A Lightweight Approach to Continuous Authentication through Keystroke Dynamics Using Neural Networks

Hiwot Kibebew
hiwluvsjesus@gmail.com

Taye Abdulkadir
HiLCoE School of Computer Science and Technology
taye.abdulkadir@hilcoeschool.com

Abstract

Integrating continuous authentication into computers through keystroke dynamics using machine learning/artificial intelligence demands real-time monitoring of signals from the keyboard in order to verify if the active user is legitimate. Since such process needs to run continuously, it must use minimal memory and computing power as a functional requirement. However, prior studies depend on complex high-dimensional feature engineering strategies for key-names, overlapping time-based features and heavier deep learning architectures.

To bridge the gap, this paper proposes lightweight approaches to feature engineering and shallower architecture for continuous authentication using keystroke dynamics. A shallow architecture of hybrid 1-dimensional Convolutional Neural Networks (1D-CNN) with Long-Short Term Memory (LSTM) is implemented. The model is trained and validated using Buffalo keystroke dynamics free-text dataset achieving a low Equal Error Rate (EER). The approach provides lightweight treatment for the problem through denser vector representation of keystrokes and deeper data understanding. The main contribution of the paper is the design of compact representation of keystroke data and a shallower architecture that works on shorter keystroke sequence length, hence with reduced computational cost.

Keywords: Machine Learning, Biometrics, Continuous authentication, Digraphs, Keystroke dynamics, Linear dependence, Neural Network, Trigraphs

1. Introduction

Authentication is the systematic process of approving a claimant's identity against a predefined distinctive identity profile established by a system to ensure legitimacy. The broadly used user authentication mechanisms or credentials depend on knowledge, possession, and biometrics to verify identity. The point of entry (PoE) nature of these traditional authentication mechanisms is leaving devices vulnerable to unauthorized access after a successful initial authentication.

The increasingly adopted form of authentication, multi-factor authentication (MFA) requires multiple factors for authorized access strengthening security with extra layers. MFA diminishes identity-fraud by forcing impostors to spoof multiple factors of authentication. Nonetheless, it adds to the complexity and time needed for authentication, often resulting in user frustration and consequently lower adoption rates [1].

As security expert Bruce Schneier postulated, '*amateurs tend to attack machines, while professionals target people*' [2]. Complex procedures can frustrate legitimate users, possibly leading them to adopt insecure workarounds. Security mechanisms are vital to defend users' privacy data, not to create excessive cognitive load. After all, security exists to protect users, not to burden them. Therefore, it's better to evaluate mechanisms that encourage user acceptance and minimize friction in the authentication process [1]. As a result, debate continues about the best strategies for the handling of security and usability as two sides of a coin.

Recent developments in security have heightened the need for continuous authentication (CA). CA refers to the ongoing process of verifying a user's identity throughout his/her session, instead of relying on a single login event. Several attempts have been made to strengthen CA by incorporating physiological, behavioural, and context-oriented characteristics so that systems remain in the hands of approved users [1,3].

Keystroke dynamics (KD) is a behavioural biometric authentication credential that analyses how a user types on a keyboard. Keystroke dynamics is "not what you type, but how you type" [4]. That is why keystroke dynamics is considered as a form of behavioural biometrics. KD involves analysing the patterns and rhythms of an individual's typing habit. As early as 1897, Bryan and Harter [5] documented that telegraph dispatchers could identify operators by their distinctive way of typing simply by listening. This illustrates the fact that the concept originates from human intelligence, where latent learning (an implicit learning without conscious effort) enabled people to continuously authenticate each other using their unique way of typing.

A recent meta-survey [6], benchmarking prior surveys, identified the advantages of KD as threefold: (1) cost-effectiveness, given that no additional hardware is required; (2) non-intrusiveness, ensuring minimal disruption to users; and (3) the potential for seamless remote deployment.

Even if KD is emerging as a promising modality for CA, the extraction of robust features from KD data remains a challenge. KD models need to be optimized for real-time performance while considering the computational limitations of Personal Computers (PCs) [7,8]. Existing KD based CA approaches often rely on heavily feature engineered features and computationally extensive models. This limits their suitability for real-time use. Additionally, clear and efficient method for representing physical keyboard input as a compact feature is not designed yet. This creates the need for lightweight and accurate approach that can operate under resource constrained PCs. The following research questions (RQ) are formulated to address the above stated problems.

- RQ1: How effective are different feature extraction techniques from keystroke data in capturing user-specific typing patterns for CA?
- RQ2: What deep learning architectures are best suited for CA using keystroke dynamics with minimal training time requirements?
- RQ3: How can these models be optimized to achieve real-time performance while maintaining high accuracy within the computational limitations of PCs?

2. Related Works

In the study by Xiaofeng et. al [9], the researchers designed a CNN (Convolutional Neural Network) followed by RNN (Recurrent Neural Network) architecture for CA using KD. In this research, the sequential data of the Buffalo KD data [10] was converted into a vectorized format that combines key-id (specific value representing key), hold time, press-release and press-press times between keys and this vectorized form is used to represent a fixed keystroke sequence. The study reported an "Equal Error Rate (EER)" of 3.04% using 100 keystroke sequences.

