Journal of Neonatal Surgery
ISSN(Online): 2226-0439
Vol. 14, Issue 4 (2025)
https://www.jneonatalsurg.com

OPEN ACCESS

Deep Learning-Based Authentication Using Keystroke-Dynamics

Namisha Bhasin[1*], Sanjay Kumar Sharma[1], Rajesh Mishra[1]

[1]Gautam Buddha University, Greater Noida, India

[*]**Corresponding author:**

Namisha Bhasin

Cite this paper as: Namisha Bhasin, Sanjay Kumar Sharma, Rajesh Mishra, (2025) Deep Learning-Based Authentication Using Keystroke-Dynamics. *Journal of Neonatal Surgery*, 14 (4), 524-535.

ABSTRACT

Keystroke dynamics where a user is authenticated based on his/her typing patterns. It is considered as best solution to authenticate a user as this problem is solved by considering behavioural characteristics which is very difficult to copy. In this research paper we solved the problem of static keystroke dynamics by deep learning approach.in this paper we use the concept of quantile transformation which reduces the impact of outliers. For pattern reorganization, various optimization algorithms are used. For global pattern recognition various Metaheuristics algorithms and for local pattern ADAM optimization algorithm is used. The best solution is achieved by the Firefly Optimization Algorithm (a nature-inspired, swarm-based metaheuristic) which excels at global pattern reorganization by using bioluminescent-based attraction. Here, less-optimal solutions are drawn toward better ones through intensity-based movements, with attractiveness decreasing over distance. This mechanism enables efficient exploration of the search space and helps locate the best solution.

Keywords: *CONV1D, whale optimization, firefly, enhanced wolf, gannet optimization*

1. INTRODUCTION

A user accesses the computer system by entering the login credentials. Anyone having that information can access the system who can be a genuine /imposter. But if access of a computer system is based on unique feature of a user. In market various solutions exist like one-time password, ATM card and face reorganization. But these solutions are not full proof as for OTP we need to carry a mobile device and if that is not there a genuine user cannot access the system which leads to denial of service. Similarly for ATM card also there is a requirement to carry the card. In face reorganization system any change in the face like a smiling line or closed eye does not allow to access the system[1].

The solution to above problems is keystroke dynamics where a user authentication is proved by a user's typing rhythm. For this just a keyboard is required which itself is an integral part of the computer system. It means no extra device is required to carry. A user is authenticated after giving the login credentials is known as static keystroke dynamics which is a type of certificate that typed password is by genuine user or not. To achieve this a user has to type 20 to 25 letters and that typed word pattern should match with the stored pattern values. For many logins credential strong password is required and necessary requirement where a password should be a combination of small, capital letter, numeric and special character. It means they are like which do not carry any meaning. This concept has two advantages 1) these words don't carry any meaning and as this word is type most of the time by the genuine user who after sometime has remember it so type that password more confidentially and with a definite type of pattern, 2) the imposter cannot type the same word with that confidence hence tying pattern of a his/her will be different from the required pattern[2-3].

Deep learning now dominates keystroke dynamics research leveraging rich representations, attention mechanisms, and multi-modal data to yield state-of-the-art authentication systems. Deep Learning for Keystroke Dynamics harnesses Convolutional Neural Networks (CNNs) to extract spatial-temporal typing features and Recurrent Neural Networks (RNNs) especially when LSTM/GRU variants to model sequential keystroke patterns, enabling accurate classification of normal vs. anomalous typing behaviours[4].

While highly effective, these deep learning-based systems require extensive labelled datasets and significant computational resources, and they remain vulnerable to adversarial attacks, maliciously crafted inputs that can cause misclassification with minimal perturbations. MLPs struggle with non-linearly separable data and rely on hyperplanes for decision boundaries, making them limited in expressiveness; they are prone to overfitting on small or noisy datasets unless carefully regularized, and demand meticulous tuning of layers, neurons, activation functions, and optimizers with no universally accepted best practice available[5-6].

istory of women with coagulation disorder and all women on anticoagulant therapy were the exclusion criteria for this study. All the antenatal women were willing to participate and signed the informed consent document was enrolled in the study. Demographic characteristics included age, booking status, area of residence, socioeconomic status, and gestational age at presentation were noted. Clinical characteristics including presenting complaints, fetal heart sounds (normal, reduced, and absent), and obstetric factors were

Autoencoders are trained exclusively on 'normal' typing behaviour, so genuine-but-new patterns like those resulting from injury or fatigue can produce high reconstruction errors and trigger false alarms. They also require large, clean datasets of normal behaviour; any shortage or imbalance can distort detection thresholds. Moreover, setting the cutoff between 'normal' and 'anomalous' based on reconstruction error typically relies on heuristic tuning rather than a principled method. Typing behaviour varies significantly across keyboards and usage contexts, so Random Forest models trained on one type such as mechanical keyboards often struggle to generalize to others. While RF excels in structured fixed-text scenarios (e.g., the Buffalo dataset), it underperforms in free-text environments due to higher variability. Additionally, RF relies heavily on manual feature extraction such as digraph and trigraph timing matrices and its performance heavily depends on the quality of these crafted features[7-8].

LSTMs, with their multiple gates and memory cells face heavy computational demands, remain vulnerable to exploding gradients and overfitting, and struggle to maintain performance on extremely long or highly variable keystroke sequences. Statistical / Distance-based Methods are highly sensitive to tuning and threshold settings, raising thresholds to reduce false accepts often inflates false rejects, degrading user experience and vulnerable to typing drift over time, necessitating frequent retraining. Hidden Markov Model(HMM) based methods, while they can model temporal patterns, they are complex to configure and remain susceptible to template drift and text variability, undermining long-term stability[9-11].

