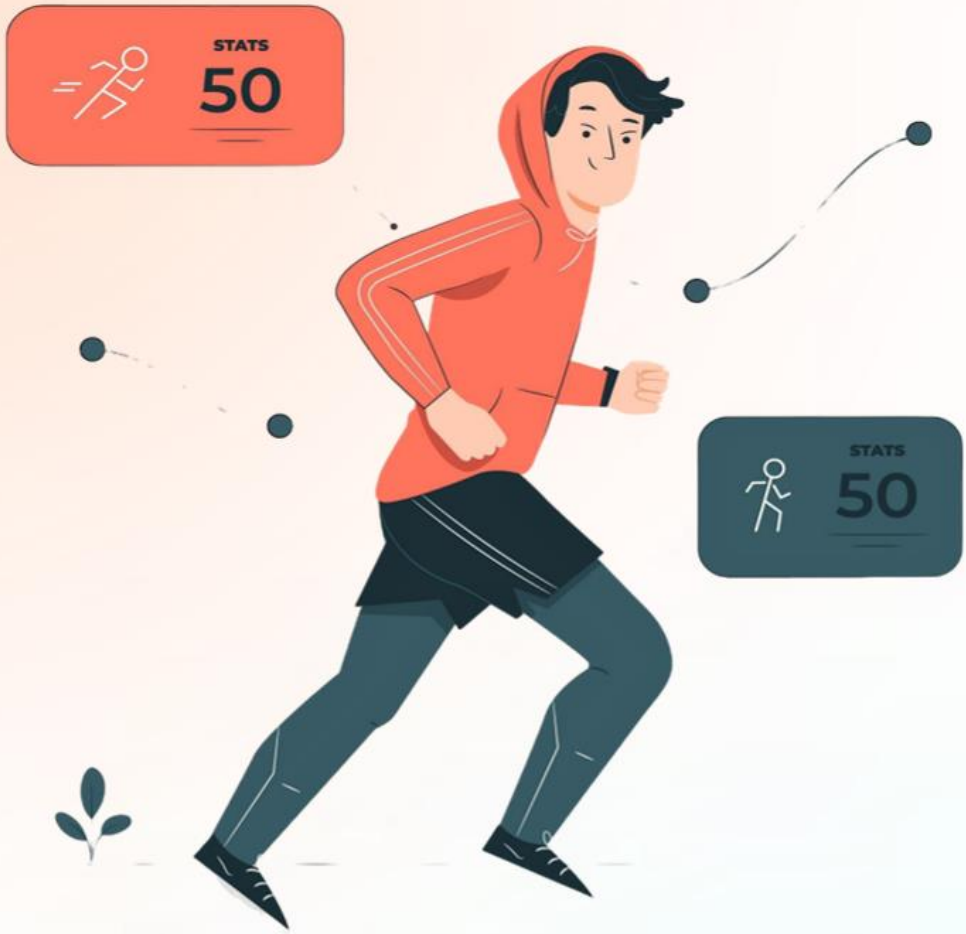
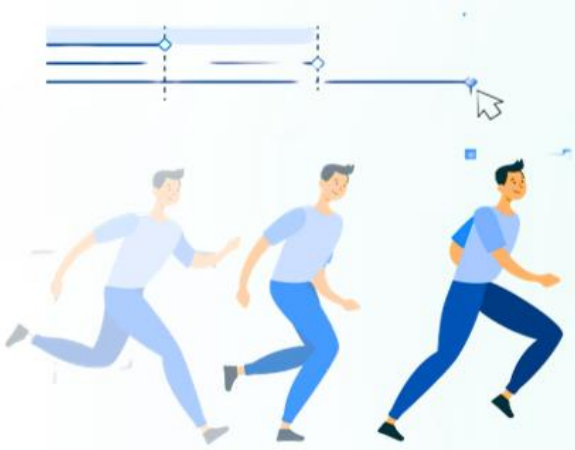
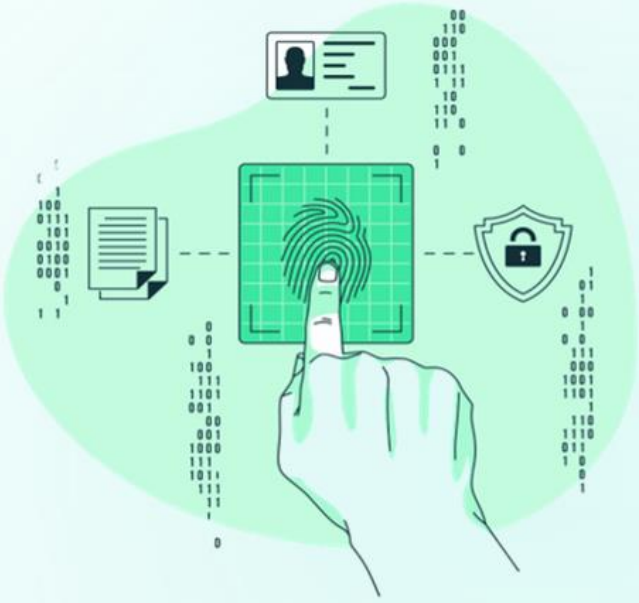