Yang et al., 2021 [11] proposed a keystroke model using learned-embedding for converting key-names into vectors, Bi-LSTM (Bi-directional Long-short Term Memory), type of RNN, and attention mechanism to learn keystroke features and analyse users' typing behaviour. They integrated key-code, dwell-time (D-time) and flight-time features into what they call 3-dimensional keystroke action. Then embedding technique is used to convert these three features into a suitable format for the Bi-LSTM algorithm. They obtained 8.28% EER using 30 keystrokes and 2.78% when using 190 keystrokes on the Clarkson II dataset [12]. Their feature engineering mechanism, which is learned embedding is data dependent and the additional layer of attention mechanism operates in quadratic computational complexity.

Li et.al [7] proposed two architectures; 1dimensional CNN with RNN model and 2-dimensional CNN (using five-channel, image-like matrix). The best accuracy obtained in the methods are 98.50 and 98.56 respectively on a keystroke sequence length of 100. The major drawback of these approaches however is that the keystroke analysis considers only 42-keys. Additionally, the image like matrix loses chronological information of keystroke data.

Although deep learning has shown promising results for KD-based CA, several fundamental limitations remain insufficiently addressed. First, existing approaches rely on extensive and high-dimensional feature engineering, which increases computational complexity. Despite the practical importance of efficiency, the design of lightweight models has received comparatively limited attention in the literature. Second, there is no well-defined mechanism for representing fixed keyboard inputs as compact and static vector forms. Current studies typically employ redundant timing features, which increase model complexity. Third, less focus is given on efficient feature representation, particularly for key identity. Existing learned-embedding approaches remain highly dependent on dataset size and characteristics, thereby limiting their generalizability. The combined impact of redundant feature design, lack of compact representation strategies, and insufficient focus on computational efficiency implies a methodological gap in current research. Therefore, this study addresses these limitations by proposing a lightweight neural network architecture that addresses feature redundancy, introduces a compact representation of keystroke inputs, and maintains competitive authentication performance under realistic computational constraints.

3. The Research Method and Dataset

The research method used is the DSRM (Design Science Research Methodology) pioneered by Peffers et al. [13]. The focus of the outcome on adaptability aligns with the "O/P-2" rule from a DSR selection guide in [14]. To embrace machine learning (ML), the study integrates CRoss-Industry Standard Process for Machine Learning with Quality Assurance (CRISP-ML (Q)) process model [15]. CRISP-ML(Q)'s business and data understanding phase aligns with DSRM's problem definition focus. In this merged phase, the limitations of the current authentication systems and suitability of KD data for CA is assessed. Then clear objectives are defined using business and data understanding. The subsequent phases mirror CRISP-ML(Q)'s approach, encompassing data preparation, modelling, evaluation, deployment, monitoring and maintenance. Finally, a DSRM-inspired communication phase ensured to effectively present the major findings. The quality assurance process in each phase of CRISP-ML(Q) promotes identification and mitigation of risks in every step of the research process.

The Buffalo Keystroke Dynamics dataset [10] was sampled to extract the portion relevant to the study's objective. The data sampled was collected from 75 baseline users who typed free text and used the same physical keyboard across all sessions. Each user's sessions are stored separately. The dataset gives key-names together with timestamps (CPU ticks). The two kinds of features that can be extracted from such data are:

1. Key-names
2. Timing features

Feature Engineering of Key-Names

Figure 1 illustrates the keyboard used in the data collection of the research data mapped into a 2-dimentional vector with annotated scatter plot. The thesis proposes three distinct feature engineering options to represent key-names. Option I is a 21-dimensional vector by concatenating functional groupings with spatial metrics, including Euclidean distances (using equation 1) and angles relative to the seven most frequent keys in the dataset (using equation 2). Option II condenses this result into a 6-dimensional vector using Principal Component Analysis (PCA), retaining 94.3% of the original variance. Finally, Option III focuses exclusively on spatial relationships, applying PCA to a matrix of distances between all 104 standard keys to produce a 3-dimensional vector preserving 96.22% of the variance and creates a unique 3D projection where every key remains distinct without overlap. While Option I provides the most granular raw data, Option III offers the spatial representation, mapping the physical keyboard layout into a three-dimensional space (Figure 2).

$$\text{distance_from_K}_1=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \tag{1}$$

$$\text{angle_from_K}_1=\operatorname{atan}\big((x_2-x_1),(y_2-y_1)\big) \tag{2}$$

