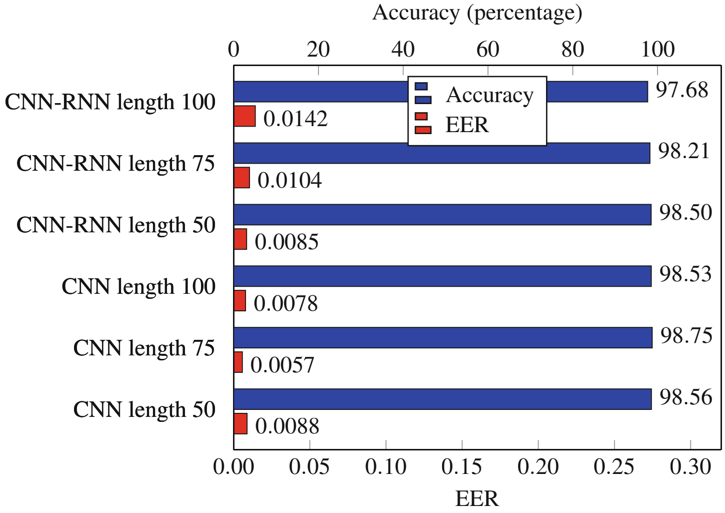

Fig. 4 Keystroke lengths (Buffalo dataset)

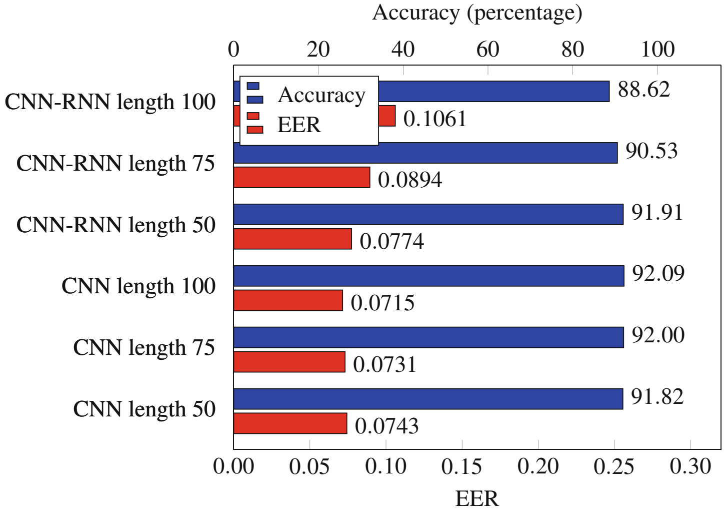

Fig. 5 Keystroke lengths (Clarkson II dataset)

5.2.2 CNN Kernel Sizes

In any CNN, the kernel size is a critical parameter. To determine the optimal kernel size, we experiment with three square kernels (3 × 3, 5 × 5, and 7 × 7) in our basic CNN model. For the CNN part of our hybrid CNN-RNN model, we experiment with

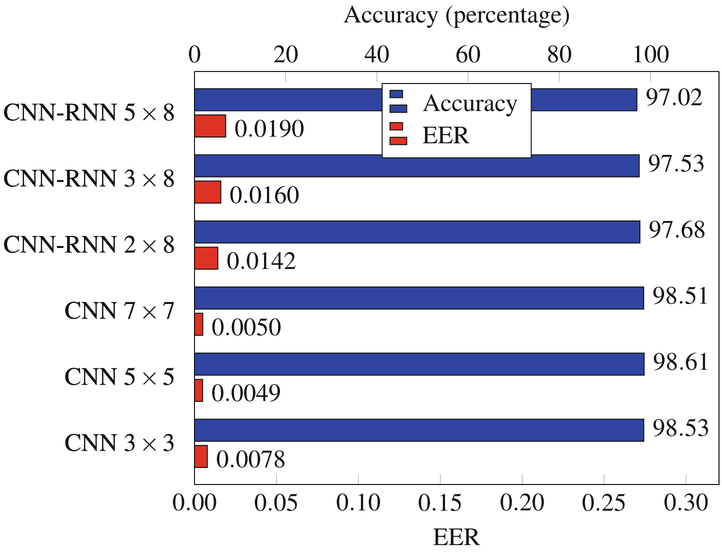

Fig. 6 Kernel size (Buffalo dataset)

