Authio: Neural Network Authentication using Biometric Keystroke Dynamics

Alex Langshur (adl@stanford.edu)
Harry Mellsop (hmellsop@stanford.edu)
https://github.com/Junior-House/authio
CS230 – Deep Learning
Stanford University

Abstract

In 2017, over three billion passwords were stolen [8], compromising enormous quantities of sensitive data. In order to augment the security flaws presented by traditional password-based security systems, we applied a deep neural network architecture to leverage the subtle biometric variation in the typing patterns between individuals. We sought to determine the authenticity of a password attempt based on both the accuracy of the password entered, and on the manner in which the user typed the password. Our results indicate a high level of performance, including (crucially) a 0% false-positive rate, 98% overall accuracy, and a 3% false-negative rate, achieved through a 7-layer neural network architecture.

1 Introduction

While passwords have formed the backbone of day-to-day security systems for decades, in many respects they are extremely flawed. An average of 95 passwords are stolen per second globally, causing the loss of valuable data and billions of dollars in damages yearly [2] [5].
To augment this flawed system, we looked at other forms of security that would not require further investment in hardware. Through research, we ascertained that the typing habits of individuals vary subtly across the population, and as such could be used to discern people by the 'biometric signature' that their typing represents. With an adequate model, we could therefore determine the validity of a purported 'real' user by the manner in which they type their password.
To be explicit, we propose a system where a user types their password into the machine once. The machine will identify whether or not the user is who they say they are, with a high degree of precision and accuracy. We will seek to minimise, in particular, the false-positive rate, as this would correspond to a false user being allowed into the system accidentally. In order to train the system, the user would have to enter their password more than once.

2 Related Work

In order to inspire the structure of our neural network approach, we conducted a review of existing research in the field. According to our research, Forsen et al. were the first team to investigate the feasibility of differentiating users by their typing practices [4]. Moskovitch et al. further formalised the problem statement in their 2009 paper to the IEEE International Conference on Intelligence and Security Informatics [10]. Their paper was an effort to introduce biometric measures that relied on existing hardware. They identified that the factors of keyboard fight time, dwell time, frequency of typing errors and use of particular control keys were the most indicative of typing habits. As such, this guided our selection of typing features to examine.

With respect to specific models, Sungzoon Cho and colleagues were able to achieve a 0.0% false acceptance rate and a 1% false recognition rate using a multilayer perceptron neural network [3]. These results were replicated by Daw-Tung Lin, who found a 0.0% false acceptance rate and a 1.1% false rejection rate. This was therefore our target to beat.

In practical applications, biometric keystroke identification has been employed by Coursera – one of the largest online course vendors globally. They describe their "Signature Track" system in a 2013 paper, in which students type a short phrase to authenticate their identity before submitting an online assessment [1] [9]. However, Coursera's product was discontinued in 2017, allegedly due to inaccuracies in prediction and reports about the procedure being inconvenient.

Therefore, it is clear that the latent structure in the data exists – in a controlled setting, it has been demonstrated that with a sufficiently sophisticated model users can be identified almost perfectly. However, in practice, implementations have proven to be more challenging. We hypothesise that when constrained to only password entry on a particular machine, and with the correct model, we will be able to alleviate these practical problems.

3 Dataset and Features

3.1 Data Structure

Based on our observations from the work of Moskovitch et al, raw keystroke data will be parameterised by a sequence of key events and durations. For a sequence of n keystrokes, we collect three data points for each pair of adjacent keys k and $k+1$ in a sequence. These are: the 'Hold' time, from when key k is pressed to when key k is released; the 'Up-Down' time, from when key k is released to when key $k+1$ is pressed; and the 'Down-Down' time, from when key k is pressed to when key $k+1$ is pressed. This scheme was inspired by the work of Killourhy, Maxion, and Moskovitch to correspond with their dataset [10] [7].

For a trivial case where our password is `matt`, the data is represented in code as the following. Here, t_{ij} denotes the value of data point j of character index i for the password example $s =$ `"matt"`:

$$\text{Data}(s) = \left\{ (m, \text{None}, H) = t_{11}, (a, m, DD) = t_{12}, (a, m, UD) = t_{13}, \ldots, (t, t, UD) = t_{43} \right\}$$

If there are no repeated sequences of 2 characters in the string, we find that the size of the data for one attempt will be $\left|\text{Data}(s)\right| = 3 \cdot \texttt{len}(s)$. For passwords that repeat features, such as `"mattt"`, the duplicate features will map to the mean of all matching keystoke pattern times. In these cases, $|\text{Data}(s)| < 3 \cdot \texttt{len}(s)$, though for reasonably non-repetitive passwords we found that this was not an issue. In our case, the password we used for training and testing (`.tie5Roanl`) had no repeated sequences. We developed this method of data capture during our previous logistic regression based attempt at solving this problem.