This research aims to tackle key limitations of current keystroke-dynamics models by leveraging optimized deep-learning techniques to enhance robustness, adaptability, and security surpassing traditional approaches and strengthening KD-based authentication in today's dynamic digital landscape.

Contributions:

- This research solves the static keystroke dynamics problem using three optimization techniques 1) firefly, 2) whale 3) enhanced whale and 4) enhanced gannet optimization algorithm with conv1D.
- These optimization techniques are stochastic, nature-inspired metaheuristic algorithms designed to address complex problems. They use the concept of randomness to explore vast solution spaces and avoid getting trapped in local optima hence used for global optimization.

This research paper is explored as section 2 for background, section 3 dataset section 4 pre-processing, section 5 processing, section 6 results and section 7 conclusion.

2. BACKGROUND

The authors collected the data from 63 users for over 11 months where users can type the data by any machine. They used KNN for classification. They were able to solve the problem with 92.14% accuracy[12].

The authors collected dataset for 102 users considering two languages 1) English, and 2) Arabic named as Bilingual Keystroke Dynamics Dataset. They found that typing language was the primary driver and not spatial layout. They were able to solve KD problem with an average EERs of 0.486% when training for English and testing for Arabic was done. They achieve an EER of 0.475% when training was performed on Arabic dataset but testing was performed on English dataset[13].

The authors used Siamese neural networks and the fused CMU and Key Recs datasets. They use dataset of 51 users from CMU and 99 users from Key Recs datasets with different features. They converted the dataset to GAFMAT where Gabor filters is applied to highlight key timing patterns. The images were resized by bilinear interpolation and then classified by SNN algorithms with an accuracy of 89.5% and an EER of 10.5%. As can be seen model performance is not up to mark[14].

The author solved the problem of static KD using CMU dataset with CNN+GRU approach. The author was able to solve the KD problem with an accuracy of 99.31% and EER of 0.069[15].

The authors solved the static keystroke dynamics problem using elbow, principal component analysis, and LSTM techniques with a loss of 9.2%[16].

The authors solved the static KD problem by considering three datasets 1) CMU, 2) Key Recs and 3) GREYC-NISLAB. They fuse the dataset with interpolation-based fusion technique. They were able to solve the problem of KD using Siamese neural network with a triplet loss function and were able to achieve an EER of 0.13281[17].

The authors categorised the users in two age groups of above and below 18 with an accuracy of 80% using statistical method. To solve KD problem, they collected data from 116 users where 70 users were adults while 46 were below the age of 18[18].

The authors solved the problem of free-text keystroke dynamics problem with an EER from 0.009 to 0.127. They solved the KD problem using combination of convolution neural network and bidirectional long short term memory model. For this research paper they collected data from 10 users for 6 months[19].

The authors solved the problem of static KD using CMU dataset with quantum machine learning(QML) and hybrid algorithms . As performance of machine learning algorithms is suboptimal and this problem is solved by using QML. They were able to achieve an accuracy of 100% with hybrid SVC[20].

The authors solved the problem of static keystroke dynamics problem by converting data into 3-dimenssional image. For training they used convolution neural network and for testing distance metrics. They were able to achieve an EER of 9.17% to 22.95% with username and password[21].

3. DATASET

For this research paper CMU[22] static keystroke dataset is considered where fifty-one users participated in the experiment. The users have to type a word ".tie5Roanl" 400 times in 8 sessions and in each session, they have to type 50 times this word. The word carries no meaning. It represents one of the strongest category passwords as it is composed of small, capital, numeric and special character. Some users registered for data collection but did not participate so they were removed from the final list as user1 did not participate so subject list starts from user 2. There are 31 features which represents 1) hold time(H), 2) down-down(DD) and 3)up-down(UD) dataset. hold time is calculated based on the time consumed in releasing-pressing of a key whereas down-down time reflects the time consumed in pressing of two consecutive keys. Similarly, up-down tells time consumed in pressing and releasing of two consecutive keys.

4. PREPROCESSING

In preprocessing all steps are similar to [23] so that a comparison between the models can be performed. So, on the same course of action, the dataset considered in this research paper is CMU dataset. Here, users type the word".tie5Roanl" and it contains 400 rows and 31 features. In the next step, the dataset is synthetic, with a standard deviation ranging from 400 to over 2000 rows and 31 features for each user. The dataset is then converted to Gaussian distribution shape with the help of Quantile Transformation. For training and testing dataset is divided into 70:30 ratio.

5. PROCESSING

For processing following steps are considered:

1. Sliding windows and updating: to extract temporal patterns windows of overlapping fixed length is used where in fixed length 64 samples are considered and for overlap 20% of previous data with new samples is considered.
2. Window Filtering by Class Purity: if less than 90% of samples belongs to the same class then that window of samples is discarded.
3. Convolution neural network 1D(CONV1D)[24]: for this research paper a model with two convolution layers having 64 and 128 filters, max-pooling, global max-pooling and dense layer are used.
4. Optimization algorithm: for local ADAM[25] optimization(Adam is a gradient-based optimizer which with the help of Momentum and Root Mean Square Propagation adaptively tune learning rates for each parameter.). to compute the gradient $g_t = \nabla_\theta L(\theta_t)$, for learning rate of each parameter by the exponential moving average of recent squared gradients with decay rate(γ) is represented as: $E|g^2| = \gamma E|g^2|_{t-1} + (1-\gamma)\, g_t^2$, then parameters are updated as $\theta(t+1) = \theta(t) - \frac{\eta}{\sqrt{E[g2]t+\epsilon}}.g(t)$ where $\epsilon \approx 10^{-8}$ η and is base learning rate.
5. For global patterns different optimization algorithms are used as listed below as shown in Figure 1:

Figure 1: steps followed to solve the authentication by keystroke dynamics problem

Here is an overview of various swarm-based and nature-inspired metaheuristic optimization techniques, with emphasis on their unique mechanisms for global pattern reorganization:

a. **Gannet Optimization Algorithm**[26]:

The Gannet Optimization Algorithm (GOA) simulates gannets' foraging behaviours through two phases exploration using U/V-shaped dives to locate promising regions, and exploitation using sudden turns and random walks to refine solutions within them. It alternates between exploration and exploitation using four main dive and motion strategies: U-shaped dive, V-shaped dive, sudden rotation, and random wandering. They explore the search space through U/V-shaped dives. But when gannets detect nearby prey, it focused to sudden rotation and random walk. The optimizer simulates foraging behaviour of gannets where exploration and exploitation are chosen randomly, update of position by appropriate dive is performed. It has Population-Based Search where each vector in the population represents model weights $\vec{v_i} \in R^n$ and fitness refers to how well a candidate weight vector generalizes to unseen data. That's why validation loss is used here and not training loss which helps to assess each vector during the fitness evaluation. For search following steps are followed:

1) Exploration-involves broadly searching the solution space to discover new potential regions where optimal solution might exist which helps in avoiding getting stuck in local optimum solution. For optimum solution 30% of GOA is considered for exploration. To add Gaussian noise scaled by iteration:$\vec{v_i} = \vec{v_i} + N\left(0{,}0.05, \left(1 - \frac{g}{G}\right)\right)$, where g is the current generation index, beginning at 0 or 1 and G is the total number of generations defined for EGOA.

2) Exploitation which is considered 60% of the GOA algorithm for optimum solution, move towards the current best solution with noise:$\vec{v_i} = \vec{v_i} + 0.7(\vec{v}\,best - \vec{v_i}) + N(0{,}0.01)$ where v_i is the *current candidate* solution where a vector representing one individual's neural network weights in the population and v_{best} represents the best weight vector found so far.

3) Random Jump is considered 10% of GOA for optimum solution it is reset to a random solution:$\vec{v_i} \sim U(l_0, hi)$ where objective is to minimize $\mathcal{L}_{val} = CROSSENTROPY(y_{val}, \hat{y}_{val})$.

For CONV1D where 1) filters learn temporal patterns is represented as $output_j[t] = \sum_{k=1}^{k} x_{t+k-1} \cdot w_{jk}$, 2) MaxPooling1D is used which helps in down sampling by taking max value is represented as window$pooled[t] = max(x[t], x[t+1])$, 3) Dense Layers which is fully connected layers with ReLU activation presented as: $a^{(l)} = max\left(0, w_a^{(l)} a^{(l-1)} + b^{(l)}\right)$. and

4) Softmax Layer which converts logits to class probabilities represented as $\hat{y}_i = \frac{e^{zi}}{\sum_f e^z j}$.

b. **Firefly optimization Algorithm**[27]:

It is a type of Metaheuristics algorithm whose base is bioluminescence which is a chemically produced light in their abdomen,

and it is visible in the visible spectrum which is used to communicate. It works on the following rules: 1) Unisexual interaction where all fireflies are considered unisex, so any firefly can be attracted to any other regardless of gender; 2) Brightness-driven attraction where a firefly's attractiveness is proportional to its brightness considered as fitness, less bright fireflies move toward brighter ones; attractiveness decays with distance; 3) Random movement when equal where if brightnesses are equal, they move randomly to explore the space.

Here, impact of different factors is described as: 1) brightness is defined as I(x) and objective function as *f(x)* then for optimum result $I_i=f(x_i)$; 2)distance calculated as Euclidean distance as attractiveness and brightness fall off with distance $r_{ij}=|| x_i-x_j ||$; 3) attractiveness based on light principles $\beta^{(r)} =\beta_0 \exp(-\gamma r^2)$ where β_0 brightness at zero distance, γ light absorption coefficient; 4) movement where firefly i is attracted to j represented as: $x_i^{t+1}=x_i^t+\beta_0 e^{-\gamma r2ij}(x_j^t-x_i^t)+ \alpha^t.\epsilon^t$ where middle term i pulls towards j; 5) convergence control where large γ means limited visibility, and γ=0 means global visibility. Low value of α_t helps converge from exploration to exploitation.

c. **Whale Optimization algorithm(WOA)**[28]:

It also comes under the category of Metaheuristic algorithms. In this algorithm, the concept of Humpback whales to trap schools of fish by swimming in 9-shaped spirals or circles is used. They update their position following either spiral search or random smaller moves. It follows the bubble-net feeding strategy of the humpback whale as follows: 1) Encircling prey or exploitation where whale move towards the best-known solution with probability less than 0.5 and |A|<1 is represented as:

$D=| C.X^* (t)-X(t) |,$

$X(t+1) = X^*(t)-A.D,$

where $A=2a_tr_i-a_t$, $C=2r_2$, r1, r2~ U (0,1), at decreases linearly from 2 to 0.;

2) Bubble-net spiral attack which is also known as exploitation where whales spiral towards prey with more than 0.5 probability is defined as:

$D'=| X^*(t)-X(t) | ,$

$X(t+1)=D'. e^{bl} \cos(2\Pi l)+X^*(t)$ where b is spiral shape constant and l~U(-1,1).;

and 3) Search for prey also known as exploration where whales move randomly to explore new solutions with probability less than 0.5 and |A|>=1 is represented as

$|X(t+1)|=X_{rand} - A|C.X_{rand} -X(t)|,$

X_{rand} is used to choose whale from population, it ensures global exploration when |A| is large.

