User Recognition from Keystroke Dynamics

Saurav Mehra
2022465
IIIT-Delhi

Ankur Bohra
2022079
IIIT-Delhi

I. Introduction

Keystroke dynamics describes a person's typing "rhythm" through a time series of data, recording each key press and key release. Although keystroke dynamics biometrics have not proved robust enough to be a sole criterion for authentication, it is a low-cost method that can be easily integrated and used in conjunction with conventional robust authentication methods. It evokes further interest as one of the few other authentication methods that test the user's identity, without relying on proxies such as information.
We initially used Manhattan Distance and Gaussian Mixture Models for user authentication for both fixed text and free text. We further explored more complex models involving Feed-Forward Neural Networks as well as Recurrent Neural Networks.

II. Literature Review

There are a plethora of approaches to keystroke dynamics, that use a variety of statistical measures ranging in complexity [1]. The use of keystroke dynamics for user authentication was first delved into in the 1970s [7]. While this initially focused on fixed text datasets, a password typed multiple times for example, continuous authentication of a user in an unconstrained environment has drawn research interest in recent times.
Post feature extraction from the datasets, the common techniques for keystroke dynamics involve distance-based classification, K-means methods [8], Support Vector Machines (SVMs) [9] and Hidden Markov Models [10] to name a few. For the purposes of this project, we have limited ourselves to minimizing error through Manhattan distance [2], Gaussian Mixture Models [3] as well as Feed-Forward Neural Networks and Recurrent Neural Networks [6].
Keystroke dynamics are not limited in function to merely user authentication, but their practicality in emotion recognition has also been explored [11]. However, as far as classification is concerned, the most commonly used metrics to assess the performance of the models are False Acceptance Rate (FAR) and False Rejection Rate (FRR), along with Equal Error Rate (EER).

III. Dataset Details

The datasets in this field can be classified into two types: fixed text and free text. In the former, the subjects are required to type a set of fixed letters repeatedly, while, in the latter, the subjects are asked a few questions which they are free to answer as they wish.
For the fixed text dataset, we are using the dataset published from a 2019 CMU study [2]. It contains data from 51 subjects typing the same password (**.tie5Roanl**) 400 times across 8 sessions. Each row contains 34 parameters, indicating the details of the subject as well as all the information about the typed password, such as the holding period, time between presses and so on.

The second dataset is from a 2016 study published by the University of Buffalo [4]. It contains data collected from 148 users over 3 sessions, with each session having 5,700 keystrokes. For each user and session, the key, the action (keyUp or keyDown) and the timestamp in milliseconds is stored.
This dataset is preprocessed to create digraphs [5] which store the relation between two consecutively pressed keys. Each digraph stores the following information in the form of a vector:

1) The first key pressed
2) The second key pressed
3) Approximation of the lateral distance between the two keys on a physical keyboard
4) Approximation of the vertical distance between the two keys on a physical keyboard
5) The time the first key was pressed down for
6) The time the second key was pressed down for
7) The press-to-press time, or the time from the first key press till the second key was pressed
8) The release to press time, or the time from the release of the first key till the second key was pressed

IV. Proposed Architecture

A. Filtered Manhattan Distance

Manhattan Distance is also known as the city block distance or the L1 distance. This method is used for the dataset containing fixed text, where each sample observation is easily represented as a vector. We have increased its robustness

by filtering those points from the dataset which differ from the mean by greater than 2 standard deviations (a standard threshold in anomaly detection).
Manhattan distance allows each dimension to be treated independently, and is not overly affected by a single dominant dimension as in Euclidean distance. It can be decomposed into its different dimensions and be computed easily.
We have applied this model on the fixed text dataset [2]. For each user, of their 400 typed passwords, a subset is chosen as the training set. Each training set is filtered in order to remove the outliers, and the mean from the resulting set is calculated. The remaining data is used as the testing set against which we minimize the equal error rate using the ROC curve.