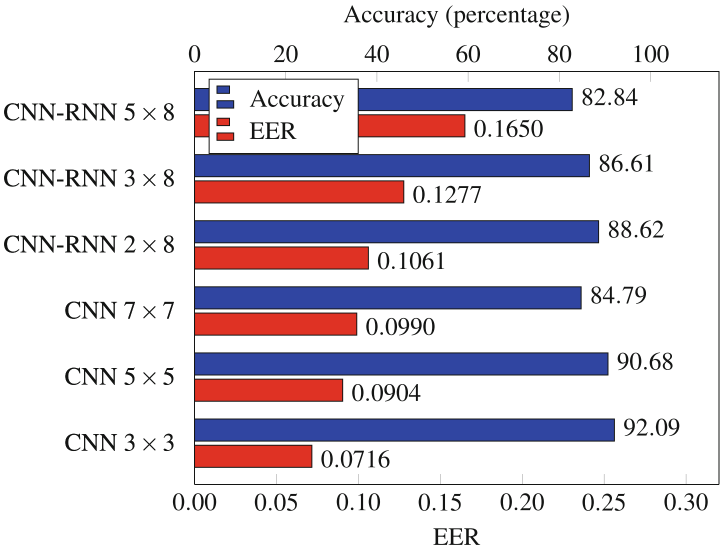

Fig. 7 Kernel size (Clarkson II dataset)

three sizes of rectangle kernels (2 × 8, 3 × 8, and 5 × 8). These experimental results for the basic CNN and hybrid CNN-RNN are given in Figs. 6 and 7.

We note that the kernel size makes no appreciable difference for the basic CNN model on the Buffalo dataset, while the two larger kernels both perform equally well

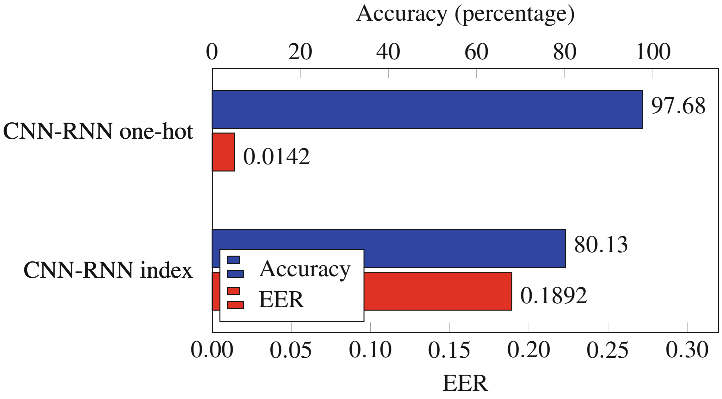

Fig. 8 Embedding methods (Buffalo dataset)

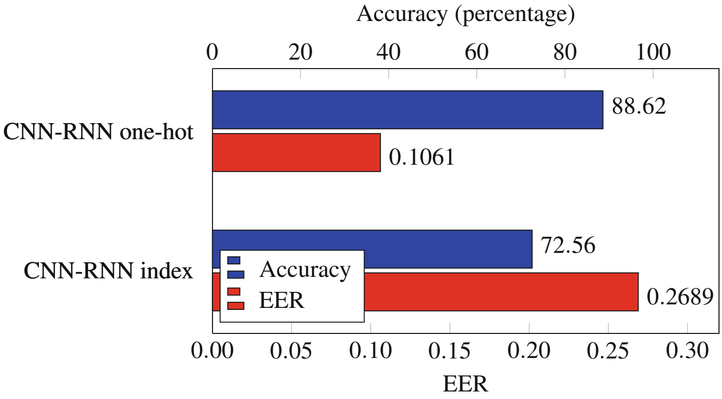

Fig. 9 Embedding methods (Clarkson II dataset)

on the Clarkson II dataset. For the CNN-RNN model, the results are also mixed, with the smaller kernel giving the best results over the two datasets. We adopt 3×3 square kernels for CNN-based models and 2×8 kernels for CNN-RNN based model in subsequent experiments.