For stochastic diversity random values r1,r2,p,l are used and by decreasing a_t more focus on exploitation than exploration is possible.

d. **Enhanced Wolf Optimization Algorithm(EWOA)**[29]:

It is categorized as a metaheuristic algorithm inspired by grey wolf hunting behaviour. Here, grey wolves in a pack surround prey leaders estimate the prey's location. Non leader wolves then encircle these leaders by adjusting their positions towards them. During optimization, wolves update their positions by considering the top three leaders in the pack also known as α, β, and δ where α is considered as best solution, β is considered as 2nd best solution, and δ is 3rd best solutions is known as social hierarchy. Other solutions achieved with the help of α, β, and δ are known as ω(omegas). Second step, is Exploitation Phase also known as Encircling Prey where a calculation for each wolf distance and encircling updates is calculated as follows:

$D_\alpha =|C_1 . X_\alpha -X_k|,$

$X_1=X_\alpha -A_1. D_\alpha,$

$D_\beta =|C_2. X_\beta -X_k|$

$X_2=X_\beta -A_2. D_\beta,,$

$D_\delta =|C_3 .X_\delta -X_k|,$

$X_3=X_\delta -A_3 .D_\delta,$

$$Xk(t+1) = \frac{X1 + X2 + X3}{3}$$

where A_i=2a.r-a and C_i=2.r and r~U(0,1) and to shrink the search scope over iterations value of a decreases from 2 to 0(controls convergence).

When |A|<1 also known as exploitation when wolves concentrate around leaders and opposite is called exploration.

6. RESULTS

The results are evaluated on various features as follows:

1. Accuracy where proportion of correctly predicted instances among all samples is calculated.
2. Precision depicts the ratio of correctly predicted positive observations to all predicted positives is considered.
3. Recall where the ratio of correctly predicted positive observations to all actual positives is considered.
4. F1-Score a very important point where the harmonic mean of precision and recall, balancing both metrics is calculated.
5. Equal Error Rate where the threshold at which the false positive rate (FPR) equals the false negative rate (FNR) on the ROC curve is checked.
6. Macro average where the metric independently for each class is calculated, and then an unweighted mean is calculated.
7. Weighted average which Computes metrics per class and then averages them using support-based weights.

Total test cases are 90 as class purity filtering (windowing) where windows with 90% of same labels are discarded. Processing of Conv1D model is presented in Figure 2 .

Layer (type)	Output Shape	Param #
input_layer_3 (InputLayer)	(None, 64, 7)	0
conv1d_6 (Conv1D)	(None, 64, 64)	1,408
max_pooling1d_3 (MaxPooling1D)	(None, 32, 64)	0
conv1d_7 (Conv1D)	(None, 32, 128)	24,704
global_max_pooling1d_3 (GlobalMaxPooling1D)	(None, 128)	0
dense_6 (Dense)	(None, 32)	4,128
dense_7 (Dense)	(None, 2)	66

Total params: 30,306 (118.38 KB)
Trainable params: 30,306 (118.38 KB)
Non-trainable params: 0 (0.00 B)

Figure 2: presentation of Conv1D model

A. The static keystroke dynamics problem when considering gannet optimization algorithm(GOA) an accuracy of 99% and EER of 0.000% is achieved as shown in Figure 3.

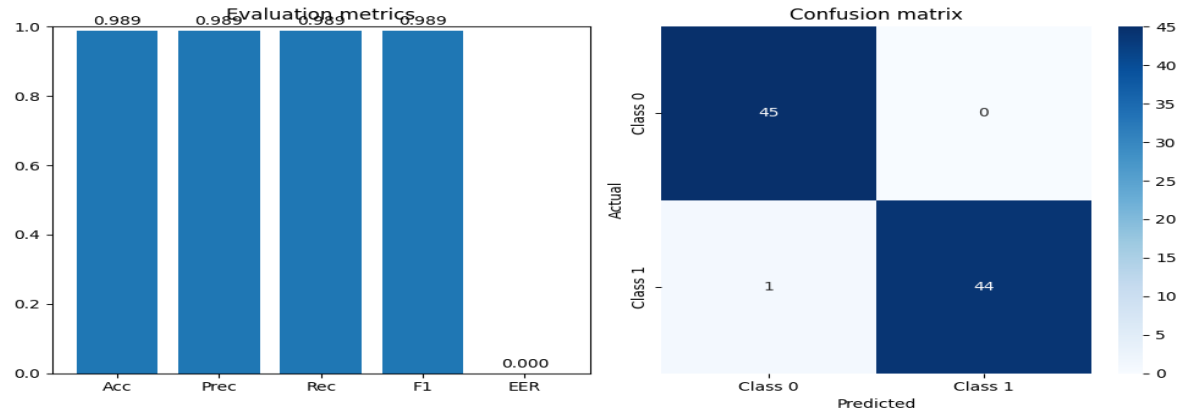

Figure 3: presentation of EER and confusion matrix for GOA

Plot of train and validation accuracy and loss is presented in Figure 4.