B. Gaussian Mixture Models (GMMs)

Mixture models generally consist of multiple components, each possibly assuming a different distribution. Standard distributions are typically limited when one wishes to model complex distributions like several possibly distant areas of high density. The assumption for a mixture model, however, is that the bevahiour of the output is explained by one of several generating functions, and that the particular generating function is sampled with some probability (the component's mixture weight). This allows mixture models to better model more complex distributions. This is especially appropriate for keystroke dynamics, as the behaviour for a pattern of keys is expected to be highly dependent on factors like current positions of the hands. Mixture models enable us to capture this model of a behaviour as a mixture of several *sub-behaviours*. Gaussian mixture models are a popular class of mixture models where each component is also a Gaussian, possibly with different means and covariances. They are the standard choice for mixture models, much like Gaussians are the standard choice for single distributions.
We use GMMs separately for both the fixed text and free text datasets.
For the fixed text dataset, the GMM models the distribution of the complete observation vector for each user. The profile of each user is represented by the GMM trained on their inputs. To check whether a given input was legitimate, a score for the input is computed against the target profile, and a simple threshold (computed during training) decides whether the input is indeed legitimate or an imposter. We define the score as the negative of the log-likelihood (which itself ranges from $-\infty$ to 0) of the sample under the model. So the score ranges from 0 to ∞ and lower scores are associated with higher likelihoods. The threshold is calculated to minimize the equal error rate as for filtered Manhattan distance.
For the free text dataset, there is no direct representation of each observation as the specific digraphs observed vary with each user and each observation. This is a major drawback for the free text application as the representation of inputs as vectors is a crucial idea in most methodologies. Instead of modelling each observation, the GMM now models the distribution of the press-to-press time of each digraph of each user. Thus the profile of a user is described by several GMMs each modeling a specific digraph. This aligns well with the behaviour capturing interpretation of mixture models discussed earlier, although correlation between digraphs is not well captured. To determine whether an input observation is legitimate, we use the similarity score defined in [3]. It is the (weighted) number of digraphs of the observation that individually pass a legitimacy test for the target profile. The legitimacy test for each digraph is whether the press-to-press time of the digraph is within some specific number of standard deviations of some component (the zone of acceptance, Fig 1) in the mixture model. If the distance of the digraph is within the threshold for a component, the number of passed digraphs is increased by the mixture weight of the component (so that the uncertainty between the specific generating functions is well represented).

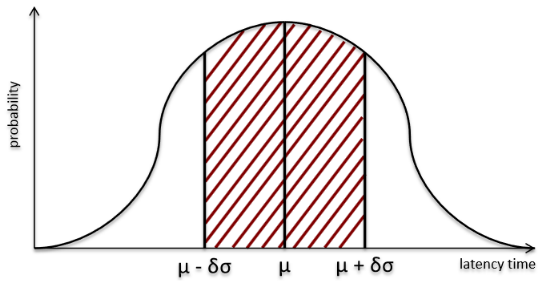

Fig. 1. The zone of acceptance as shown in [3]

C. Feed-Forward Neural Networks

Feed-forward neural networks (FFNNs) operating on the principle of back propagation provide a simple yet nuanced approach to tackle this problem for free text datasets with input sequences of equal length.
For each of the 51 users in the fixed text dataset, a one-hot encoding is performed. A sequential model is created that takes in the entirety of the 31 parameters recorded in the dataset. There are two hidden dense layers in the network with different activation functions, followed by a final dense output layer with $softMax$ applied to it. The network tries to minimize the categorical cross-entropy loss using the Adam optimizer. Varying the activation functions in the hidden layer, significantly different results are observed.

D. Recurrent Neural Networks

[6] discussed the use of Recurrent Neural Networks (RNNs) as a deep learning method for classifying users for both fixed and free text datasets.
We have made use of RNNs on the fixed text dataset. It has been observed in the past [1] that simple neural networks suffer from relatively poor classification performance since they often get trapped in local minima during classification.
RNNs, on the other hand, are especially suitable for processing and classifying temporal data. The RNN uses the features showing the relation between two keys pressed in continuation as done in other simpler models. Furthermore, it establishes a

chronological order for the entire keystroke sequence.
We have incorporated the use of Long Short-Term Memory Networks (LSTMs), which are a variant of RNNs. LSTMs allow long-term dependencies to be captured, making them viable for keystroke dynamics. Multiple LSTM layers, with different activation functions ($tanh$, with $softMax$ applied at the last layer), can be stacked, enabling each stack to learn a different level of abstraction from the input and learn more subtle patterns in the data. Dropout is also applied at each layer to prevent overfitting, and a final pair of dense and softmax layers enables classification from the output of the core model. Similar to FFNNs, Adam optimization is used to minimize the loss.