5.2.3 Embedding Method

As mentioned above in Sect. 3.4, we consider two embedding methods, namely, index encoding and one-hot encoding. These experimental results are given in Figs. 8 and 9 for the Buffalo and Clarkson II datasets, respectively. From these results, it is clear that one-hot encoding is far superior to index encoding, and hence in subsequent experiments, we use one-hot encoding.

5.2.4 RNN Structure

We experiment with three types of RNN-based networks in our CNN-RNN architecture. Specifically, we consider a plain RNN, GRU, and LSTM. The advantages of GRU and LSTM are that they can capture more long-term information than a plain RNN. The results of these experiments are given in Figs. 10 and 11.

For the Buffalo keystroke dataset, the performances of our three different models are virtually identical, which indicates that the most valuable information is contained in adjacent keystroke pairs. However, for the Clarkson II keystroke dataset, we find that the GRU is more effective than the other two architectures. A plausible explanation is that LSTM is more prone to overfitting, while RNN is simply less powerful. And it appears that the GRU is slightly better at dealing with noisy data.

5.2.5 Cutout Experiments

It is likely that the data extracted from keystroke dynamics sequences is noisy because of the various extraneous factors that can influence typing behavior. We use cutout regularization, since it is useful at preventing overfitting, and since it is believed to reduce the effect of noisy information. The results of our cutout experiments are given in Figs. 12 and 13. We observe that cutout regularization has a significant positive effect on the performance of our models, which is most obvious in the CNN-based model. This is reasonable, since the cutout concept derives from the field of computer vision and our input data (i.e., KDI) is an image-like data structure.

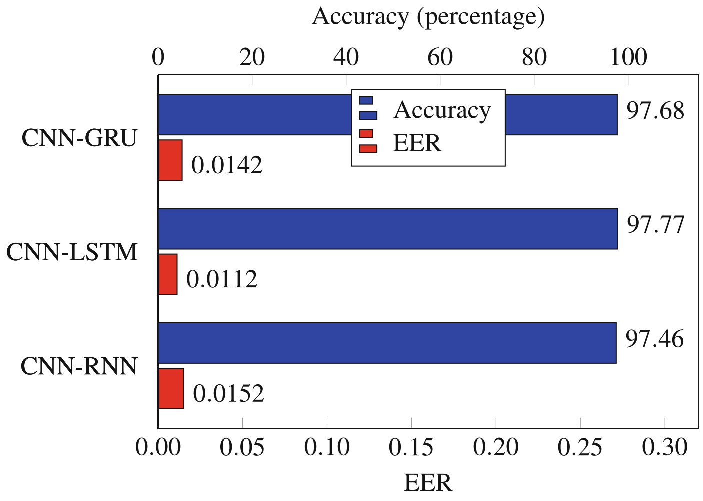

Fig. 10 CNN-RNN (Buffalo dataset)