3.2 Data Collection Preprocessing

We made use of the CMU Keystroke Dynamics Benchmark Dataset, which contains password attempts from 51 users, each of whom typed the password `.tie5Roanl` 400 times. This data is specific to a singular password and key sequence (e.g. the participants had to type the exactly correct keystrokes or the sample would be rejected).

Moreover, we collected data on our own typing habits. Our own personal typing data has been uploaded to the Github repository if it is of interest. We developed a data-gathering system designed to identify and extract data in the same format as that utilised by the CMU Keystroke Dynamics Benchmark Dataset [6]. This piece of software can be found in the `collection` directory of the Github repository, and heavily leverages functionality from the `pynput` Python library.

3.3 Data Usage Strategy

The use of this dataset in our research follows a two-step strategy. In the first step, we will select a single user in the CMU dataset as the "valid user" (i.e. the user that should be accepted by the authentication system). We will train this user's 400 valid password repetitions against the 50 other users, whose password repetitions will all be dubbed "invalid" (i.e. biometric keystroke data that should be rejected by the authentication system). This approach will allow us to quickly and iteratively derive an accurate model that will accept the valid data and reject the invalid data. Then, we will perform this for each user in the database, to analyse the accuracy that is achieved for each individual user. This will allow us to amalgamate the results and extrapolate a notion of the model's performance in general.

In the second step, we will target one of our own collections of typing habit data as the 'valid' data, while using the entirety of the CMU Keystroke Dynamics Benchmark Dataset as 'invalid' data. By using 51 'invalid' users with over 20,000 password repetitions, our goal is to approximate *all* the general typing subtleties that keyboard-users exhibit (clearly, this approximation would be far more powerful with even more data) and explicitly distinguish this general class of habits from our own combinatorial profile of typing habits.

4 Methods

4.1 Baseline Model

As a baseline, we applied an Adam-optimised logistic regression model to this classification problem, which leveraged inference on a high-dimensional hyper-ellipsoid decision boundary that we attempted to construct artificially in our selection of input features. Our results were positive, indicating a sub-20% false-positive rate and 60% overall accuracy. We postulated that with a refined model we could perform better – we believe that this classification problem would benefit strongly from an organically-imposed (self-learned) hyper-ellipsoid decision boundary, through a neural network, rather than through inorganically targeted features.

4.2 Architecture Development

We initially began by exploring exploring a variety of neural network architectures through tuning hyperparameters such as the number of layers, as well as the the number of units per layer. In this stage, the goal was to lower the training set bias as much as possible. One interesting challenge here is that it is very difficult to conceptualise oracle-level performance on this task given that it is not a task humans can perform. As such, we had no 'target' accuracy that we were pursuing – we were only hoping to observe the lowest error rates possible. Through this unguided process, the following model, which uses 13,010 trainable parameters, was derived without a tangible evaluation target:

Layer Type	**Units**	**Parameters**
Input Layer	31	0
Fully-connected Layer	64	2048
Fully-connected Layer	64	4160
Fully-connected Layer	64	4160
Fully-connected Layer	32	2080
Fully-connected Layer	16	528
Output Layer	2	34

This first attempt at an improved model set the initial bar high, with an accuracy of approximately 98%, a recall of 96%, and perfect precision. Despite results that reflect a highly effective classification model, we wanted to reduce the likelihood that a real user would be rejected erroneously, while maintaining perfect precision. As such, we continued tuning and building out the architecture above, and arrived at the following model structure, which uses 68,914 trainable parameters and 416 non-trainable parameters:

Layer Type	**Units**	**Parameters**
Input Layer	31	0
Fully-connected Layer	256	8192
Fully-connected Layer	128	32896
Batch Normalisation	128	512
Fully-connected Layer	128	16512
Fully-connected Layer	64	8256
Batch Normalisation	64	256
Fully-connected Layer	32	2080
Fully-connected Layer	16	528
Batch Normalisation	16	64
Fully-connected	2	34

4.3 Regularisation

After completing our initial neural network model, we noticed a problematic amount of variance in the trained results – a significant mismatch between training and validation set accuracy. To fix this, we employed several regularisation techniques, of which two provided successful regularisation effects without sacrificing bias. To start this process, we added dropout layers after every fully-connected layer (other than the output layer). We initially set the dropout probability parameter to 0.2 for each dropout unit and achieved a next-to-zero variance value.