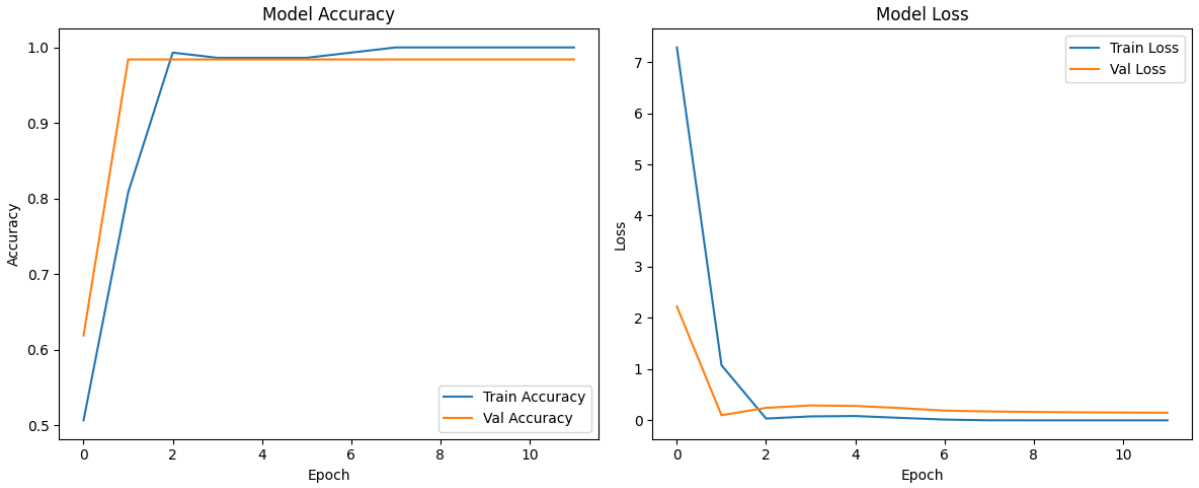

Figure 4: plot of accuracy and loss for GOA

A Classification Report is represented in Table 1.

Table 1: classification report for GOA

	Precision	Recall	f1-score
0	0.98	1.00	0.99
1	1.00	0.98	0.99
Accuracy		0.99	
macro avg	0.99	0.99	0.99
weighted avg	0.99	0.99	0.99

B. Firefly optimization algorithm(FOA): the keystroke dynamics problem is solved with 100% accuracy and an EER of 0.00 as shown in Figure 5.

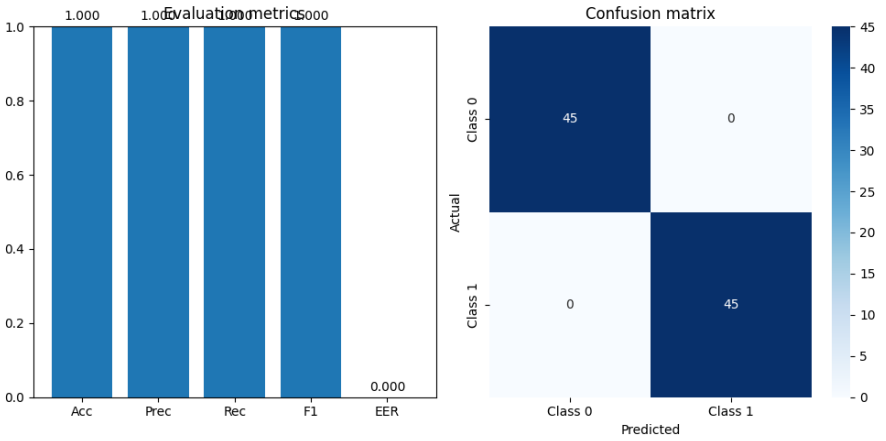

Figure 5: presentation of result and EER by FOA for keystroke dynamics

A plot of model accuracy and loss is presented in Figure 6.

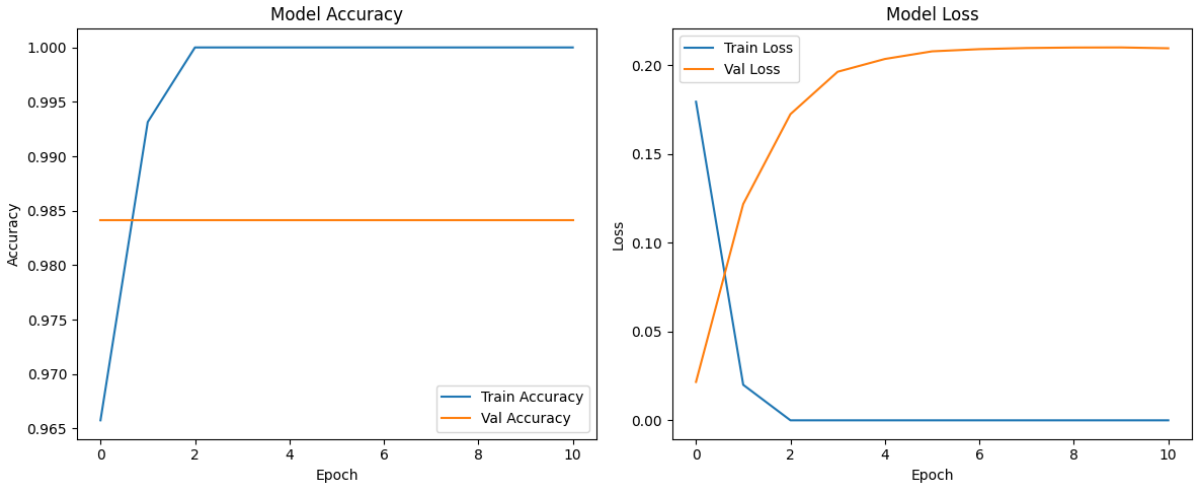

Figure 6: plot of accuracy and loss curve for FOA

Classification Report for FOA is presented in Table 2.