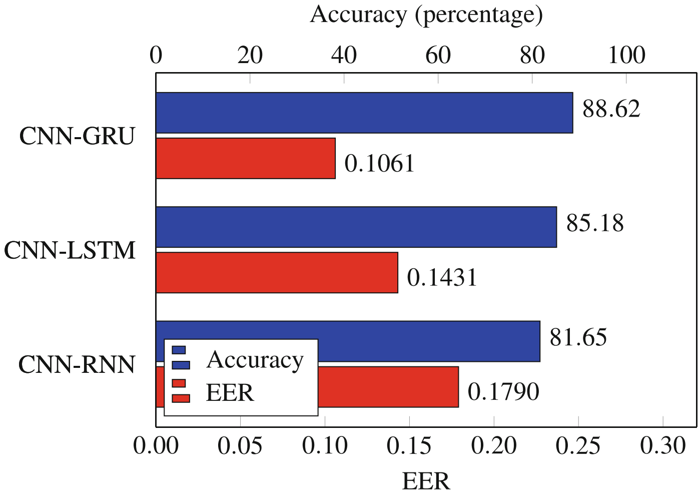

Fig. 11 CNN-RNN (Clarkson II dataset)

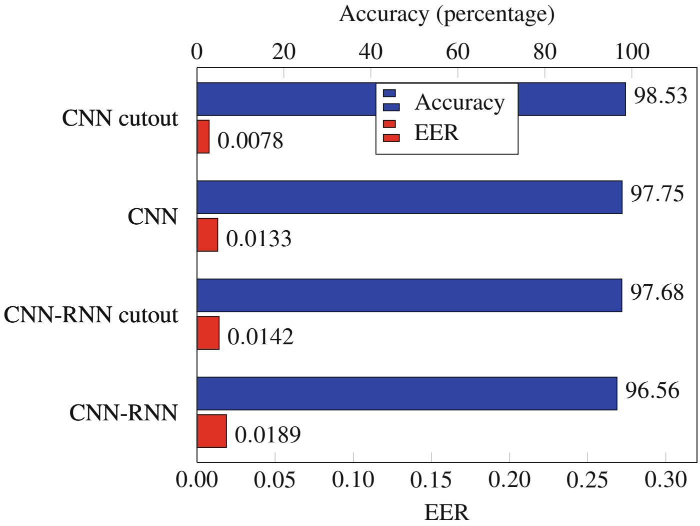

Fig. 12 Cutout regularization (Buffalo dataset)

5.3 *Discussion*

In our experiments, the performance on the Buffalo dataset is consistently higher than that of the Clarkson II dataset. It is likely the case that the latter dataset contains noisier data, as it was collected over a period of 2.5 years and under far less controlled conditions. We also find that our CNN-based model (KDI + CNN)

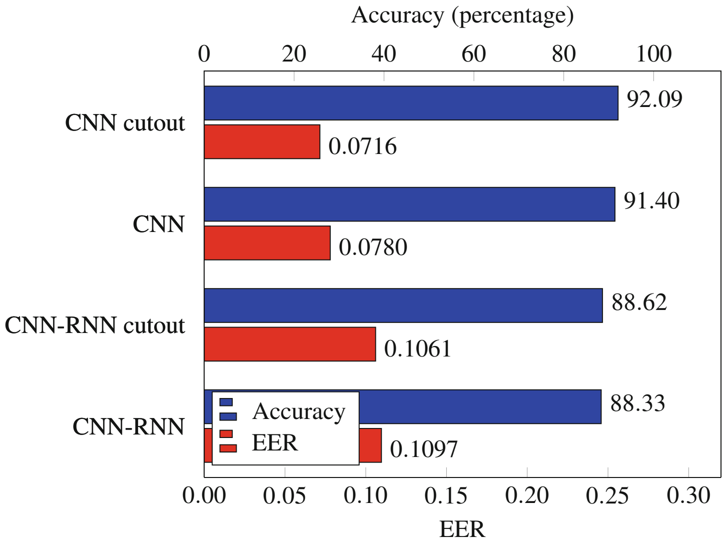

Fig. 13 Cutout regularization (Clarkson II dataset)