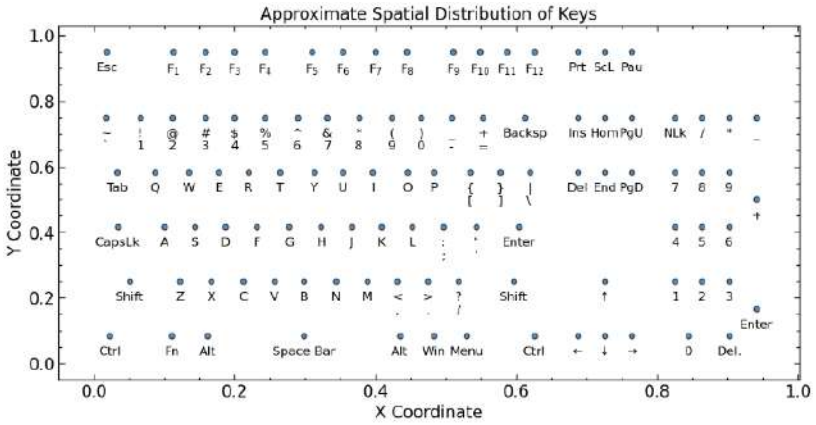


Figure 1: Spatial representation of keys on two dimensions

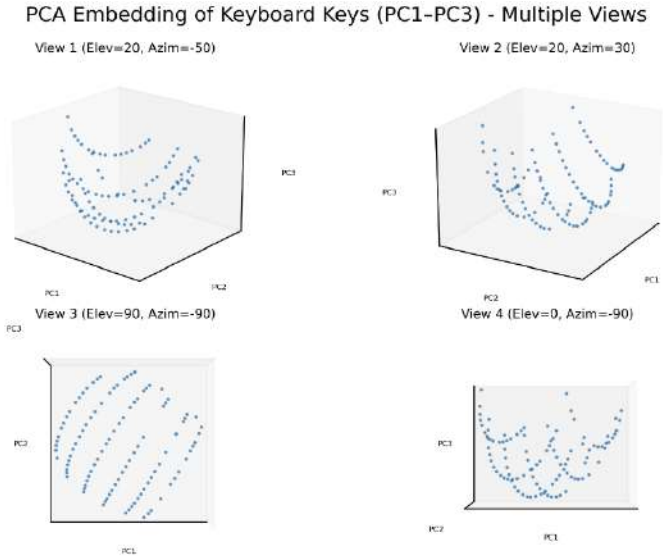


Figure 2: Three-dimensional visualization of keys on a keyboard

Feature Engineering of Timing Features

Six timing features are generated from consecutive press-release events of two keys [6,7]. These features are illustrated in Figure 3. But simple math shows there is linear dependence between the timing features. The information that is contained in the six timing features can be summarized in three features, two D-time features and one flight time feature, down-down-time (DD-time) without linear dependence in equations (3), (4) and (5).

The simple proof is available on the thesis [16].

$$DU = D_U + DD \tag{3}$$

$$UD = DD - D_E \tag{4}$$

$$UU = UD + D_U = (DD - D_E) + D_U \tag{5}$$

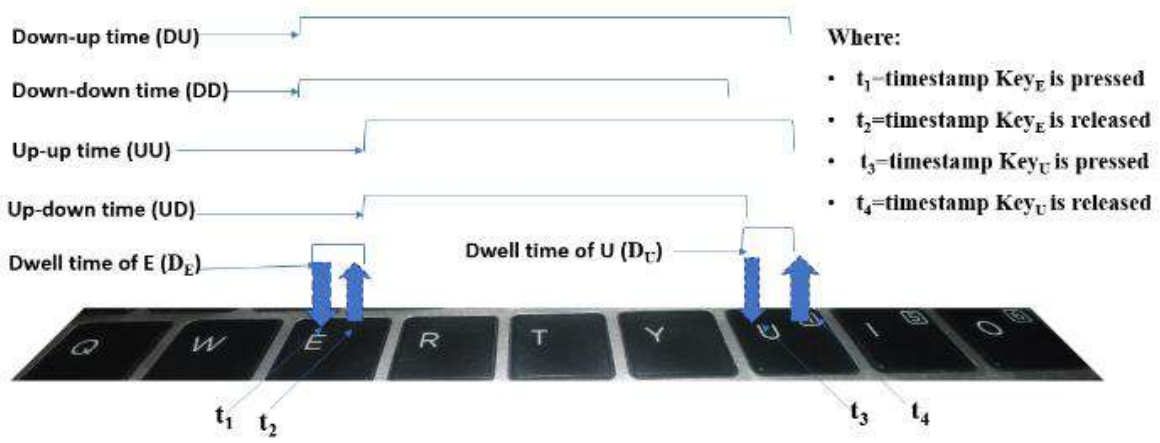


Figure 3: Six time-based features generated from key-press and release events of two keys.

Data Transformation

Figure 4 illustrates the steps that are followed to convert the raw-data into 2-dimensional tabular data. The raw data is first converted to a vector as [K1, D1, DD12, K2, D2, DD23, …., Kn, Dn,0], where Kn represents key-names mapped with Microsoft virtual key-codes and Dn and DDn represent time durations in seconds.

