Deep Secure: A Fast and Simple Neural Network based approach for User Authentication and Identification via Keystroke Dynamics

Saket Maheshwary, Soumyajit Ganguly, Vikram Pudi
Data Sciences and Analytics Center, Kohli Center on Intelligent Systems
International Institute of Information Technology Hyderabad, India
{saket.maheshwary, soumyajit.ganguly}@research.iiit.ac.in, vikram@iiit.ac.in

Abstract

In this paper we investigate the problem of user authentication and identification based on their keystroke timing patterns. To this end, we propose **Deep Secure**: a neural network architecture to authenticate a specific user or identify different users based on their keystroke statistics. A portion of prior literature try to tackle this problem using manually engineered feature combinations while others use complex and hard to train (time-consuming) models like Deep Belief Networks. Our proposed approach is computationally upto 10 times faster when compared with previous neural network models and does not require any manually engineered features. We show how Deep Secure can be used for both user authentication and user identification problem settings with a simple change separating the models. The efficacy (both training and testing) and effectiveness of our proposed model is demonstrated through comparisons with existing methods using the CMU keystroke dynamics benchmark dataset. We achieved an Equal Error Rate(EER) of **0.030**, (12.45% improvement) for the user authentication task and an overall accuracy of **93.59%** (12.39% improvement) for user identification over state-of-the-art techniques.

1 Introduction

In this era where everyone wants secure, faster, more reliable and easy to use means of communication, there are many instances where user information such as personal details and passwords get compromised thus posing a threat to system security. In order to tackle the challenges posed on the system security biometrics, recent research [Giot *et al.*, 2010] prove to be a vital asset. Biometric systems are divided into two classes namely physiology based ones and behavior based ones. Physiology based approach allows authentication via use of retina, voice and fingerprint touch. In contrast, behavior based approach includes keystroke dynamics on keyboard or touch screens and mouse click patterns.

In this paper we propose Deep Secure: an algorithm to deal with *keystroke dynamics* – a behavior based unique timing patterns in an individuals typing rhythm which is used as a protective measure. These rhythms and patterns of tapping are idiosyncratic [A. Dvorak and Ford, 1936] the same way as hand writings or signatures are, due to their similar governing neuro-physiological mechanisms. Back in the 19th century, telegraph operators could recognize each other based on ones specific tapping style [Leggett and Williams, 1988]. Based on the analysis of the keystroke timing patterns, it is possible to differentiate between actual user and an intruder. By keystroke dynamics we refer to any feature related to the keys that a user presses such as key down time, key up time, flight time etc. In this paper, we concentrate on authenticating and identifying users based on static text such as user password. The mechanism of keystroke dynamics can be integrated easily into existing computer systems as it does not require any additional hardware like sensors thus making it a cost effective and user friendly technique for authenticating and identifying users with high accuracy.

It is appropriate to use keystroke dynamics for user authentication and identification as studies [Syed *et al.*, 2011; 2014] have shown that users have unique typing patterns and style. Moreover, [Syed *et al.*, 2011; 2014] has proven some interesting results in their research work as well. In their first hypothesis, they proved that the users present significantly dissimilar typing patterns. Second, they have shown details about the relationship between users occurrence of sequence of events and their typing style and ability. Then they explained sequence of key up and key down events on the actual set of keys. They have also shown that there is no correlation between users typing skills and the sequence of events. Hence all these factors make it difficult for intruders to match with the actual users typing patterns. Keystroke dynamics is concerned with users timing details of typing data and hence various features could be generated from these timing patterns. In this paper we are using timing features only on static text.

Neural network based models are frequently used off late in the field of computer vision, speech signal processing and text representation. They are now being adopted in the domains of security as well [Deng and Zhong, 2013]. Unlike classical methods which rely on complex distance metrics or manual feature engineering, these neural network based techniques have multiple advantages over the previous both in task specific performance and scalability. Motivated by the superior results obtained by the neural network models, we present Deep Secure for authenticating and identifying users based on keystroke timing pattern. Our main contributions in this paper are as follows:

- We propose Deep Secure, a novel neural network architecture to perform user-based keystroke authentication and identification.
- The design of our proposed model is simple, as it is based on a simple feed-forward neural network. This makes it easy to interpret, implement, maintain, embed and modify as the situation demands.