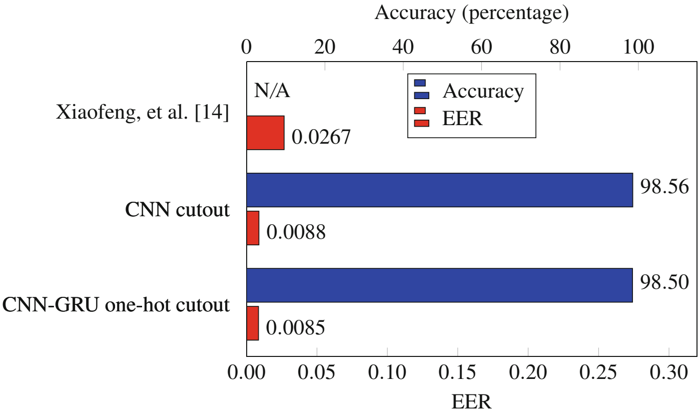

Fig. 14 Comparison to previous work (Buffalo dataset)

consistently generates better results than our RNN-CNN based model (Free-KDS + CNN-RNN). Comparing our results with the previous work in [14], we observe that in terms of EER, our two models both perform better on the Buffalo dataset, but slightly worse on the Clarkson II dataset. These results are summarized in Figs. 14 and 15.

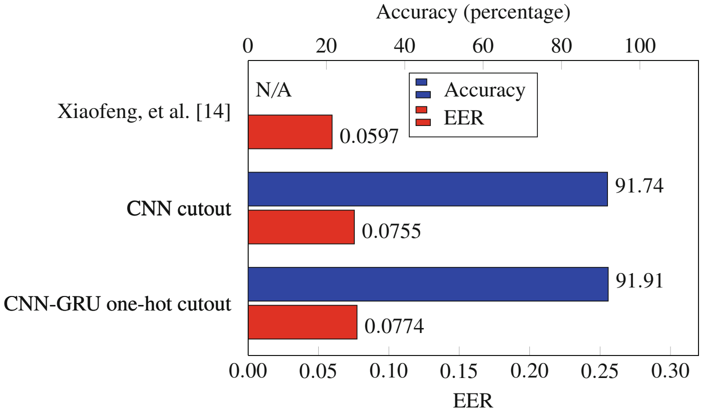

Fig. 15 Comparison to previous work (Clarkson II dataset)

6 Conclusion

This research focused on authentication based on keystroke dynamics derived features in the free-text case. We found that dividing the sequence into a number of fixed-length subsequences was an effective feature engineering strategy. In addition, we developed and analyzed an image-like engineered feature structure that we refer to as KDI, and we compared this to another structure that we refer to as KDS. The KDI was used as the input for our CNN experiments, while the KDS served as the input data for our CNN-RNN experiments. In both cases, we applied cutout regularization.

The experimental results reported here show that our pure CNN architecture outperforms our combination of CNN and RNN, and cutout significantly improves the performances of both models. Moreover, our two modeling approaches both outperform previous work on the Buffalo keystroke dataset and yield competitive results for the Clarkson II dataset.

In the realm of future work, we conjecture that generative adversarial networks (GAN) will prove useful in this problem domain. More fundamentally, we believe that improved (and larger) datasets are necessary if we are to make significant further progress on this challenging authentication problem.

Appendix

See Figs. 16 and 17.