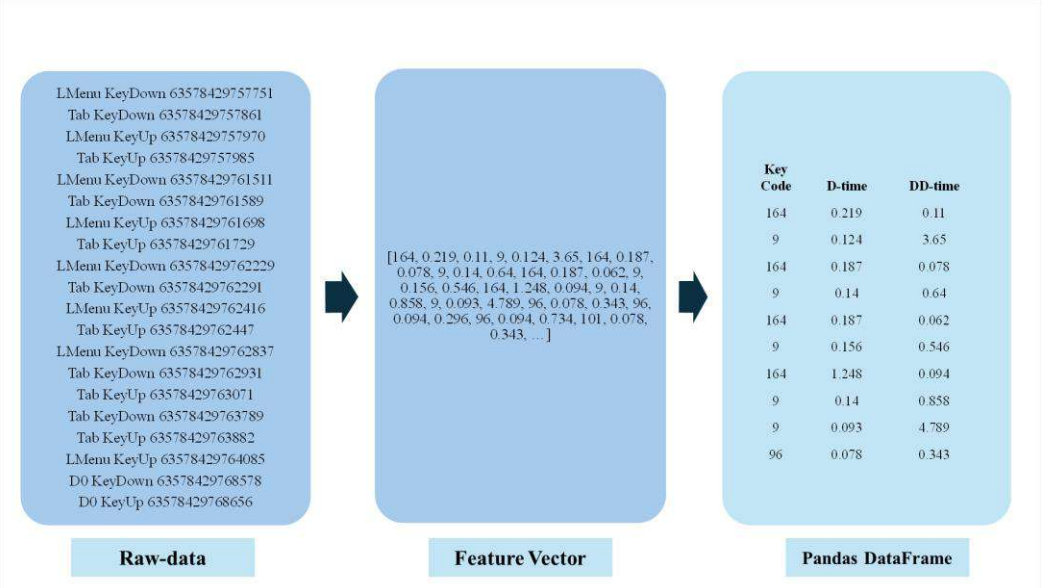


Figure 4: Diagram of data transformation from raw to 2-dimensional data

Data Visualization

Data visualization using histograms revealed that both timing features exhibit right skewness even after removal of extreme values. Nearly 98% of values of D-time fall between 0.01- and 0.3-seconds. Looking closely at the long-tail region, it is found that the main contributors were functional keys such as Left-Shift, Backspace, Left-Control and Right-Shift, which naturally tend to be held longer. Figure 5 advocates the domain of acceptable ranges for these keys ought to be different.

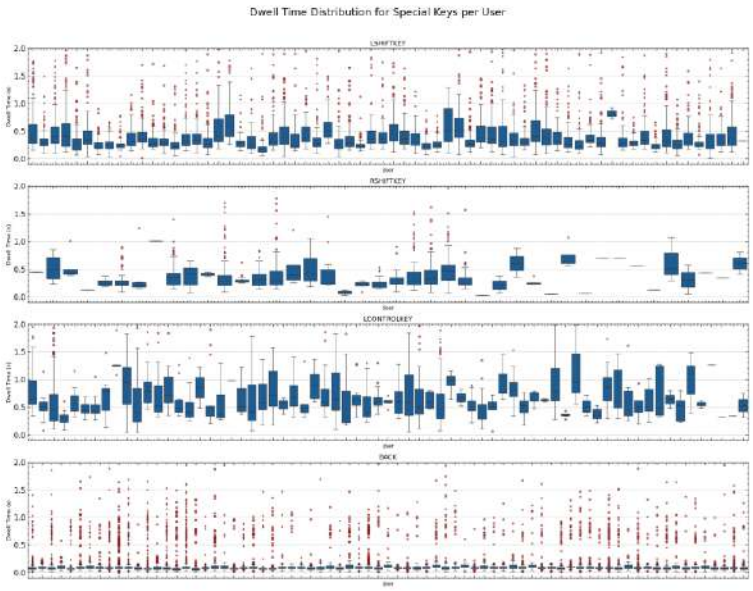


Figure 5: Per-user box plot representation of D-time for frequent special keys across all sessions

The other timing feature, DD-time exhibits a heavier right skew and greater dispersion than D-time, with higher variance across users. Figure 6 illustrates the distribution of the DD-time without merging per session user data that is clipped below 5 seconds using scatter plots. Each sub-section in the figure corresponds to one session. Across all three sessions, the plots reveal a dense cluster of points near zero, indicating that most DD-time intervals are very short. As the DD-time increases beyond 2 seconds, the dots become more dispersed, indicating that longer intervals between key presses are relatively rare. The majority of typing behaviour involves rapid successive keystrokes, while pauses exceeding 2 seconds occur infrequently.

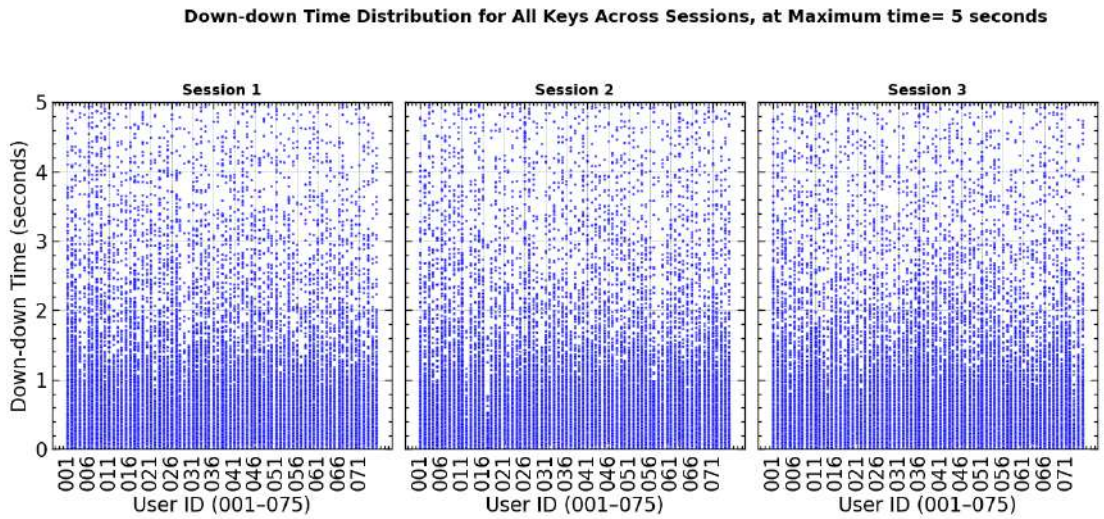


Figure 6: Scatter plot showing distribution of DD-time

Outlier Handling

Initial experiments using machine learning-based multivariate outlier detection mechanisms resulted the timing features within known acceptable ranges being identified as outliers. This proves that there is no significant correlation between the timing features.