Table 2: classification report for FOA for solving KD problem

	Precision	Recall	f1-score
0	1.00	1.00	1.00
1	1.00	1.00	1.00
Accuracy		1.00	
macro avg	1.00	1.00	1.00
weighted avg	1.00	1.00	1.00

C. Whale Optimization algorithm(WOA): keystroke dynamics problem is solved using WOA with an accuracy of 96%and an EER of 0.043 as shown in Figure 7.

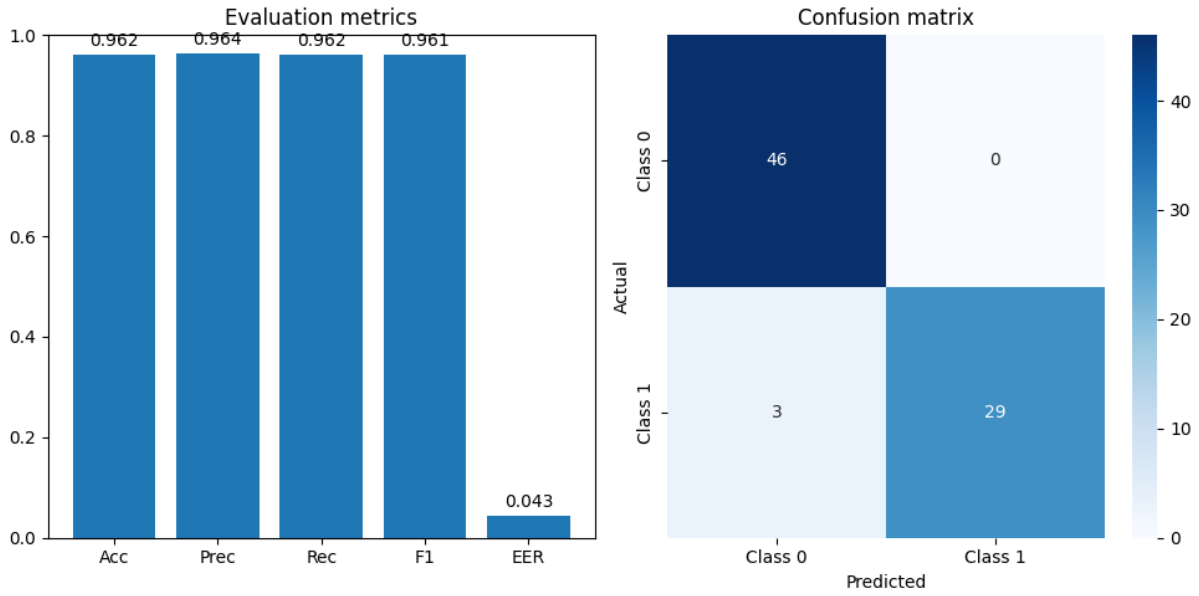

Figure 7: presentation of result values and confusion matrix when solved using WOA

Plot of model accuracy and loss is presented in Figure 8.

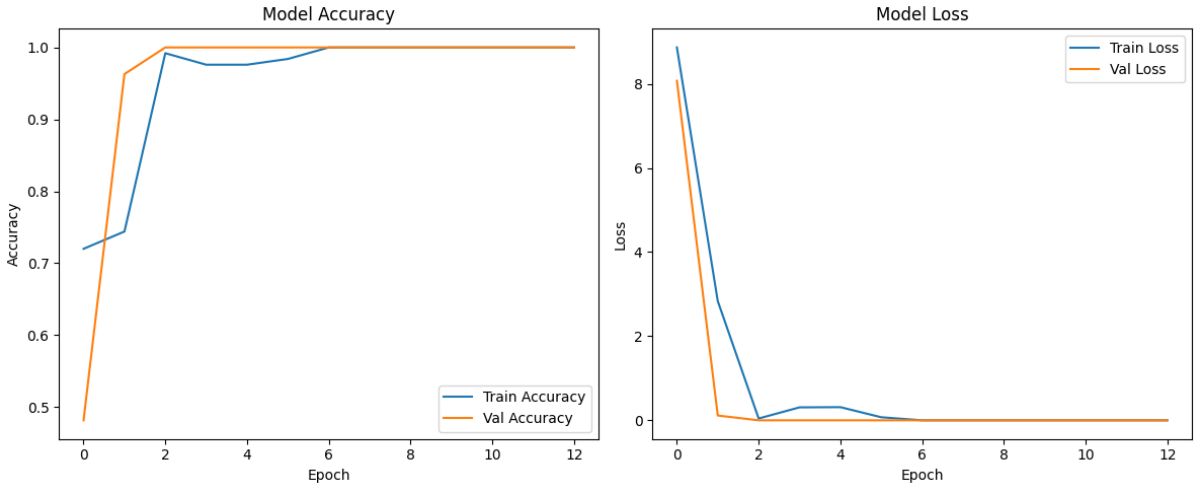

Figure 8: presentation of accuracy and loss curve when solved with WOA

Presentation of Classification Report is presented in Table 3.

Table 3: Classification Report when KD problem solved with WOA

	Precision	Recall	f1-score
0	0.94	1.00	0.97
1	1.00	0.91	0.95
Accuracy		0.96	
macro avg	0.97	0.95	0.96
weighted avg	0.96	0.96	0.96

D. Enhanced Wolf Optimization Algorithm(EWOA): the static keystroke dynamics problem is solved using EWOA with an accuracy of 99% and an EER of 0.00 as shown in Figure 9.