2 Related Work

Classifying users based on keystroke timing patterns has been in limelight when [Forsen *et al.*, 1977] first investigated whether users could be distinguished by the way they type on keyboard. Researchers have been studying the user typing patterns and behavior for identification. Then [Gaines *et al.*, 1980] investigated the possibility of using keystroke timings as to whether typists could be identified by analyzing keystroke times as they type long passages of text. Later [Monrose and Rubin, 2000] extracted keystroke features using the mean and variance of digraphs and trigraphs. A detailed survey was published [Peacock *et al.*, 2004] on the keystroke dynamics literature using the Euclidean distance metric with Bayesian like classifiers. Initially [Bergadano *et al.*, 2002] and later [Gunetti and Picardi, 2005] proposed the usage of relative order of duration times for different n-graphs to extract keystroke features that was found to be more robust to the intra-class variations than absolute timing. Great results for text-free keystroke dynamics identification was published [Gunetti and Picardi, 2005] where the authors merge relative and absolute timing information on features.

[Zhong *et al.*, 2012] proposed a new distance metric by combining Mahalanobis and Manhattan distances. Many machine learning techniques have also been proposed for keystroke dynamics as an authentication system. These vary from the likes of Decision Trees [Brieman *et al.*, 1984], Support Vector Machines [Cortes and Vapnik, 1995], Neural Networks [Kubat, 1999], Nearest Neighbor Algorithms [Cover and Hart, 1967] and Ensemble Algorithms [Schapire, 1999] among others. Recently [Deng and Zhong, 2013] introduced two new algorithms namely the Gaussian mixture model with the universal background model (GMM-UBM) and the deep belief nets (DBN). Unlike most existing approaches, which only use genuine users data at training time, these two generative model-based approaches leverage data from background users to enhance the models discriminative capability without seeing the imposters data at training time.

One problem faced by researchers working on these type of problems is that majority of the researchers are preparing their own dataset by collecting data via different techniques and the performance criteria is not uniform as well hence comparison on similar grounds among the proposed algorithms becomes a difficult task. To address this issue, keystroke dynamics benchmark dataset is publicly provided with performance values of popular keystroke dynamics algorithms [Killourhy and Maxion, 2009] to provide a standard universal experimental platform. [Killourhy and Maxion, 2009] collected and published a keystroke dynamics benchmark dataset containing 51 subjects with 400 keystroke timing patterns collected for each subject. Besides, they also evaluated fourteen available keystroke dynamics algorithms on this dataset, including Neural Networks, KNNs, Out-lier Elimination, SVMs etc. Various distance metrics including Euclidean distance, Manhattan distance and Mahalanobis distance were used. This keystroke timing pattern dataset along with the evaluation criteria and performance values stated provides a benchmark to compare the progresses of new proposed keystroke timing pattern algorithms on same grounds.