To address the skewness without resorting to artificial data augmentation, two strategies were adopted in unification with domain knowledge. First, transformation alternatives for time-based features were applied after (logarithmic transformation and denoising as low pass filter using Fast Fourier Transformation). Second, long DD-time values were explicitly modelled as an inherent characteristic of a user's typing behaviour considering them as pauses. Specifically, a relatively long inactivity period (2-5 seconds) was interpreted as a natural pause. Then pause was represented as a binary value, which is 1 if DD-time ranges between 2 and 5 seconds or zero otherwise.

Data Segmentation and Train-Test Split

After data facts are understood and organized, the study implemented a structured train-validation-test split (70/10/20). Users with fewer than 1,000 sequences were excluded, resulting in a final total of 71 users. Data was segmented into fixed-length sequences using overlapping sliding windows to maximize sample volume.

Segmentation boundaries were defined by "*segment_here*" flags triggered by pauses exceeding 1.5s (D-time) or 5s (DD-time) (see Table 1). Genuine data was balanced with equal-sized impostor samples drawn from both internal (35 users) and external (35 users) pools to form the final model-ready tensors (Figure 7).

Table 1: Valid ranges and data segmentation

Feature	Range	Handling	Log Range
Key-Name	All	Encoding	Not applied
D-Time	(0.015,1.5]	Accepted	(0.0149, 0.9163]
	>1.5	New segment	--
DD-Time	(0.01,2]	Accept	(0.0149, 1.0986]
	(2,5]	Clip, 2sec	=1.0986
	>5	New segment	--

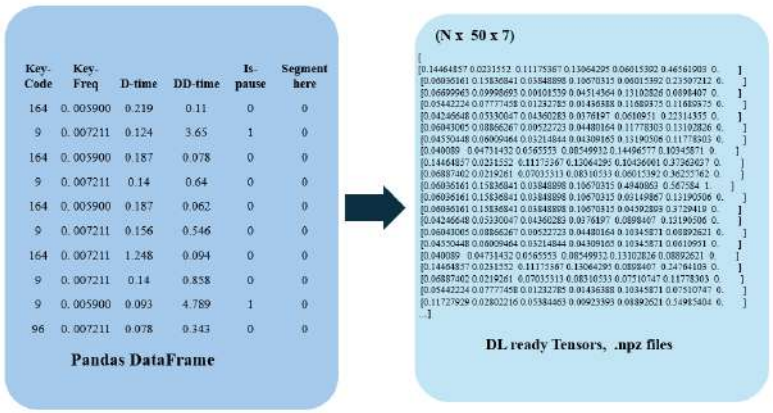


Figure 7: Transformation to model-ready sequence tensors

4. Results and Discussion

Experimental Setup

The experiments are done in a separate mini-Conda environment and the IDE (Integrated Development Environment) used is PyCharm with TensorFlow GPU (Graphics Processing Unit) environment. The experiments in the research were conducted using an NVIDIA GeForce RTX 3050 GPU with 4 GB of dedicated memory. The system was configured with driver version 580.97 and CUDA version 13.0. Experiments that required additional RAM were conducted using Tesla T4 (16 GB GPU) on Google Colab.

From a total of 71 users' data, the data of 30 users is experimented on to decide on promising architectures and encoding options as pilot experimentation. The preliminary experiments recognised that 3D-key-encodings consistently outperform other encoding alternatives in three chosen architectures with total of 12 variants per architecture. Multi-layer Perceptron (MLP) based architectures were worst performers as they scatter the data before extracting relationship between features, while CNN and TCN (Temporal Convolutional Network) architectures achieved promising results.

Table 2: Feature variants, intuition for having the model variants and definitions

Feature variants	Intuition	Definition
Raw	To keep it as a baseline feature representation	3D key-embedding, 1D key-frequency, D-time, DD-time and pause flag
Log	Reduce skewness without increased dimensionality	D-time and DD-time are log transformed using ln(1+x) and replaced
Raw +Log	Let the model learn from joint representations	Both the log transformed feature, and the raw value are kept
Raw +FFT	Give additional "*denoised*" version with the raw features	Time-based features are FFT transformed and then inverse Fourier transform after retaining only 20 low frequency components
Raw +Log+ FFT	For maximum expressiveness of features	Combines both transformations with raw features

Overall, the results confirm that CNNs and TCNs are the better fit for keystroke dynamics because they learn the timing rhythm and sequence structure before flattening the features into a 1D vector. CNNs consistently gave a relatively better accuracy so the next experimentations focused on improving and extending the CNN architecture.

In order to find best combination of hyperparameters for a final model, a grid search of 108 hyperparameter combinations including number of filters for CNN and Dense layers, LSTM units and dropout units were conducted on a data of a single user. The grid searches gave us two top performing architectures, CNN with LSTM and pure CNN.

