Neural Network Authentication using Biometric Keystroke Dynamics

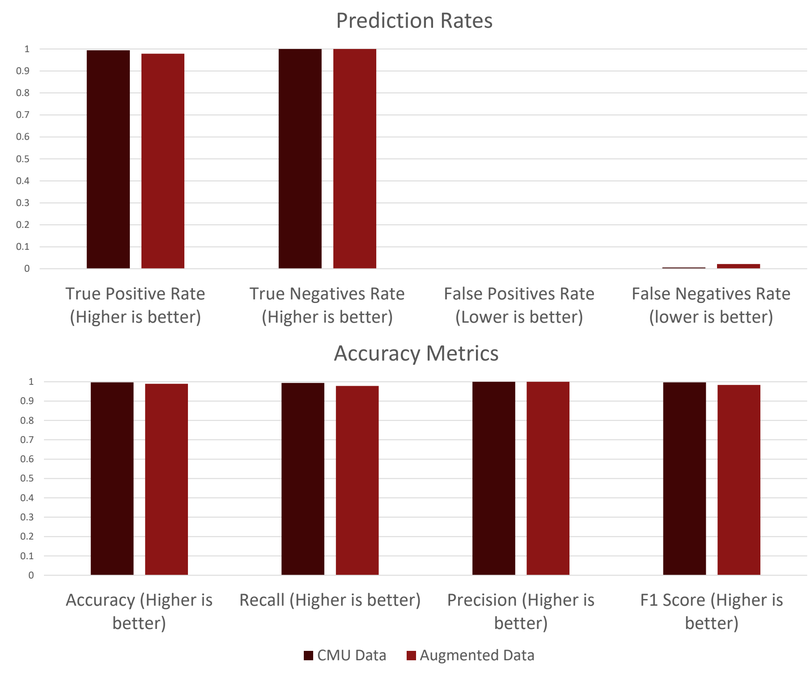
Alex Langshur (adl), Henry Mellsop (hmellsop)
Stanford University, Fall 2019

Abstract

In 2017, over three billion passwords were stolen, 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 dense neural network architecture.

Results

We measure the model's performance on two datasets. One dataset was the CMU Keystroke Dynamics Benchmark Dataset. The other consisted of the CMU dataset merged with data that we collected ourselves. We refer to this as 'Augmented Data'. More information on these datasets can be found in the data collection section.



Clearly, the model performed extremely well across the board. On both datasets, we observe perfect precision, as well as a 0% false positive rate, which is extremely important for this task as this represents 0% admission of false agents into the system.

The discrepancy between the datasets is likely caused by some variation in the data distribution caused by subtle differences between the method through which we capture data, and the method used by the CMU researchers.

However, despite the slight variation in performance between the datasets, the model still performed extremely well and could feasibly be deployed to augment security in practice.

Data Collection & Feature Extraction

We were fortunate to have access to the CMU Keystroke Dynamics Benchmark dataset, which we were able to pre-process into the format outlined below. This dataset consists of 500 password attempts from 51 participants. Ideally, we would have liked more data than this for more thorough validation, but unfortunately, we were unable to find a more comprehensive dataset.

In an effort to acquire more data, and also to demonstrate real-world potential, we also collected our own data using Python's `pynput` library. We combined the CMU data, and data from ourselves, into one augmented dataset.

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

For feature extraction on the raw data, we took inspiration from the work of Moskovitch et al. and settled on the following parameterization. For a sequence of *n* keystrokes we collect three normalized datapoints for each pair of adjacent keys *k* and *k + 1* in a sequence. We collect the 'hold' time (how long a single key is held down), the 'up-down' time (the time between when one key is released, and the next is pressed) and the down-down time (the time from when one key is pressed to when the next is pressed. For the trivial case of a password being `matt`, the above expression represents this encoding.

Discussion & Future 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.