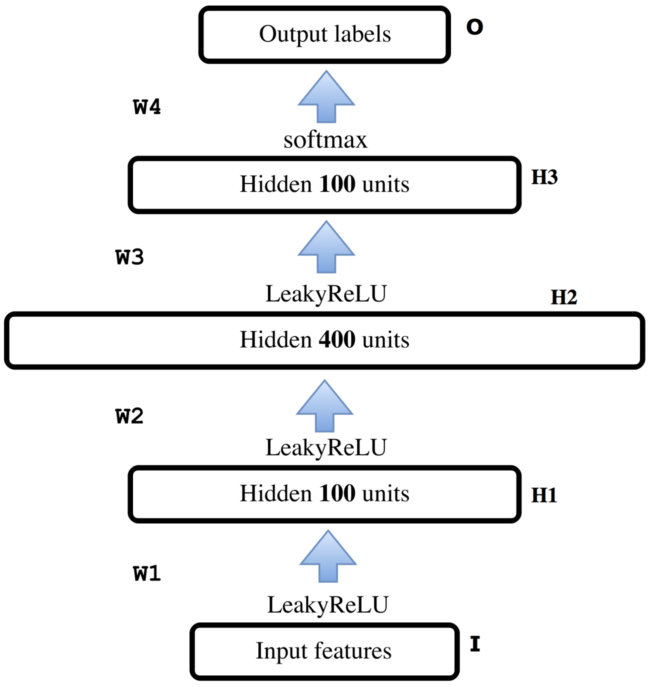


Figure 1: Illustration of deep secure architecture. Here **W1**, **W2**, **W3** and **W4** represents the weight matrices which are our parameters to be learnt. **H1**, **H2** and **H3** are the hidden layers of varying sizes. **I** and **O** are input and output layers respectively.

3 Deep Secure

We first formally define our neural-network architecture which we use as a binary classification problem - given a keystroke pattern and an user, does this pattern belong to this user or not. Towards the end of this section we discuss various configurations of our neural network and how to use them in the keystroke based user identification setting.

Let us define our input feature vectors as $x \in \mathbb{R}^D$ and our labels as $y \in [1, 0]$. Our input training data are sampled either from one particular user or impostors (any other user in the dataset). Similarly the training labels are 1 (user) or 0 (impostor) accordingly. We aim to learn a complex, non-linear hypothesis $h_{W,b}(x)$ that fits to our data. Here W is the weight matrix which we learn and b the bias term. As can be deciphered from Figure 1, our network has 4 weight matrices W_1, W_2, W_3, W_4. All the hidden layers and our input layer is followed by the rectified linear activation function given by $f(z) = max(0, x)$. We define the loss by the negative log-likelihood function which maximized the probability that sample $x^{(i)}$ gets classified as user or impostor - formally shown in Equation 1 where W represents the parameters for our model. Learning is done through back-propagation of the losses through our network [Hecht-Nielsen, 1989].

$$P(y^{(i)} = 1|x^{(i)}; W) = \frac{1}{1 + exp(-W^T.x^{(i)})} \qquad (1)$$

The above explained architecture can be used for a particular user authentication. We need to train n such models for n users in the database. Further details are provided in our training subsection below. Besides this our model can be slightly tuned to represent a different scenario of user identification. In this problem setting, we need to identify a particular user from a large pool of given users. From binary classification, now we have a multiclass classification problem. We can change the last layer of our neural network from 2 dimensions to n dimensions and use a soft-max activation function before our output layer as shown in Figure 1. This results in maximizing the probability of training sample $x^{(i)}$ getting classified as label $y^{(i)}$. The corresponding loss function is described in Equation 2. Training only one model is sufficient considering we only come across users from our defined training set.

$$P(y^{(i)} = k|x^{(i)}; W) = \frac{exp(W^{(k)T}x^{(i)})}{\sum_{j=1}^{K} exp(W^{(j)T}x^{(i)})} \qquad (2)$$

3.1 Architecture

At a high-level we are using a simple feed-forward neural network with a input layer with the size of our feature vectors and three hidden layers of 100, 400 and 100 dimensions respectively. All the layers are fully connected. We gradually increase our hidden layer size followed by a gradual deceasing to the output layer. The higher size of hidden layers introduces sparsity in our network and helps in capturing the inter-feature relations which might be present. This stacking up of hidden layers add hierarchy and compositionality to the feature interactions. By this process we eliminate the need of manual feature engineering as we let our network take care of it. As is shown later in the Results section, this model of ours is more robust and less prone to overfitting on this key-stroke recognition problem when compared to a simpler 1 hidden layer model. Following subsections explain other building blocks of Deep Secure. We later discuss ablation studies for each in Table 3.

3.2 Dropout

We use dropout [Srivastava *et al.*, 2014] after our hidden layers which act as a regularizer and restricts over-fitting. During our training stage we randomly delete the nodes of each hidden layer with a certain probability $p[0, 1]$ for each input sample. These neurons do not participate in the back-propagation learning. In testing time, the weights are correspondingly divided by $1 - p$. Using dropout, forces the rest of the neurons in the hidden layers to learn more robust features and depend lesser on other specific neurons. In [Srivastava *et al.*, 2014] more details are provided which show that using dropout can be an economic alternative to ensembling various network architectures.