The key challenges faced in creating a robust model after deciding on top two architectures is however building a model that fits on all users test sets. This led to exploration of feature transformation options. Table 2 summarizes the feature variants objectively considered for the final set of experiments.

The training setup on the final experiments are summarized in Table 3. The evaluation across 71 users (710 models total) demonstrated that the CNN-LSTM model paired with logarithmic

transformed featrues achieved the best generalization, yielding an average Equal Error Rate (EER) of 6.79% on external impostors and 5.14% on internal users (Figure 8). The combining the feature extraction of 1D-CNNs with the temporal memory of LSTMs, supplemented by logarithmic feature scaling, provides the most robust framework for capturing individual typing rhythms as illustrated by winning configuration Figure 9.

The result from the Fast Fourier transformation low pass filtering is not explainable even if it revealed good accuracy on some user's data. The result is not consistent. Additionally, a timestep of 50 is very short length data to decompose a discrete data into its sinusoidal waves and bring it back to the time domain values after low-pass-filtering. Filters of the CNN and "denoised" version of data from log1p and Standardscaler give more sound results.

Table 3: Training setup of the models' final models

Category	**Parameter /Setting**	**Value /Behaviour**	**Note**
Training configuration for both architectures	Batch size	16	-
	Epochs	40	-
	Early stopping	Monitor: validation-loss,	Restore best weights patience=10
	ReduceLROnPlateau	Monitor: validation-loss, Patience: 4, Factor: 0.5	Reduce learning rate when validation loss stops improving
	Optimizer	Adam, lr=5e-4	-
	Loss	Binary cross-entropy	-
	Dropout	CNN layers 0.2, Dense 0.2/0.5	Reduce overfitting
Model Architecture CNN	Conv1D-layers	2 layers kernel=3	Padding='same' Activation =ReLU Filter Layer1= 128 Filter Layer2= 256
	Pooling	Global average pooling	-
	Dense layer	128 units	Activation =ReLU Regulization=dropout
	Output	Dense 1 unit,	Binary classification, sigmoid
Model Architecture CNN+LSTM	Conv1D-layers	2 layers, kernel =3	Padding='same' Activation =ReLU Filter Layer1= 64 Filter Layer2= 128
	LSTM layer	128 units	Dropout=0.2
	Dense layer	128 units	ReLU activation, Dropout=0.5
	Output	Dense 1 unit	Binary classification, sigmoid
Evaluation	Metrics	Accuracy and EER	Internal and external impostor data
	EER calculation	ROC FPR – FNR intersection	