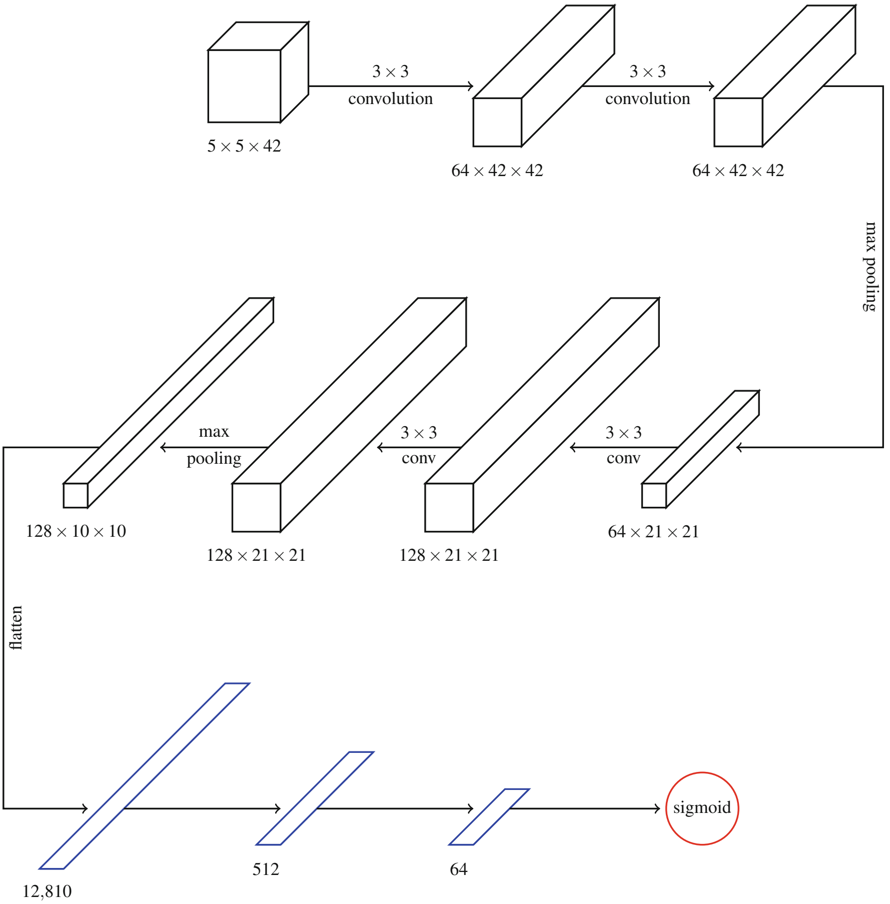

Fig. 16 Architecture of CNN for free-text datasets

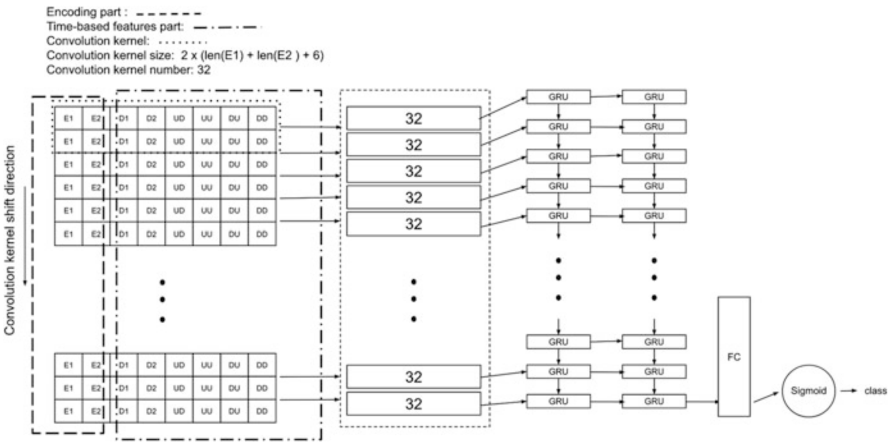
Encoding part :
Time-based features part:
Convolution kernel:
Convolution kernel size: 2 x (len(E1) + len(E2) + 6)
Convolution kernel number: 32
Convolution kernel shift direction
E1
E2
D1
D2
UD
UU
DU
DD
32
GRU
FC
Sigmoid
class