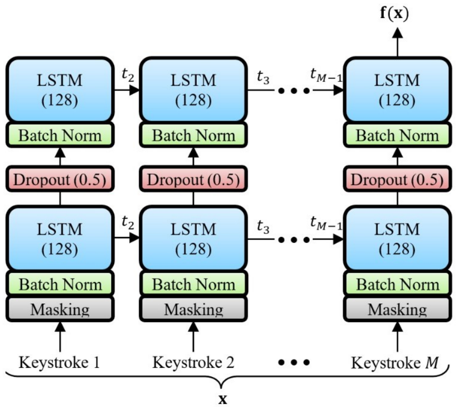

Fig. 2. The RNN architecture suggested in [12]

A limitation of this model that was encountered while using FFNNs, is that the number of keystrokes in each sequence must be the same. While this obstacle is not encountered in the fixed text dataset, since all sequences are of the same length, it poses a challenge while constructing a similar model for the free text dataset. A common method [6] to solve this issue is to simply break long keystroke sequences into several sequences of fixed size, and zero-pad any sequences left with shorter lengths. These padded keystrokes are ignored during training as to not confuse them with actual keystrokes.
A major difference in the application of RNNs to free text is that the same index may represent completely different keystrokes across different sequences. This calls for the addition of the key-code itself as a feature in the keystroke vector, in addition to the press/release statistics calculated in the fixed text case. Even after adding a key-code feature, we observed that the modified model was not able to satisfactorily classify keystroke sequences.
The models covered so far have implicitly imposed the constraint that users must have ample training data already available at the time of model training, and the addition of each new user must be followed by retraining the entire model. This is inconvenient and costly in a real life setting. Our literature review suggested that this can be avoided by instead training a model to learn an *embedding* for input keystroke sequences such that distance in the embedding space is positively correlated with similarity between typing profiles. There are several ways a model can be trained to get the optimal embedding space, for example, extension of the $softMax$ model, contrastive loss, and triplet loss [6] to name a few. As we have already implemented the $softMax$ model, we explore the embedding extension of the model, in which the last dense and $softMax$ layers are removed after training the model for classification. The remaining model can be interpreted as mapping an input sequence to an appropriate embedding space. During testing, we can find the distance between the embeddings of two given input sequences, and use a threshold to classify the sequences as belonging to the same user, or not. We observed that this method does not yield results comparable to the previously discussed models. The classification model does not promote the intra-class cohesion and inter-class separation goals appropriate for an embedding space, so this is not entirely unexpected.

V. Results

For the model based on the filtered Manhattan distance with a training size of 0.75, the Equal Error Rate (EER) ranges from $0.04 - 0.36$. The average EER can be observed as 0.17. The FAR vs. FRR curve for different users can be observed in Figure 3.

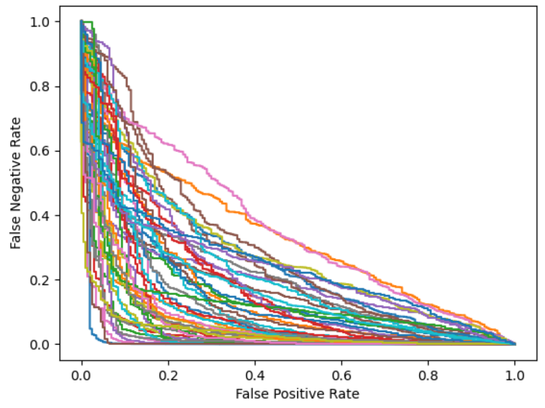

Fig. 3. FAR vs. FRR curve using Filtered Manhattan Distance

We also observe that even though Manhattan distance is more robust to outliers than Euclidean distance, filtering increases the EER.

The results of applying the Gaussian Mixture Model on the free text database with 2 Gaussians and varied threshold values can be seen in Table I and is visually depicted in Figure 4.

For the Gaussian Mixture Model applied to the fixed text dataset, the average EER can be observed as 0.14 with 2 Gaussian distribution components. The FAR vs. FRR curve for different users based on this model can be observed in Figure 5.