Acceleration Based User Authentication





Acknowledgement

We would like to express our sincere appreciation to our module leader,
Dr. Neamah Al-Naffakh
for his invaluable support and guidance. We were thankful to our team members who are actively contribute to make this entire group project success.

~~~~~~~
~~~~~~~

Table of Contents

1. Introduction

The growing popularity of wearable devices such as smartwatches and fitness trackers has opened new frontiers in secure and continuous user authentication. The necessity for strong security measures is highlighted by the fact that there are currently more mobile devices in use worldwide than the global population, and that these devices have access to vital private data, such as financial and medical records. Traditional authentication methods like passwords and PINs, are often inconvenient and vulnerable to common security threats. This has led to the exploration of transparent and unobtrusive biometric systems that authenticate users based on behavioral patterns captured through wearable computing devices. (Al-Naffakh, et al., 2017)

Among these advancements, acceleration-based user authentication emerged as a particularly promising approach. By leveraging motion data captured by accelerometers and gyroscopes embedded in wearable devices, this approach examines behavioral characteristics such as human gait, arm movement, or hand gesture. Unlike physiological biometrics (such as fingerprints or facial recognition), individuals possess these qualities by nature, which makes them appropriate for secure authentication while reducing the requirement for user engagement.

This paper investigates the use of acceleration-based data for user authentication, presenting it as a viable alternative to traditional methods. By examining subtle variations in how individuals move or interact with devices, this approach aims to reduce the risk of identity theft and unauthorized access. Furthermore, the project explores the accuracy, precision, recall etc. of using accelerometer data for authentication, considering various factors.

1.1. Synopsis,

The **Introduction** outlines the need and importance, followed by a **Background** section reviewing recent advancements in acceleration-based user authentication systems. The **Testing Methodology** describes the experimental setup, the approach we used, domain comparisons. The **Evaluation** section includes an analysis of user patterns, system performance based on different metrics. **Optimization** focuses on improving model robustness , usability and improving evaluation the metrics while the **Conclusions** assess the viability of this approach. Finally, the **References** include citations and **Appendix** include MATLAB code to ensure transparency and reproducibility.