3.3 Batch Normalization

After every fully-connected layer, we use batch normalization [Ioffe and Szegedy, 2015] before the respective activation functions. Using batch normalization we monitored the gap between training and testing loss over epochs narrowed down. This led to better generalization.

3.4 Leaky ReLU

Non-linear function Rectified linear unit (ReLU) is preferred to sigmoid or hyperbolic-tan because it simplifies backpropagation, makes learning faster while also avoiding saturation. However for large gradients, ReLU can cause particular neurons to die and not participate in learning at-all [Maas *et al.*, 2013]. LeakyReLU's have a small positive gradient $f(z) = max(0.01x, x)$ which prevent this dying of a neuron. We applied Leaky ReLU as our activation function after the fully connected layers.

3.5 Adam

In recent times, several algorithms (with implemented software tools) are available for training a deep neural network. While stochastic gradient descent (SGD) for quite some time have been the top choice, there has been study which indicate some of the obvious flaws [Le *et al.*, 2011] in the vanilla implementation. There have been some attempts to automatically tune its learning rate thus resulting in much faster convergence. For Deep Secure we use Adam [Kingma and Ba, 2014] instead of SGD which required a lot of fine-tuning with the learning rate and over 500 epochs to converge.

3.6 Inputs to Deep Secure

The dataset [Killourhy and Maxion, 2009] provides three types of timing information namely the hold time, key down-key down time and key up-key down time. Following are the details of three categories of timing features which is used to generate features using keystroke timing dataset [Killourhy and Maxion, 2009]. Figure 2 illustrates various timing features given as input to Deep Secure where up arrow indicates key press and down arrow indicates key release. The dataset consists of keystroke timing information of 51 users, where each user is made to type **.tie5Roanl** as password. All the 51 users enrolled for this data collection task typed the same password in 8 different sessions with 50 repetitions per session thus making each user to type 400 times in total.

- **Hold Time** also known as dwell time, is the duration of time for which the key is held down i.e. the amount of time between pressing and releasing a single key. In figure 2, H_i represents the hold time. Hold time contributes to *eleven* features (where *ten* features are corresponding to the ten characters of password and *one* feature corresponds to the return key).
- **Down-Down Time** key down key down time is the time from when key1 was pressed to when key2 was pressed.

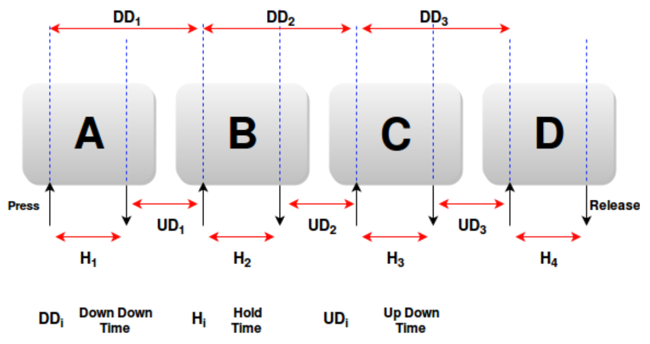


Figure 2: Illustration of keystroke timing features given as inputs to Deep Secure.

In figure 1, the times DD_i depicts the down down time. It contributes to *ten* features in our feature space.

- **Up-Down Time** key up key down time is the time from when key1 was released to when key2 was pressed. This time can be negative as well. In figure 1, the times UD_i depicts the up down time. It contributes to *ten* features in our feature space.

Name	Specification
Dropout 1	0.3
Dropout 2	0.5
Dropout 3	0.3
LeakyReLU	0.01
Adam	0.001
Epochs	200

Table 1: Hyperparameters used in Deep Secure

4 Experimental Setup

In this section we discuss the experimental setup, evaluation criteria used and the performance of our proposed model. We evaluated our approach on the CMU keystroke dynamics benchmark dataset [Killourhy and Maxion, 2009] where *51* users were designated for this task. We demonstrate the effectiveness of our model with the average equal error rate (EER) [Killourhy and Maxion, 2009] and accuracy. We compare the results with various proposed anomaly detectors/classifiers which have been used in literature. We used the python library Keras [Chollet, 2015] for building our neural network architecture. All our experiments were carried out on a Pentium 4^{th} generation machine with $4GB$ memory.