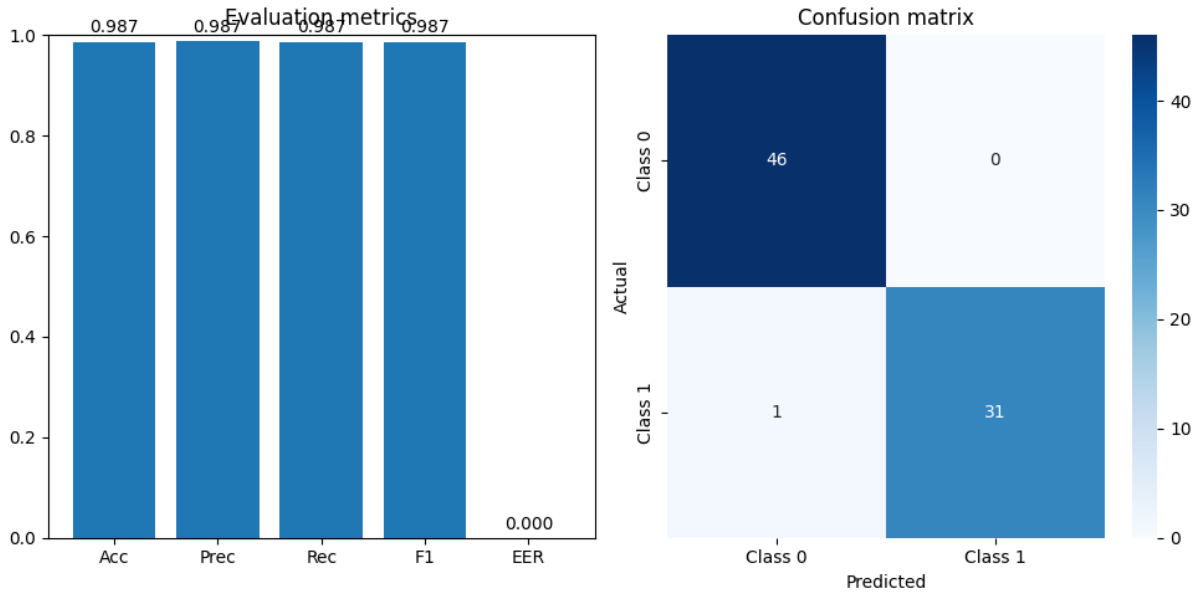

Figure 9: presentation of result and confusion matrix when solved with EWOA

Plot of accuracy and loss curve when static KD problem is solved with EWOA is presented in Figure 10.

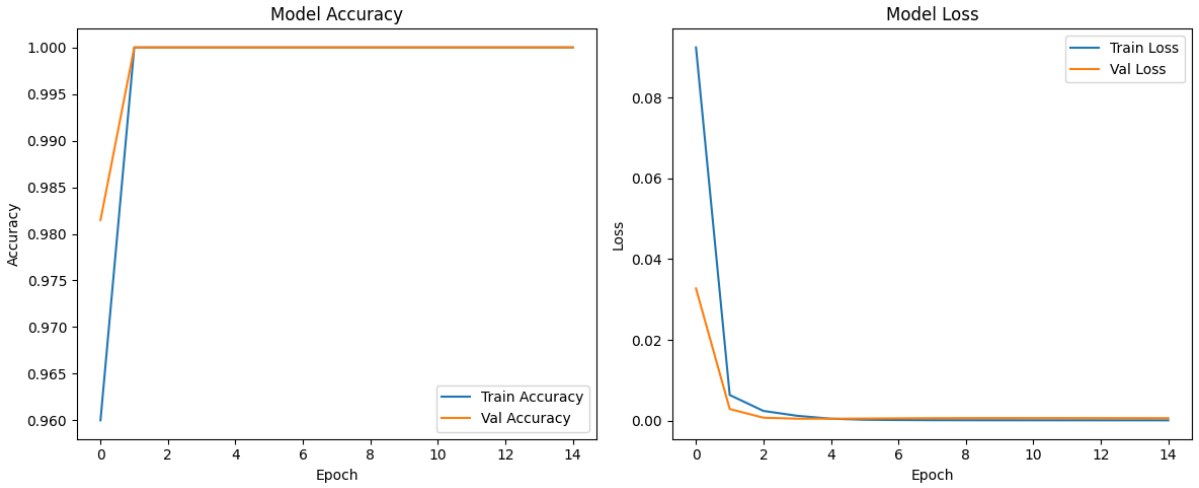

Figure 10: plot of accuracy and loss curve when KD problem is solved with EWOA

A classification report is presented in Table 4.

Table 4: Classification Report when KD problem solved with EWOA

	Precision	Recall	f1-score
0	0.98	1.00	0.99
1	1.00	0.97	0.98
Accuracy		0.99	
macro avg	0.99	0.98	0.99
weighted avg	0.99	0.99	0.99

A comparison is presented in Table 5 between problem of KD solved in this research paper and by [23](the best performance is given by CatBoost algorithm).

Table 5: a comparison between different models

Method used	Accuracy	EER
CNN with GOA	99.00	0.00%
CNN with FOA	100	0.00%
CNN with WOA	96	0.043%
CNN with EWOA	99	0.00%
CatBoost[23]	99.95	0.65%

7. CONCLUSION

The static keystroke dynamics problem is an important step which is performed at the time of login. A user types the username/password by his/her own way. This typing pattern is matched with the values store in database. In this research paper we solved this problem with deep learning approach where windowing concept and different metaheuristic algorithms for optimization are used. The best results are achieved with FireFly optimization algorithm with an accuracy of 100% and an EER of 0.00. Future research will be focused on more static keystroke dynamics dataset and authentication using free-text with different enhanced optimization techniques.