1.2. Keywords

SVM – Support Vector Machine

GA – Genetic Algorithm

LOUO – Leave One User Out

DTW - Dynamic Time Wasting

AUC – Area Under Curve

NN – Neural Network

FAR – False Acceptance Rate

FRR – False Rejection Rate

EER – Equal Error Rate

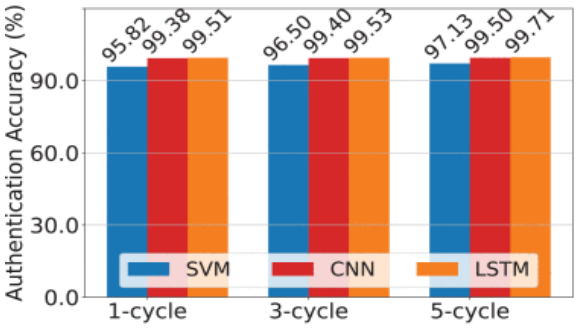
PCA – Principal Component Analysis

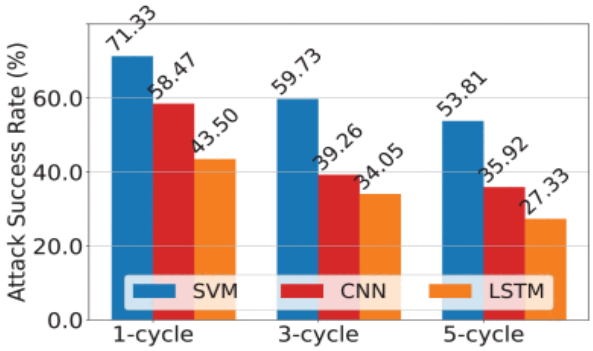
2. Literature Review

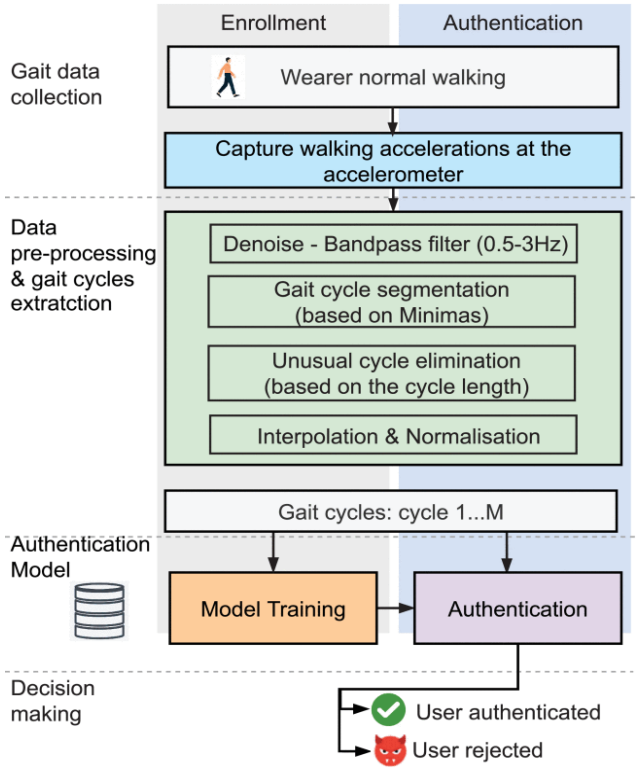
In today's digital era, demand for secure and seamless user authentication has surged. Acceleration based authentication, which utilizes motion data from devices like smartphones and wearables, has become favorable option because of its discreet and user-friendliness. They analyze unique motion patterns, like walking styles or device usage behaviors, to provide personalized layer of security. This review highlights recent advancements, focusing on techniques, classifiers, and system effectiveness.

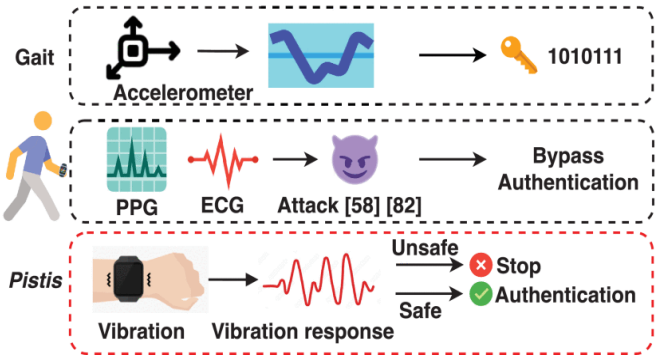
1. Gait User Authentication System on Wearable Devices Using Vibration Pattern and Acceleration Based Data