A decrease in EER can be observed as the number of components in the mixture model increase, as seen in Table

Components	Threshold	FAR (%)	FRR (%)
2G	0.4	87.45	0
2G	0.45	71.49	0
2G	0.5	46.22	0
2G	0.55	18.77	3.37
2G	0.6	3.95	29.72
2G	0.65	0.25	78.37
2G	0.7	0	100

TABLE I
FAR AND FRR FOR FREE TEXT GAUSSIAN MIXTURE MODEL

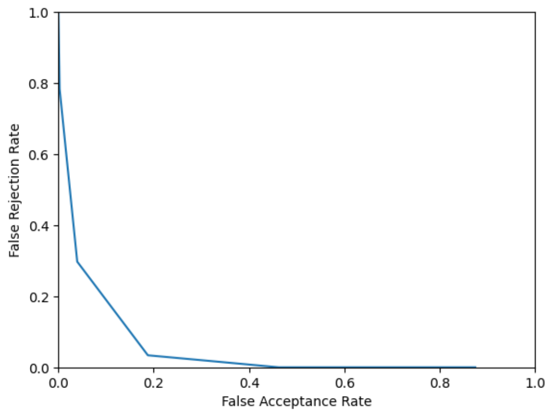

Fig. 4. FAR vs. FRR ROC for free text GMM

II. This empirically shows how increasing the number of components in a mixture model allows capturing more complex distributions. Notably, the effect saturates as the distribution is increasingly well approximated.

Components	EER
1G	0.15
2G	0.14
3G	0.132
4G	0.131
5G	0.128
10G	0.125

TABLE II
EER FOR FIXED TEXT GMM WITH VARYING COMPONENTS

The results of the application of FFNNs on the fixed text dataset with varied activation functions on the hidden layer can be seen in Table III, with the model being trained for 300 epochs.

Activation Function	Test Accuracy (%)	EER
Linear	81.7	0.0404
Sigmoid	81.7	0.0381
$tanh$	85.4	0.0302
ReLU	85.6	0.0301

TABLE III
TEST ACCURACY AND EER FOR FIXED TEXT FFNN

It can be observed that adding a certain amount of non-linearity improves the performance of the model. Figure 6 shows the variation of the training accuracy with the number of epochs.

The results of the RNN model on the fixed text dataset with varying number of LSTM layers can be seen in Table IV. Each

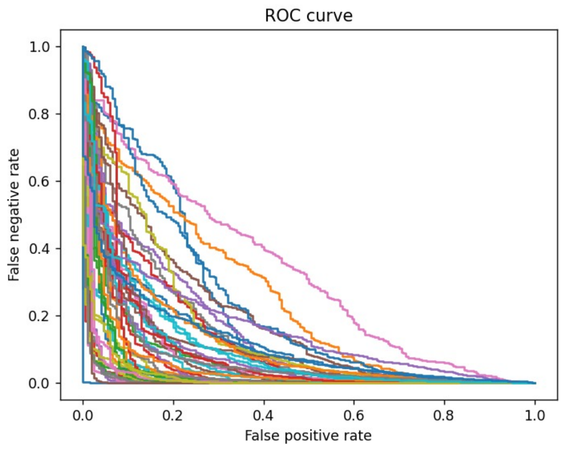

Fig. 5. FAR vs. FRR for fixed text GMM

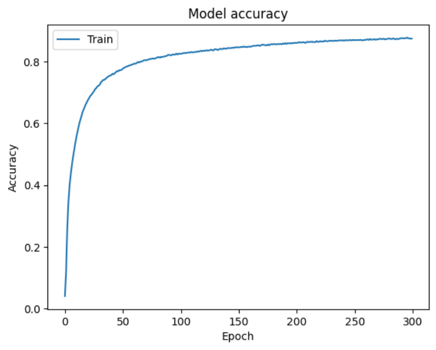

Fig. 6. Training Accuracy vs Number of epochs for FFNN

model was run for 20 epochs, with $tanh$ activation on each of the hidden layers and sigmoid activation on the output layer. Figure 7 shows the ROC curve for the model with 3 layers.