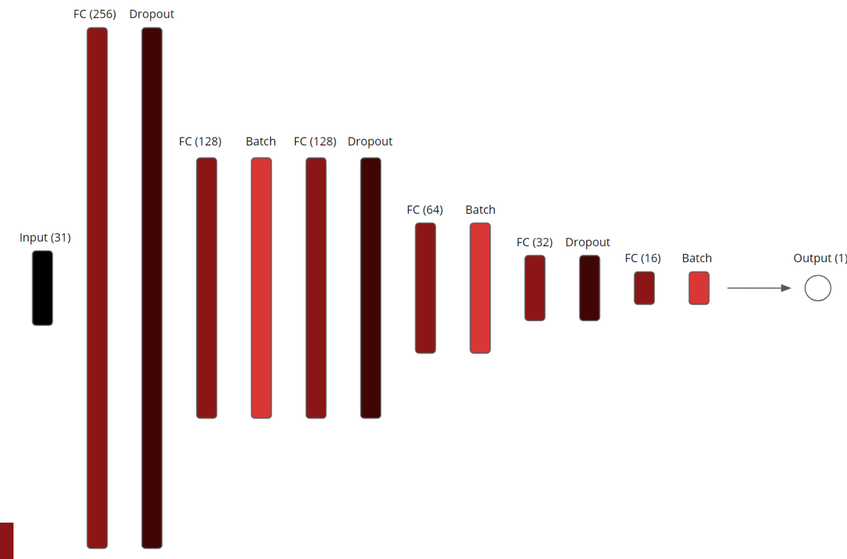
One area for improvement here is to investigate how the model adapts to variable passwords. Currently, the model has only been tested on the password `.tie5Roanl`. There are open questions about how the model might adapt to passwords of different lengths, or passwords that contain more character repetitions, etc. If we had access to a more sophisticated dataset, we would be able to provide more insight into what might need to be tweaked, but unfortunately no such dataset is available.

Moreover, in a practical implementation of this software, it would be infeasible to ask the user to enter their password 400 times. As such, an important and interesting question to explore is to ascertain how many times a user needs to enter the password before the results are acceptable, and moreover, to establish if we could continue to refine our estimates as user typing may continue to evolve.

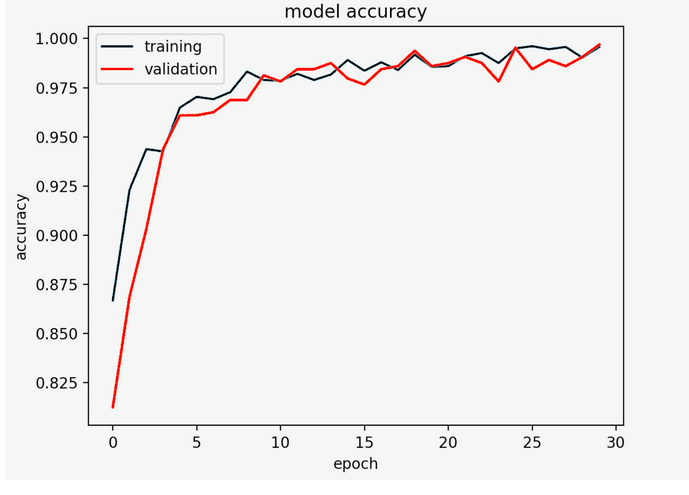
References

[1] Kevin S. Killourhy and Roy A. Maxion, "Comparing Anomaly Detectors for Keystroke Dynamics," in Proceedings of the 39th Annual International Conference on Dependable Systems and Networks (DSN-2009), pages 125-134, Estoril, Lisbon, Portugal, June 29-July 2, 2009. IEEE Computer Society Press, Los Alamitos, California, 2009.
[2] Yunbin Deng and Yu Zhong, "Keystroke Dynamics User Authentication Based on Gaussian Mixture Model and Deep Belief Nets," ISRN Signal Processing, vol. 2013, Article ID 565183, 7 pages, 2013. https://doi.org/10.1155/2013/565183.
[3] R. Moskovitch, C. Feher, A. Messerman, N. Kirschnick, T. Mustafi, "Identity Theft, Computers and Behavioral Biometrics," 2009. http://www.ise.bgu.ac.il/faculty/liorr/idth.pdf.
[4 Sungzoon Cho and Chigeun Han, "Web based Keystroke Dynamics Identity Verification using Neural Network," 2000. https://citeseerx.ist.psu.edu/viewdoc/.
[5] Daw-Tung Lin, "Computer-access authentication with neural network based keystroke identity verification," 1997.
[6] Will Koehrsen, "Beyond accuracy precision and recall," 2018. https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c.

Models



After iteratively tuning model hyperparameters, we arrived at the above architecture. This deep neural network model is comprised of 68,914 trainable parameters and 416 non-trainable parameters. To minimize model variance, we utilized a combination of regularization techniques. We found that by alternating batch normalization and dropout layers (with probability of dropout set to 0.01), the model reached a near-zero variance without increasing bias or hurting accuracy, recall, and precision.



In training the model, we used the standard Adam Optimization algorithm on mini-batches of size 48, with a learning rate of 0.001. 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.