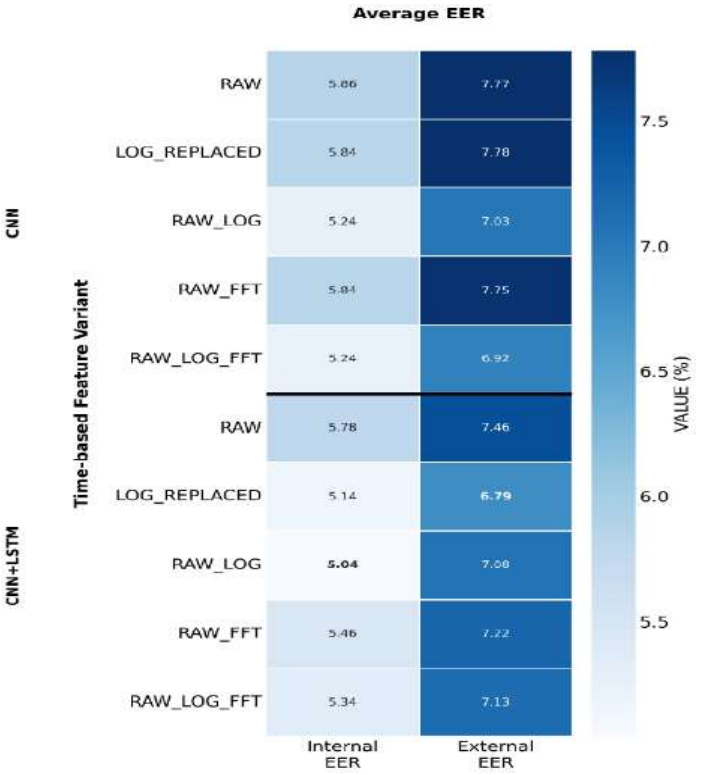


Figure 8: Performance comparison of CNN and CNN-LSTM

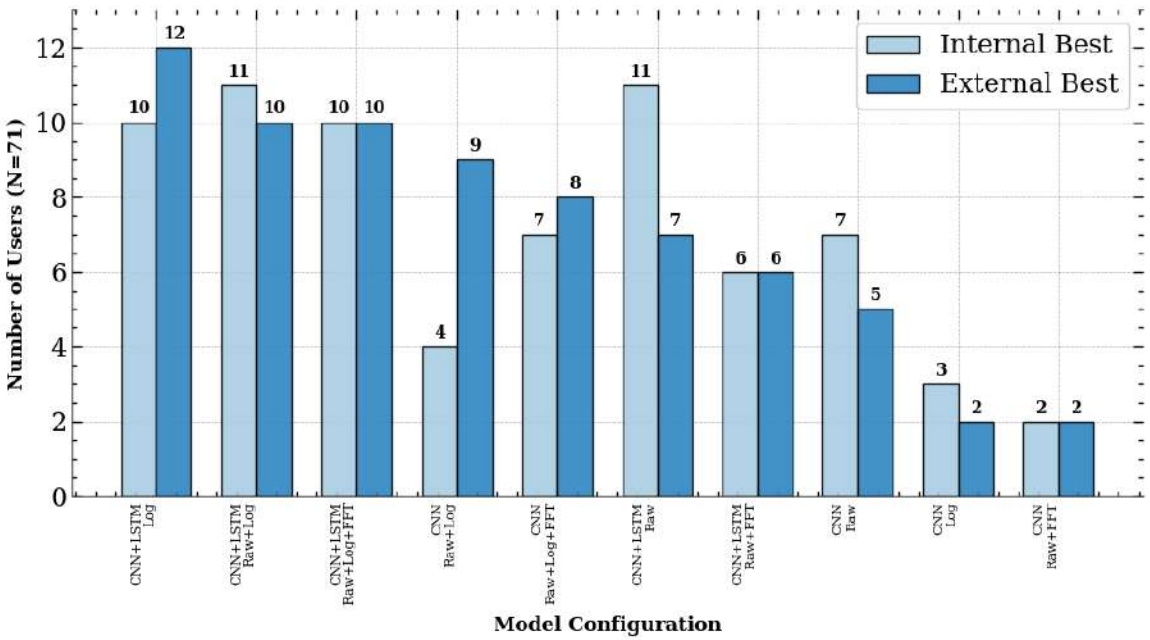


Figure 9: Winning configurations across all users

Model Explainability

Given a sequence of 50 keystrokes represented by 50 timesteps in a 2-dimensional array (50x7), three consecutive rows represent the information of a trigraph. The first 1D-CNN layer uses 2D-kernels of size 3 (3x7) that convolve over the time-axis, hence 1D with stride of 1 and same padding to give 50 filtered features that are 64 in dimension, (50*64). These are the first trigraph information filtered. Then, the (50*64) array is forwarded to another CNN layer with number of filters doubled after 20% of the neurons are zeroed by a dropout rate of 0.2 to overcome overfitting. This layer again using its kernel of size 3, (3*64) does the convolution on all the 50 timesteps learning "trigraphs of trigraphs". Since there are 128 filters, it results an array of size (50*128). Then additional LSTM layer is added. LSTMs have special kinds of RNNs that have gates to remember and forget information using their long and short-term memories. Then the output relies on seeing a dense, (50*128) sequence, "trigraph of trigraphs" to model consistency and changes over time of typing behavior. The dropout rate of the final fully connected layer for this hybrid architecture is 0.5 (result of grid search).

Figure 10 illustrates the flow chart for the proposed CA process using KD. It begins with capturing the user's keystrokes and extracting features; key-name-encoding, D-time and DD-time between keys. The input data is then segmented into a fixed keystroke sequence and a confidence score is calculated to assess the probability that the current user matches the enrolled profile. If the confidence score is acceptable, the system proceeds to the continuous authentication loop. If the confidence score is not acceptable then, the system calculates the ANIA (average number of impostor actions) to further check on application-based decision. Based on this evaluation, the system determines whether the associated risk is acceptable, if it is, the session continues, otherwise, system-defined security measures, such as session lock or re-authentication, are triggered to protect against unauthorized access.

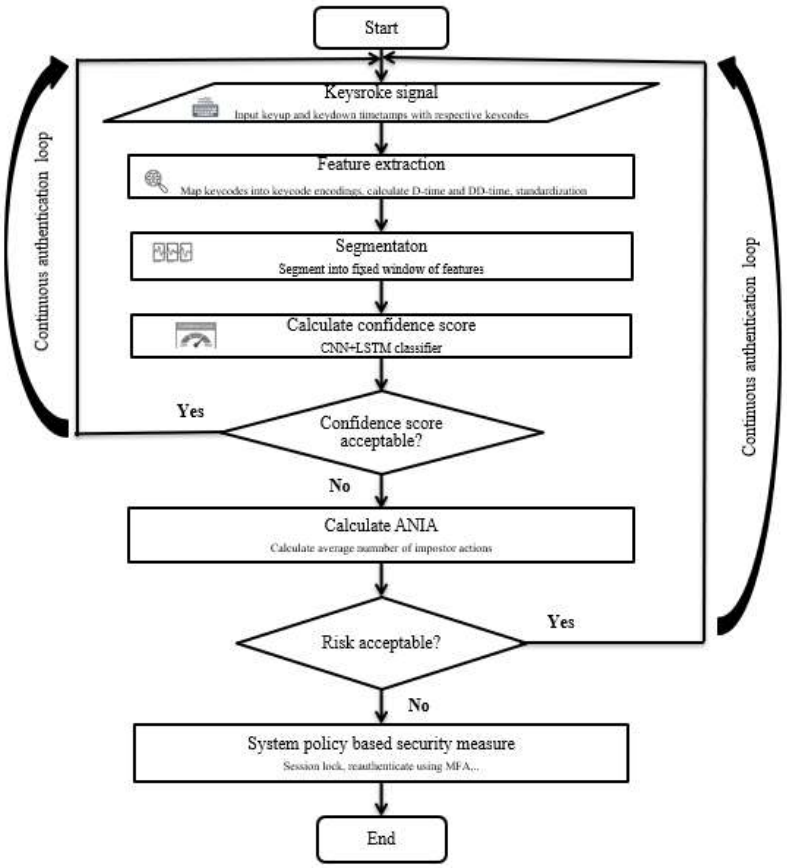


Figure 10: Flow chart of CA using KD

5. Conclusion

The study aimed to design and optimize an efficient CA model through KD. The research explored on possible ways of how keystroke dynamics features can be represented and modeled to achieve accurate and deployable model. In the study:

- Key-name feature is transformed and engineered as a dense vector
- Dense, space-efficient encoding of trigraphs reduces redundancy compared to sliding-window representations.
- Logarithmic transformation improves robustness by addressing skewness in timing distributions.
- Lightweight CNN+LSTM architecture can achieve competitive performance
- Feature redundancy exists in current time-based representations and should be addressed in future designs.

Overall, the results confirm that lightweight and well-designed feature representations can enable accurate, efficient, and practical CA for real-world systems.

Contribution

The research has results that contribute to exaptation and improvement quadrants of DSR. The key-encoding is derived from the concept of word embedding in natural language processing (NLP). The study provides new insights into dense and space-efficient encoding of the categorical key-name data. The study extends the knowledge of categorical encoding in DL architectures, confirming the fast generalization capability of DL models on dense data. Improvement is made on representation of timing features, avoiding linear dependence. Besides, this is the first study to explicitly demonstrate the existence of feature redundancy in time-based features.

A compact representation of trigraphs using a vector of size (7×3) is proposed. In contrast to prior research, that represents digraphs using vectors such as (K1, K2, D1, D2, UD, UU, DU, DD), with a minimum feature vector of 180-dimesion (2x90) after truncating infrequent keys. The proposed dense feature vector representation avoids redundancy in information and hence improves space efficiency.

The study treats pause as meaningful signal rather than mere noise or an outlier. The research finally demonstrates that robust continuous authentication does not require excessively deep or computationally expensive models.

References

[1] M. Smith-Creasey, Continuous Biometric Authentication Systems, S. i. C. Science, Ed., Cham, Switzerland: Springer International Publishing, 2024.

[2] B. Schneier, "Inside risks: semantic network attacks," *Commun. ACM,* vol. 43, no. 12, p. 168, Dec. 2000.

[3] A. F. Baig and S. Eskeland, "Security, Privacy, and Usability in Continuous Authentication: A Survey," *Sensors,* vol. 21, no. 17, p. 5967, 2021.

[4] F. Monrose and A. Rubin, "Authentication via Keystroke Dynamics," in *Proceedings of the 4th ACM conference on Computer and communications security*, Zurich, Switzerland, 1997.

[5] W. L. Bryan and N. Harter, "Studies in the physiology and psychology of the telegraphic language," *Psychol. Rev.*, vol. 4, no. 1, pp. 27-53, Jan. 1897.

[6] R. Shadman, A. A. Wahab, M. Manno, M. Lukaszewski, D. Hou, and F. Hussain, "Keystroke Dynamics: Concepts, Techniques, and Applications," *ACM Computing Surveys*, vol. 57, no. 11, pp. 1-35, June 2025.

[7] J. Li, H. Chang and M. Stamp, "Free-Text Keystroke Dynamics for User Authentication," in Advances in Information Security, Berlin, Germany, Springer, 2022, pp. 357-380.

[8] C. Dwivedi, D. Kalra, D. Naidu, and S. Aggarwal, "Keystroke dynamics based biometric authentication: A hybrid classifier approach," in IEEE Symposium Series on Computational Intelligence, Bengaluru, India, Nov. 2018.

[9] L. Xiaofeng, Z. Shengfei, and Y. Shengwei, "Continuous authentication by free-text keystroke based on CNN plus RNN," Procedia Comput. Sci., vol. 147, p. 314–318, Jan. 2019.

[10] Y. L. Sun, H. Ceker and S. Upadhyaya, "Shared Keystroke Dataset for Continuous Authentication," 2016 IEEE International Workshop on Information Forensics and Security (WIFS), pp. 1-6, Dec. 2016.

[11] L. Yang, C. Li, R. You, B. Tu, and L. Li, "TKCA: a timely keystroke-based continuous user authentication with short keystroke sequence in uncontrolled settings," Cybersecurity, vol. 4, no. 1, p. 13, May 2021.

[12] C. Murphy, J. Huang, D. Hou, and S. Schuckers, "Shared dataset on natural human-computer interaction to support continuous authentication research," in *2017 IEEE Int. Jt. Conf. Biom. IJCB*, Oct. 2017.

[13] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research," Journal of Management Information Systems, vol. 24, pp. 45-77, Dec. 2007.

[14] J. R. Venable, J. Pries-Heje, and R. Baskerville, "Choosing a Design Science Research Methodology: 28th Australasian Conference on Information Systems," in ACIS2017 Conf. Proceeding, 2017.

[15] S. Studer et. al, "Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology," Mach. Learn. Knowl. Extr., vol. 3, no. 2, p. 392–413, Apr. 2021.

[16] H. Kibebew, "A Lightweight Approach to Continuous Authentication through Keystroke Dynamics Using Neural Networks, " M.S. thesis, School of Computer Science and Technology, HiLCoE, Addis Ababa, Ethiopia, December 2025.