Fig. 17 Architecture of CNN-RNN for free-text datasets

References

1. Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology*, ICET, pages 1–6, 2017.
2. Faisal Alshanketi, Issa Traore, and Ahmed Awad Ahmed. Improving performance and usability in mobile keystroke dynamic biometric authentication. In *2016 IEEE Security and Privacy Workshops*, SPW, pages 66–73, 2016.
3. Mario Luca Bernardi, Marta Cimitile, Fabio Martinelli, and Francesco Mercaldo. Keystroke analysis for user identification using deep neural networks. In *2019 International Joint Conference on Neural Networks*, IJCNN, pages 1–8, 2019.
4. Hayreddin Çeker and Shambhu Upadhyaya. Enhanced recognition of keystroke dynamics using gaussian mixture models. In *2015 IEEE Military Communications Conference*, MILCOM, pages 1305–1310, 2015.
5. Hayreddin Çeker and Shambhu Upadhyaya. Sensitivity analysis in keystroke dynamics using convolutional neural networks. In *2017 IEEE Workshop on Information Forensics and Security*, WIFS, pages 1–6, 2017.
6. Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. https://arxiv.org/abs/1412.3555, 2014.
7. Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. https://arxiv.org/abs/1708.04552, 2017.
8. Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
9. Pilsung Kang and Sungzoon Cho. Keystroke dynamics-based user authentication using long and free text strings from various input devices. *Information Sciences*, 308:72–93, 2015.
10. Junhong Kim, Haedong Kim, and Pilsung Kang. Keystroke dynamics-based user authentication using freely typed text based on user-adaptive feature extraction and novelty detection. *Applied Soft Computing*, 62:1077–1087, 2018.
11. Paweł Kobojek and Khalid Saeed. Application of recurrent neural networks for user verification based on keystroke dynamics. *Journal of Telecommunications and Information Technology*, 2016:80–90, 2016.

12. Gutha Jaya Krishna, Harshal Jaiswal, P. Sai Ravi Teja, and Vadlamani Ravi. Keystroke based user identification with XGBoost. In *2019 IEEE Region 10 Conference*, TENCON, pages 1369–1374, 2019.
13. Andreas Lanitis. A survey of the effects of aging on biometric identity verification. *International Journal of Biometrics*, 2(1):34–52, 2010.
14. Xiaofeng Lu, Shengfei Zhang, and Shengwei Yi. Free-text keystroke continuous authentication using CNN and RNN. *Journal of Tsinghua University (Science and Technology)*, 58(12):1072–1078, 2018.
15. Andrew Maas, Chris Heather, Chuong (Tom) Do, Relly Brandman, Daphne Koller, and Andrew Ng. Offering verified credentials in massive open online courses: MOOCs and technology to advance learning and learning research. *Ubiquity*, 2014:1–11, 2014.
16. Fabian Monrose and Aviel D. Rubin. Keystroke dynamics as a biometric for authentication. *Future Generation Computer Systems*, 16(4):351–359, 2000.
17. Eduard C. Popovici, Ovidiu G. Guta, Liviu A. Stancu, Stefan C. Arseni, and Octavian Fratu. MLP neural network for keystroke-based user identification system. In *11th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services*, TELSIKS, pages 155–158, 2013.
18. Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.
19. Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.
20. Y. Sun, Hayreddin Çeker, and Shambhu Upadhyaya. Shared keystroke dataset for continuous authentication. In *2016 IEEE International Workshop on Information Forensics and Security*, WIFS, pages 1–6, 2016.
21. Fabio Di Tommaso, Michele Guerra, Fabio Martinelli, Francesco Mercaldo, Massimo Piedimonte, Giovanni Rosa, and Antonella Santone. User authentication through keystroke dynamics by means of model checking: A proposal. In *2019 IEEE International Conference on Big Data*, Big Data, pages 6232–6234, 2019.
22. Esra Vural, Jiaju Huang, Daqing Hou, and Stephanie Schuckers. Clarkson University keystroke dataset. https://citer.clarkson.edu/research-resources/biometric-dataset-collections-2/clarkson-university-keystroke-dataset/.
23. Jatin Yadav, Kavita Pandey, Shashank Gupta, and Richa Sharma. Keystroke dynamics based authentication using fuzzy logic. In *2017 Tenth International Conference on Contemporary Computing*, IC3, pages 1–6, 2017.