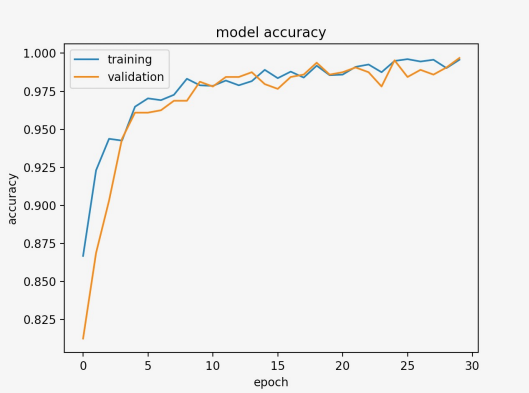
As a means of both regularisation, speeding up training, and improving overall classification accuracy, recall, and precision, we added batch normalisation layers after every other fully-connected layer. This added a total of 960 additional trainable parameters to the model and significantly lowered model bias. Moreover, we found that the large dropout probabilities that we were using, on the order of magnitude of 10^{-1}, were not conducive to effective training

when coupled with batch normalisation. To fix this problem, we removed half of the dropout layers and brought the probabilities of the remaining half down to 0.01 – this provided the best evaluation outcome, while minimising both bias and variance.

The result of this study in tweaking regularisation parameters was a final model with alternating dropout and batch normalisation layers. Adjacent dropout and batch normalisation layers proved to add too much bias, and choosing a single regularisation technique over the other didn't reduce variance enough in the larger network (in actuality, dropout *did* reduce variance enough but required large dropout probabilities that exploded the model bias). The combination and alternating pattern of these two regularisation techniques provide the complete suite of regularisation benefits without most of their drawbacks.

4.4 Training

As a choice of optimiser for training the model, we employed the standard Adam Optimisation algorithm on mini-batches of size 48, with a learning rate of $\alpha = 0.001$ and the standard values for β_1 and β_2. This optimisation algorithm provided better results than vanilla mini-batch, momentum-based, and RMSProp-driven gradient descent.



We allow the model to train for a maximum of 50 epochs. However, we instituted an early-stopping callback on the validation set accuracy with a patience setting of six. The training process usually trains for around 25-30 epochs before stopping.

5 Final Results

Our final model performed extremely well on the dataset. Firstly, when testing over the CMU dataset, we observed the following results for the 'average' user tested (out of 51 independent tests of the same size with different 'valid' users).

Results on CMU Data Only	
True Positives	320/322
True Negatives	318/318
False Positives	0/318
False Negatives	2/322
Accuracy	0.996875
Recall	0.993789
Precision	1.0
F1 Score	0.99688

Clearly, we have significantly outperformed our baseline here. Moreover, note that, in relation to the field of predictive authentication, the overall performance of the model can be considered overwhelmingly successful and is most definitely up to the critical task of classifying valid/invalid users. Crucially, the false-positive rate is 0% – this is imperative to ensure that no adversaries are accidentally granted access to our system. This is additionally reflected in the perfect precision: whenever the model predicts positively (grants user access), it does so correctly (this is far more important than general accuracy or recall). Moreover, the system made an average of two false-negatives out of the 322 positive examples it was shown. This is akin to the system rejecting a real user 0.6% of the time (meaning it may re-prompt for another password attempt).

Moreover, when we analysed the model on our own user gathered data (which is treated as valid user data), in addition to the CMU dataset (which is treated in its entirety as invalid user data), we observed very similar results.

Results on Union of Datasets	
True Positives	315/322
True Negatives	318/318
False Positives	0/318
False Negatives	7/322
Accuracy	0.9890625
Recall	0.978261
Precision	1.0
F1 Score	0.983632

The slight decrease in performance is understandable given that our data will be sampled, naturally, from a slightly different distribution to that of the CMU dataset (given different keyboards, slightly different measuring software/equipment, etc). Crucially, however, we note that we retain our 0% false-positive rate.

6 Discussion & Further Work

We note that our model outperformed that of Sungzoon Cho et al., matching the 0% false-acceptance rate (which, again, is critical in predictive authentication algorithms) but also reducing the false-rejection rate to far below 1%. As such, our DNN model is a confident step forward in the advancement of predictive biometric typing algorithms.