4.1 Training

We frame keystroke dynamics based authentication both as a one-class and multi-class classification problem.

User Authentication: For authentication, Deep Secure learns one model per user, rejects anomalies to the learned model as impostors, and accepts in-liers as the genuine user. Consider a scenario in which a users long-time password has been compromised by an impostor. We assume the user to be practiced in typing their own password, while the impostor is unfamiliar with it (e.g., typing it for the first time). We measure how well each of our detectors is able to discriminate between the impostors typing and the genuine users typing in this scenario. We start by designating one of our 51 subjects as the genuine user, and the rest as impostors. We train an anomaly detector by extracting 200 initial timing feature vectors for a genuine user from the dataset. We repeat this process, designating each of the other subjects as the genuine user in turn thus creating models equal to number of distinct subjects or users. Unlike most existing approaches, which only use actual users data at training time, our model leverage data from background users to enhance the models discriminative capability thus improving the prediction performance. We randomly took 5 samples from each background users as negative samples. Note that these 5 random samples were carefully chosen such that no impostor samples that were used in testing were shown during the time of training. For this problem setting, we use the evaluation criteria as mentioned in [Killourhy and Maxion, 2009].

User recognition: This scenario can be viewed as a muticlass classification problem where we need to identify any particular user from a given pool of users. We train only 1 model which learns to discriminate between all the users from our dataset. For experiments, we took the same 200 initial timing feature vector per-user as before. 10% of training data was kept aside as validation data for hyper-parameter tuning.

4.2 Testing

Authenticate Users: We take last 200 passwords typed by the genuine user from the dataset. These 200 timing feature vectors acts as test data. Scores generated in this step acts as the user scores. Next, we take initial 5 passwords typed by each of the 50 impostors (i.e., all subjects other than the genuine user) from the dataset which acts as the impostor scores. Thus we form a test dataset of 200 positive samples and 250 negative samples per user which we provide to Deep Secure and record the output predictions. If s denotes the predictions, the corresponding anomaly score was calculated as $1-s$ [Killourhy and Maxion, 2009]. Intuitively, if s is close to 1.0, the test vector is similar to the training vectors, and with s close to 0.0, it is dissimilar.

Identify Users: For the muticlass classification problem setting as described above, we test our model by taking the last 200 timing feature vector of each user and combine them to form our test set. Note that here we need not explicitly provide our model with positive and negative samples as the neural network handles it by design of softmax loss function. We need to perform only a forward pass with all features of the full test-set and get the corresponding predictions per sample. We calculate the final accuracy of our network which measures the percentage of correctly classified user based on their input features.

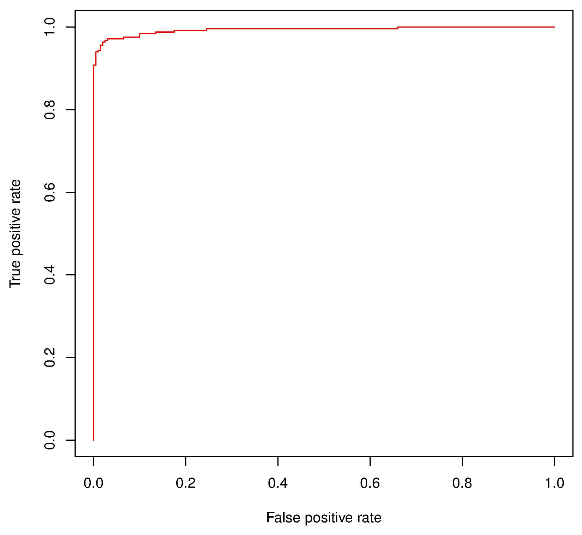


Figure 3: ROC curve for 6th user with $EER = 0.030$

Model architecture	Accuracy	Time taken (min)
Base model	85.22	4.61
with Dropout	86.69	7.18
with LReLU	87.05	4.13
with Batch-Norm	90.52	8.62
Deep Secure	**93.59**	9.18

Table 2: Ablation study of our neural network model showcasing the increase of performance for each additional component we use. Note that for this test we used tan-h activation when we opt not to use LeakyReLU. All components are added independently on the base model. Accuracy is based on the multiclass user recognition problem.