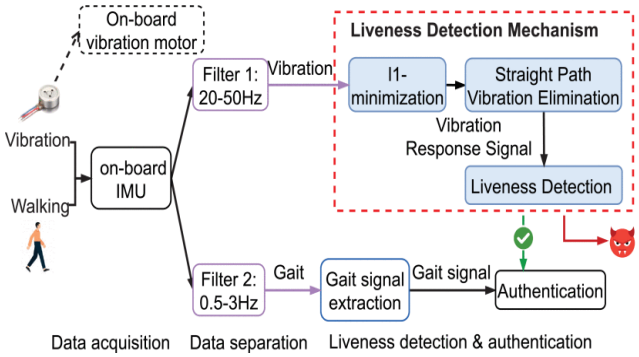
The paper addresses an authentication protocol named ***Pistis*** that embedded gait biometrics plus liveness detection mechanisms, using smart wearables equipped with motion sensors, that collect gait acceleration signals consistently. This utilized the **Support Vector Machine**, **Convolutional Neural Network**, and **Long-Short Term Memory** to train the models based on 760 gait sequences of 50 people. As performance metrics, **SVM, CNN, LSTM** approaches had accuracies of 96.5%, 99.4% and 99.53% respectively. It acquired an **Equal Error Rate** of 7% in signal correlation approach and EER of 5% With 3-D acceleration signals. (Wei Song, et al., 2022)