One area for potential improvement is to extend the model to adapt to a password of arbitrary length. Presently, the input to the neural network is a 31-vector representing the features discussed in the Data Structure section. With the relative rigidity of the model to this input size, it would be interesting to explore how it adapts to dramatically different password lengths, with correspondingly different feature vector lengths. Given extra time (and, crucially, extra data), we would like to explore how to adapt the model to deal with this. We postulate that a recurrent neural network would likely be a good choice here, but, with the limitations of available datasets, we were unable to explore this model on varying password lengths (and as such, unable to explore its overall effectiveness).

Moreover, all of our testing was performed with a consistent password – `.tie5Roanl`. While we have no reason to doubt that the effectiveness of the training and modelling technique would extend to arbitrary passwords of similar length, to deploy this system in practice we would need to provide (and analyse) a more diverse range of training passwords with different input features. Specifically, if we were to deploy the software and allow a user to use their password of choice, we would need to have a sufficient corpus of other password attempts to generate adversarial examples to train the model on (i.e. generate 'false' password attempts). Again, with access to better data, we would have been able to explore these possibilities more thoroughly.

Many of these ideas for future improvement are centered around the possibility of commercialising the algorithm presented in this paper. As suggested by most successful products that are adopted en masse, this software would have to be easy to use. This would require that we collect large quantities of valid and invalid user data for a specific password, as suggested in the previous paragraph. Yet, forcing a user to enter so many repetitions of a password would hardly constitute an easy-to-use product. As a result, we will have to introduce a form of latent biometric behavior surveying – where we learn the user's biometric typing subtleties as they casually use their keyboard, or keep training the model as they continue to log in. When a user requests a new password, we can compose a biometric typing profile for this password based on all the surveyed data. This is a single large improvement, among many others, that would be required to convert our academic algorithm to a product.

7 Conclusion

While the security presented by traditional passwords may be flawed, we believe that we have created a compelling augmentation through the use of keystroke biometrics. Our system rejects 100% of false actors on the test data we observed, and as such we posit that it would reject almost all false actors in practice – a noticeable improvement when layered on top of existing password security. Moreover, it is far more convenient and fast than two-factor authentication, while providing comparable or better improvements in false-actor rejection; keystroke biometrics are next to impossible to replicate (in contrast to security questions, or email accounts that are too-often secured by the same password as the system being attacked). As such, while there is still progress to be made, our system provides a compelling improvement that would be, with slight improvements, a meaningful and deployable addition to password security in a myriad of contexts.

8 Contributions

Alex Langshur took primary responsibility for developing the model in Tensorflow. Harry Mellsop was responsible for the initial development of the data gathering software, and for pre-processing the datasets. Of course, we both collaborated on the aforementioned, and wrote our reports together. We would also like to acknowledge Ryan Kearns, as this project builds, in part, on foundational research that we conducted as three earlier this year [6].

References

[1] Coursera, 2019, https://blog.coursera.org/about/.

[2] J. Brown. An average of 95 passwords are stolen every day, 2017, https://www.ciodive.com/news/an-average-95-passwords-stolen-per-second-in-2016-report-says/435204/.

[3] S. Cho, C. Han, D. H. Han, and H.-I. Kim. Web based keystroke dynamics identity verification using neural network, 2000, https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=5BADA02DC9A60E46B907A59EAE39C80F?doi=10.1.1.95.4863rep=rep1type=pdf.

[4] G. Forsen, M. Nelson, and J. R. Staron. Personal attributes authentication techniques. technical report radc-tr-77-333, 1977.

[5] T. Hunt. 86% of passwords are terrible (and other statistics), 2018, https://www.troyhunt.com/86-of-passwords-are-terrible-and-other-statistics/.

[6] R. Kearns, A. Langshur, and H. Mellsop. Biometric keystroke learning for user authentication, 2019.

[7] K. S. Killourhy and R. A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics, 2009, https://www.cs.cmu.edu/ keystroke/KillourhyMaxion09.pdf.

[8] S. Larson. Google says hackers steal almost 250,000 web logins each week, 2017, https://money.cnn.com/2017/11/09/technology/google-hackers-research/index.html.

[9] A. Maas, C. Heather, C. T. Do, R. Brandman, D. Koller, and A. Ng. Moocs and technology to advance learning and learning research: offering verified credentials in massive open online courses, 2014, 10.1145/2591684.

[10] R. Moskovitch, C. Feher, A. Messerman, N. Kirschnick, T. Mustafić, A. Camtepe, B. Löhlein, U. Heister, S. Möller, L. Rokach, and Y. Elovici. Identity theft, computers and behavioral biometrics, 2009, http://www.ise.bgu.ac.il/faculty/liorr/idth.pdf.