4.3 Empirical Evaluation for user authentication

Based on the genuine user scores and impostor scores generated in the steps above, we generate the ROC curve for the actual (genuine) user. Then we calculate the equal error rate from the ROC curve where the equal error rate corresponds to that point on the curve where the false positive rate (false-alarm) and false negative rate (miss) are equal. We repeat the above four steps, in total of 51 times where every time each of the subsequent user is taken as the genuine user from the 51 distinct users in turn, and calculate the equal-error rate for each of the genuine users. Finally we compute the mean of all 51 equal-error rates which gives us the performance value for all users, and the standard deviation which will give us the measure of its variance across subjects. In order to ensure comparison on same grounds we have used exactly the same evaluation criteria as stated by [Killourhy and Maxion, 2009] on our proposed approach.

4.4 Empirical Evaluation for user identification

For this evaluation, we use model accuracy from our network which effectively gives an average of correctly classified users. We tuned all the hyper-parameters for our model using 5-fold cross-validation within our training data. The final values are reported in Table 3. With our reported hyper-parameters, our training and validation loss converge at around 200 epochs. We then combine our training and validation data, train our model for 200 epochs and evaluate it using the test-samples.

5 Results

Figure 3 shows ROC curve for different users with their Equal Error Rate (EER) value and user number where user number corresponds to the user as stated in CMU dataset [Killourhy and Maxion, 2009]. Table 4 shows the comparison of 16 other proposed keystroke timing algorithms with Deep Secure. This comparison is performed on the same dataset and evaluation criteria thus assuring an objective comparison. Deep Secure is able to achieve an average equal error rate (EER) of ***0.30*** and with a standard deviation (stddev) of 0.024 across 51 subjects. The average equal error rate (EER) shown in the table below are the fractional rates between 0.0 and 1.0, not the percentages. From Table 4 it can be inferred that Deep Secure performs superior than other proposed techniques in comparison.

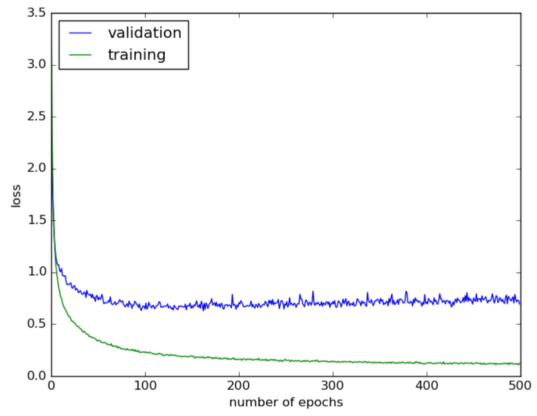
Model/Algorithm	Accuracy
Deep Secure	**93.59**
Manhattan(scaled)	81.20
Nearest Neighbor(Mahalanobis)	73.60
Random Forest	69.67
SVM	66.40
Deep Belief Nets (DBN)	65.60

Table 3: Comparison with other top proposed models in terms of Accuracy for user identification

While using a simpler 3-layer model (31-400-51) for this problem, we observe overfitting characteristics using our held-out validation dataset. This can be observed in Figure 4. The training loss approaches 0 much faster and after 100 epochs, the validation loss slowly start increasing. There is also a noticeably larger gap between the training and testing losses. Gradually increasing and decreasing the hidden layer sizes, (also adding more hidden layers) helps alleviate this problem and we can see a clear convergence of both the training and validation loss in Figure 5. The final 51-class accuracy obtained with the 3-layer was 80.11% while with our 5-layer model was 83.59% on the testing set. A faster learning rate for Adam helps in reaching the desired accuracy within 50 epochs but results are quite fluctuating both on the validation and testing data. For the activation function, LeakyReLU relatively gave minor improvements over ReLU but performed quite better than sigmoid and hyperbolic-tan as can be seen in Table 2, our ablation study. Using both Batch-Normalization and LeakyReLU led to significant increase in overall accuracy. Using dropout, though the increment was less, it helped our model generalize well as training