Number of LSTM Layers	Test Accuracy (%)	EER
1	78.87	0.0857
2	86.1	0.0477
3	86.2	0.0406
4	88.5	0.0387

TABLE IV
TEST ACCURACY AND EER FOR FIXED TEXT RNN WITH VARYING NUMBER OF LAYERS

VI. CONCLUSION

Using the fixed text dataset, we have observed that the filtered Manhattan distance model reduces the EER as compared to the unfiltered counterpart. The Gaussian Mixture Model applied to this dataset further reduces the EER. Furthermore, from table II, it can be observed that not only does the Gaussian Mixture Model perform better than a singular Gaussian distribution, it seems to improve the performance as the number of Gaussian distributions increase.
Using FFNNs, a more advanced technique, further reduces the EER. Incorporating non-linearity within the FFNNs improves the model compared to the linear counterpart. The introduction

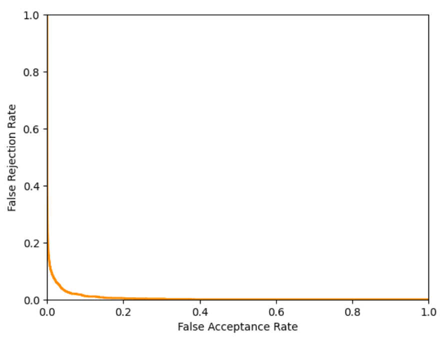

Fig. 7. ROC Curve for fixed text model with 4 LSTM layers

of deep learning in the form of LSTMs provides a similar EER, but improves upon the accuracy. Making the model by adding more LSTM layers also contributes to a more accurate model. On the other hand, for the free text Gaussian Mixture Model, varying the threshold values, we can observe that the EER achieved would roughly be around $0.2 - 0.25$. Applying a model similar to the RNN model for the fixed text dataset on the free text dataset, we observe a much more diminished accuracy.

This highlights the challenges in trying to classify a free text dataset which is much more complicated than the fixed text one. It indicates that even more refined algorithms may have to be used for satisfactorily being able to classify free text datasets and why this problem still garners interest.

VII. Individual Contributions

Saurav Mehra

- Preprocessed the datasets
- Implemented Filtered Manhattan Distance model on the fixed text dataset
- Implemented the Recurrent Neural Network Model on the fixed text dataset

Ankur Bohra

- Implemented Gaussian Mixture Model on the fixed and free text datasets
- Implemented the Feed-Forward Neural Network model on the fixed text dataset

References

[1] Y. Zhong and Y. Deng, “A survey on Keystroke Dynamics Biometrics: Approaches, Advances and Evaluations,” TheScientificWorldJournal, vol. 2013, p. 408280, 11 2013.

[2] K. Killourhy and R. Maxion, ”Keystroke Dynamics - Benchmark Data Set,” Accompaniment to ”Comparing Anomaly-Detection Algorithms for Keystroke Dynamics” (DSN-2009).

[3] H. Ceker and S. Upadhyaya, ”Enhanced recognition of keystroke dynamics using Gaussian mixture models,” in MILCOM 2015 - 2015 IEEE Military Communications Conference. IEEE, 10 2015, pp. 1305– 1310.

[4] Y. Sun, H. Ceker and S. Upadhyaya, ”Shared Keystroke Dataset for Continuous Authentication”.

[5] J. Malmström and H. Lindström, ”Typing Biometrics for User Authentication - a One-shot Approach”.

[6] A. Acien, A. Morales, J.V. Monaco, R. Vera-Rodriguez, J. Fierrez, ”TypeNet: Deep Learning Keystroke Biometrics”.

[7] R. Spillane, ”Keyboard Apparatus for Personal Identification”.

[8] P. Kang, S. Hwang and S. Cho, ”Continual Retraining of Keystroke Dynamics based Authenticator”.

[9] W. Martono, H. Ali, and M.J.E. Salami, ”Keystroke Pressure-based Typing Biometrics Authentication System Using Support Vector Machines”..

[10] W. Chen and W. Chang, ”Applying Hidden Markov Models to Keystroke Pattern Analysis for Password Verification”.

[11] K. Bakhtiyari, M. Taghavi, and H. Husain, ”Implementation of emotional-aware Computer Systems using typical Input Devices”.

[12] A. Acien, A. Morales, J.V. Monaco, R. Vera-Rodriguez, J. Fierrez, ”TypeNet: Scaling up Keystroke Biometrics”.