REFERENCES

[1] Shadman, R., Wahab, A. A., Manno, M., Lukaszewski, M., Hou, D., & Hussain, F. (2023). Keystroke dynamics: Concepts, techniques, and applications. *ACM Computing Surveys*.

[2] Roy, S., Pradhan, J., Kumar, A., Adhikary, D. R. D., Roy, U., Sinha, D., & Pal, R. K. (2022). A systematic literature review on latest keystroke dynamics based models. *IEEE Access*, *10*, 92192-92236.

[3] Kasprowski, P., Borowska, Z., & Harezlak, K. (2022). Biometric identification based on keystroke dynamics. *Sensors*, *22*(9), 3158.

[4] Soni, J., & Prabakar, N. (2021, December). KeyNet: enhancing cybersecurity with deep learning-based LSTM on keystroke dynamics for authentication. In *International Conference on Intelligent Human Computer Interaction* (pp. 761-771). Cham: Springer International Publishing.

[5] Risto, H. N., Bos, S., & Graven, O. H. (2024, December). Keystroke Dynamics Authentication with MLP, CNN, and LSTM on a Fixed-Text Data Set. In *2024 8th Cyber Security in Networking Conference (CSNet)* (pp. 76-82). IEEE.

[6] Malinowski, M., & Krawczyk-Borysiak, Z. (2024). Recognizing User Emotion Based on Keystroke Dynamics. *Przegląd Elektrotechniczny*, (6).

[7] Shekhawat, K., & Bhatt, D. P. (2022). Machine learning techniques for keystroke dynamics. In *Proceedings of Data Analytics and Management: ICDAM 2021, Volume 2* (pp. 217-227). Springer Singapore.

[8] Wang, X., & Hou, D. (2024). Enhancing Keystroke Dynamics Authentication with Ensemble Learning and Data Resampling Techniques. *Electronics*, *13*(22), 4559.

[9] Kamra, A., Khurana, S., & Goel, A. (2025, March). Keystroke Dynamics Based User Authentication Focusing on Fixed Text Approaches. In *2025 3rd International Conference on Device Intelligence, Computing and Communication Technologies (DICCT)* (pp. 427-432). IEEE.

[10] Amin, S., & Di Iorio, C. (2025). A Review of Several Keystroke Dynamics Methods. *arXiv preprint arXiv:2502.16177*.

[11] Ali, M. L., Thakur, K., & Obaidat, M. A. (2022). A hybrid method for keystroke biometric user identification. *Electronics*, *11*(17), 2782.

[12] Monrose, F., & Rubin, A. D. (2000). Keystroke dynamics as a biometric for authentication. *Future Generation computer systems*, *16*(4), 351-359.

[13] Altwaijry, N. (2023). Authentication by keystroke dynamics: The influence of typing language. *Applied Sciences*, *13*(20), 11478.

[14] Medvedev, V., Budžys, A., & Kurasova, O. (2025). A decision-making framework for user authentication using keystroke dynamics. *Computers & Security*, 104494.

[15] Chang, H. C. (2021). Keystroke dynamics based on machine learning.

[16] Bhasin, N., & Tarar, S. (2021). Three-layer authentication in keystroke dynamics using time based tool. In *Journal of Physics: Conference Series* (Vol. 1714, No. 1, p. 012030). IOP Publishing.

[17] Budžys, A., Kurasova, O., & Medvedev, V. (2025). Integrating deep learning and data fusion for advanced keystroke dynamics authentication. *Computer Standards & Interfaces*, *92*, 103931.

[18] Brekke, S., & Bours, P. (2024, November). Continuous Age Detection using Keystroke Dynamics. In *Norsk IKT-konferanse for forskning og utdanning* (No. 3).

[19] PUTRA, S. R., & CHOWANDA, A. (2025). KEYSTROKE DYNAMICS ON MULTI-SESSION AND UNCONTROLLED SETTINGS USING CNN BI-LSTM. *Journal of Theoretical and Applied Information Technology*, *103*(2).

[20] Bhasin, N., Sharma, S. K., & Mishra, R. (2025). Keystroke dynamics and quantum machine learning. *International Journal of Biometrics*, *17*(1-2), 132-150.

[21] Piugie, Y. B. W., Di Manno, J., Rosenberger, C., & Charrier, C. (2022, September). Keystroke dynamics based user authentication using deep learning neural networks. In *2022 International Conference on Cyberworlds (CW)* (pp. 220-227). IEEE.

[22] 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.

[23] AbdelRaouf, H., Chelloug, S. A., Muthanna, A., Semary, N., Amin, K., & Ibrahim, M. (2023). Efficient convolutional neural network-based keystroke dynamics for boosting user authentication. *Sensors*, *23*(10), 4898.

[24] O'shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

[25] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[26] Pan, J. S., Zhang, L. G., Wang, R. B., Snášel, V., & Chu, S. C. (2022). Gannet optimization algorithm: A new metaheuristic algorithm for solving engineering optimization problems. *Mathematics and Computers in Simulation*, *202*, 343-373.

[27] Johari, N. F., Zain, A. M., Noorfa, M. H., & Udin, A. (2013). Firefly algorithm for optimization problem. *Applied Mechanics and Materials*, *421*, 512-517.

[28] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in engineering software*, *95*, 51-67.

[29] Hou, Y., Gao, H., Wang, Z., & Du, C. (2022). Improved grey wolf optimization algorithm and application. *Sensors*, *22*(10), 3810.