Model/Algorithm	Average EER (stddev)	Source
Deep Secure	**0.030 (0.024)**	
Deep Belief Nets (DBN)	0.035 (0.027)	[Deng and Zhong, 2013]
GMM-UBM	0.055(0.052)	[Deng and Zhong, 2013]
Median Vector Proximity	0.080 (0.055)	[Al-Jarrah, 2012]
Manhattan-Mahalanobis(No Outlier)	0.084 (0.056)	[Zhong *et al.*, 2012]
Manhattan-Mahalanobis(Outlier)	0.087 (0.060)	[Zhong *et al.*, 2012]
Manhattan(scaled)	0.0962 (0.0694)	[Killourhy and Maxion, 2009]
Nearest Neighbor(Mahalanobis)	0.0996 (0.0642)	[Killourhy and Maxion, 2009]
Outlier Count(z-score)	0.1022 (0.0767)	[Killourhy and Maxion, 2009]
SVM (one-class)	0.1025 (0.0650)	[Killourhy and Maxion, 2009]
Mahalanobis	0.1101 (0.0645)	[Killourhy and Maxion, 2009]
Manhattan (Filter)	0.1360 (0.0828)	[Killourhy and Maxion, 2009]
Neural Network(Auto-assoc)	0.1614 (0.0797)	[Killourhy and Maxion, 2009]
Euclidean	0.1706 (0.0952)	[Killourhy and Maxion, 2009]
Fuzzy Logic	0.2213 (0.1051)	[Killourhy and Maxion, 2009]
k Means	0.3722 (0.1391)	[Killourhy and Maxion, 2009]
Neural Network(Standard)	0.8283 (0.1483)	[Killourhy and Maxion, 2009]

Table 4: Comparison of 16 different keystroke timing pattern algorithms in terms of Average Equal Error Rate(EER)



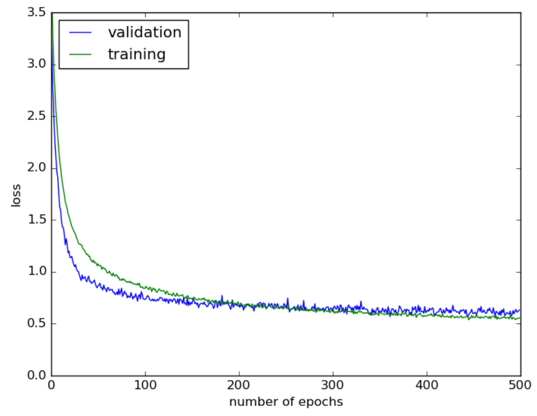


Figure 5: Train and Validation loss over epochs for our proposed 5-layer (31-100-400-100-51) neural network.

loss was kept in check. Removing the dropout after hidden layers result in heavy overfitting on our 5-layer model. These improvements came at a cost of training time as we can see in Table 2 our final model train-time almost doubled after addition of all the components. Still due to the relatively small size of our dataset, total training time was about 9 minutes. For comparison on the same dataset, it took over 65 minutes to train the Deep Belief Network model proposed in [Deng and Zhong, 2013] on same hardware.

6 Conclusion and Future Work

In this paper we introduced Deep Secure, a neural network based approach for user authentication via keystroke timing patterns. Deep Secure employs the latest advances in deep learning and optimization techniques to automatically learn feature interactions in the keystroke data. We showcase the two different scenarios where our model can be used with a slight change in its architecture. Experiments on the CMU dataset, demonstrate the superiority of our proposed neural network model over all other methods by a big margin in both the evaluation criterion. We plan on extending our models to other available datasets on this domain. We would also like to investigate if transfer learning can help with user authentication and identification for large pool of users when trained from a limited dataset.