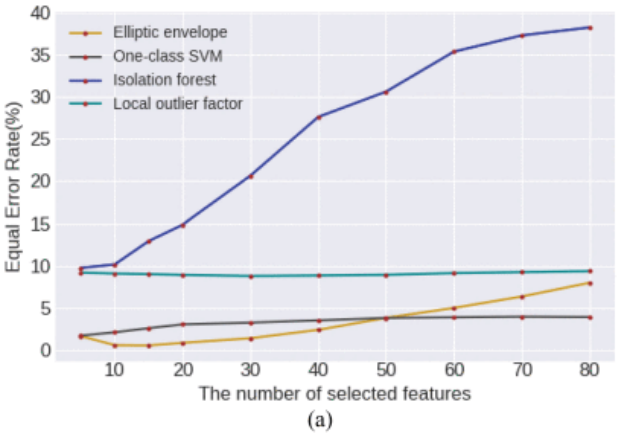


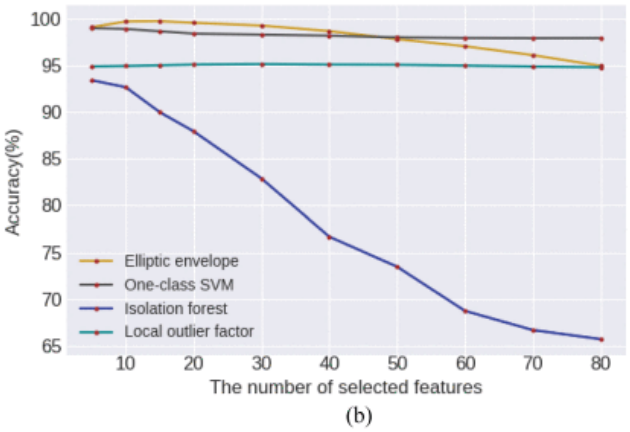




2. Multisensory-Based Continuous User Authentication of Smartphone Users

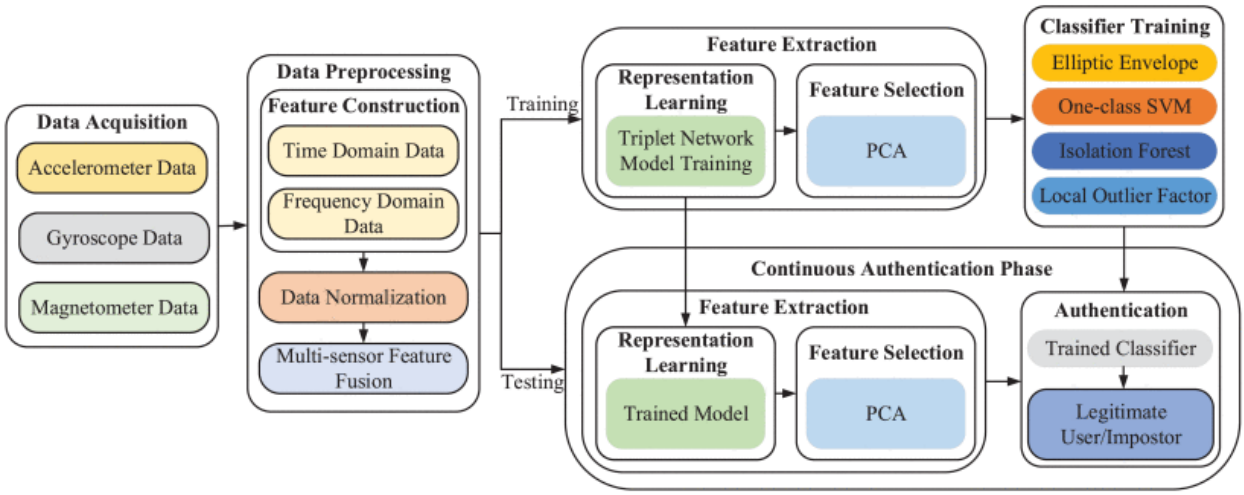
The research explores new techniques that integrate manual construction with deep metric learning to accomplish feature extraction for continuous user authentication based on behavioral biometrics. The raw time-series data were collected from accelerometer, gyroscope, and magnetometer sensors into 69 statistical features. This uses an **Elliptic Envelope Algorithm** to classify the user as legitimate or an impostor. This approach obtained an average accuracy of 99.71%, average **EER** of 0.56% on hand movement, movement, orientation, and **Grasp** dataset, along with an average accuracy of 99.59% and average **EER** of 0.61% on the **BrainRun** dataset. (Hu, et al., 2022)