References

[A. Dvorak and Ford, 1936] W. Dealey A. Dvorak, N. Merrick and G. Ford. Typewriting behavior. 1936.

[Al-Jarrah, 2012] Mudhafar M Al-Jarrah. An anomaly detector for keystroke dynamics based on medians vector proximity. *Journal of Emerging Trends in Computing and Information Sciences*, 3(6):988–993, 2012.

[Bergadano *et al.*, 2002] Francesco Bergadano, Daniele Gunetti, and Claudia Picardi. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):367–397, 2002.

[Brieman *et al.*, 1984] L Brieman, JH Friedman, R Olshen, and C Stone. Classification and regression trees (belmont, ca: Wadsworth), 1984.

[Chollet, 2015] Franois Chollet. keras. https://github.com/fchollet/keras, 2015.

[Cortes and Vapnik, 1995] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

[Cover and Hart, 1967] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[Deng and Zhong, 2013] Yunbin Deng and Yu Zhong. Keystroke dynamics user authentication based on gaussian mixture model and deep belief nets. 2013.

[Forsen *et al.*, 1977] George E Forsen, Mark R Nelson, and Raymond J Staron Jr. Personal attributes authentication techniques. Technical report, DTIC Document, 1977.

[Gaines *et al.*, 1980] R Stockton Gaines, William Lisowski, S James Press, and Norman Shapiro. Authentication by keystroke timing: Some preliminary results. Technical report, DTIC Document, 1980.

[Giot *et al.*, 2010] Romain Giot, Baptiste Hemery, and Christophe Rosenberger. Low cost and usable multimodal biometric system based on keystroke dynamics and 2d face recognition. In *ICPR*, 2010.

[Gunetti and Picardi, 2005] Daniele Gunetti and Claudia Picardi. Keystroke analysis of free text. *ACM Transactions on Information and System Security (TISSEC)*, 8(3):312–347, 2005.

[Hecht-Nielsen, 1989] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, pages 593–605 vol.1, 1989.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[Killourhy and Maxion, 2009] Kevin S. Killourhy and Roy A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *DSN*, 2009.

[Kingma and Ba, 2014] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[Kubat, 1999] Miroslav Kubat. Neural networks: a comprehensive foundation by simon haykin, macmillan, 1994, isbn 0-02-352781-7. *Knowledge Eng. Review*, 13:409–412, 1999.

[Le *et al.*, 2011] Quoc V. Le, Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, and Andrew Y. Ng. On optimization methods for deep learning. In *ICML*, 2011.

[Leggett and Williams, 1988] John Leggett and Glen Williams. Verifying identity via keystroke characterstics. *International Journal of Man-Machine Studies*, 28(1):67–76, 1988.

[Maas *et al.*, 2013] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. 2013.

[Monrose and Rubin, 2000] Fabian Monrose and Aviel D Rubin. Keystroke dynamics as a biometric for authentication. *Future Generation computer systems*, 16(4):351–359, 2000.

[Peacock *et al.*, 2004] Alen Peacock, Xian Ke, and Matthew Wilkerson. Typing patterns: A key to user identification. *IEEE Security & Privacy*, 2(5):40–47, 2004.

[Schapire, 1999] Robert E. Schapire. A brief introduction to boosting. In *IJCAI*, 1999.

[Srivastava *et al.*, 2014] 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*, 15:1929–1958, 2014.

[Syed *et al.*, 2011] Zahid Syed, Sean Banerjee, Qi Cheng, and Bojan Cukic. Effects of user habituation in keystroke dynamics on password security policy. In *HASE*, 2011.

[Syed *et al.*, 2014] Zahid Syed, Sean Banerjee, and Bojan Cukic. Leveraging variations in event sequences in keystroke-dynamics authentication systems. In *HASE*, 2014.

[Zhong *et al.*, 2012] Yu Zhong, Yunbin Deng, and Anil K. Jain. Keystroke dynamics for user authentication. In *CVPR*, 2012.