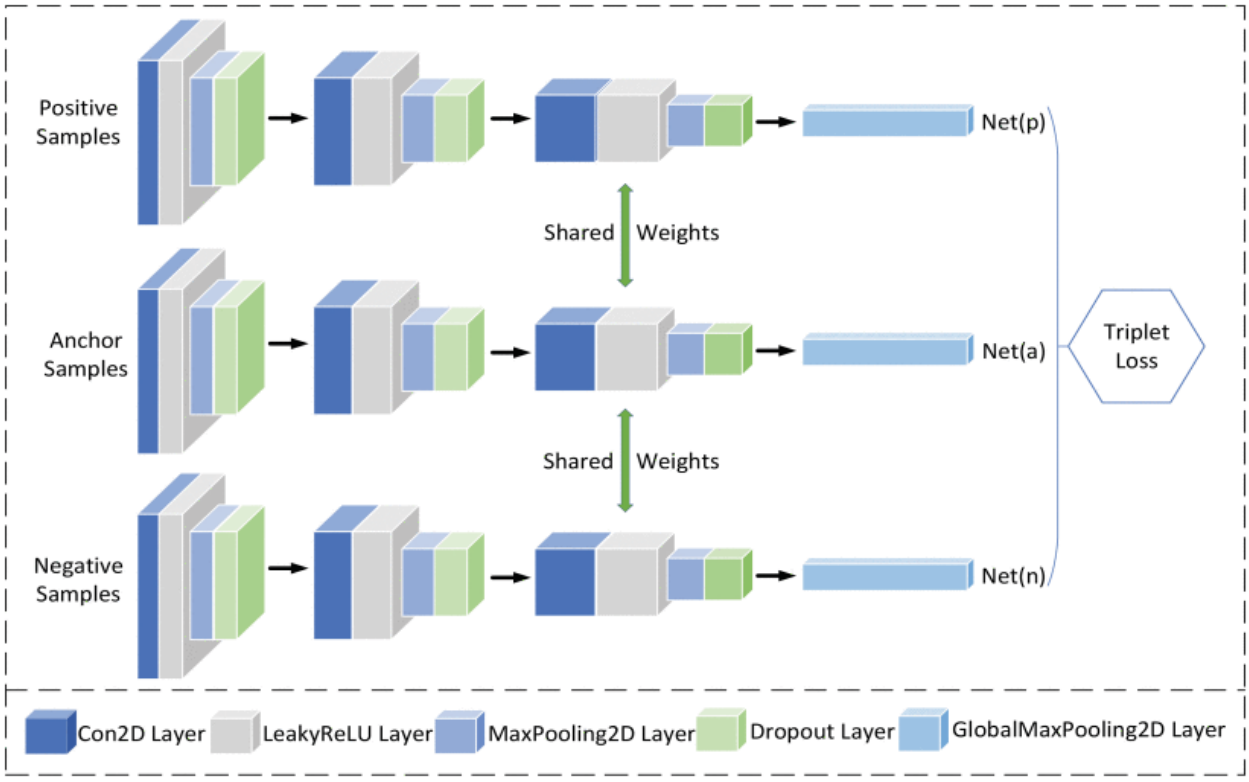


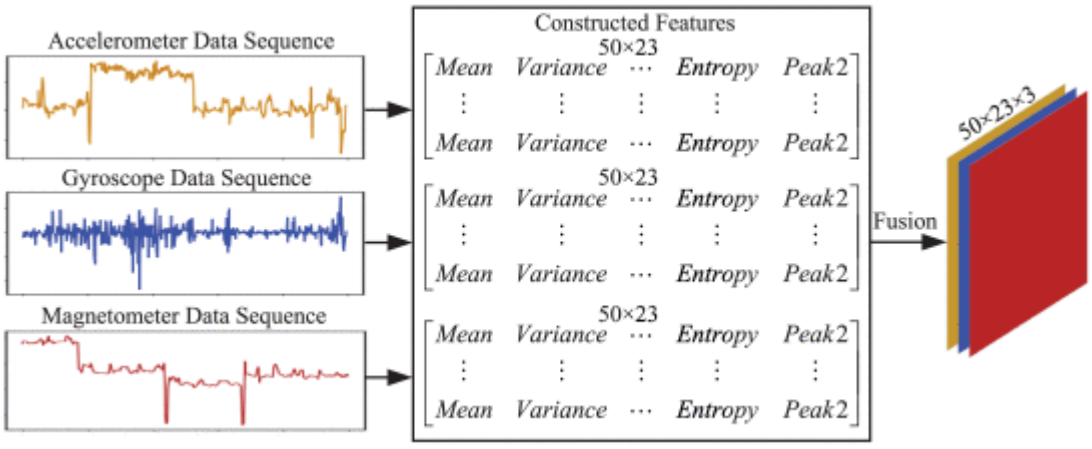


Model	FAR (%)	FRR (%)	EER (%)	Params
ResNet-50[62]	2.04	2.83	2.65	24.5M
ResNet-101[62]	1.83	2.48	2.06	42.6M
EfficientNet[63]	2.81	3.23	3.07	29.1M
MobileNetV3-small[64]	2.07	3.42	2.69	2.5M
MobileNetV3-large[64]	1.83	2.83	2.24	5.23M
MnasNet[65]	2.26	2.67	2.37	2.2M
This work	**0.62**	**0.10**	**0.56**	**0.3M**

Selected features	FAR(%)	FRR (%)	EER (%)	Accuracy (%)
5	1.95	**0.12**	1.83	98.94
10	0.69	0.13	0.64	99.56
15	**0.64**	0.13	**0.61**	**99.59**
20	0.96	0.13	0.92	99.41
30	1.67	0.12	1.57	99.12
40	3.02	0.12	2.79	98.53
50	4.98	0.11	4.45	97.69
60	6.39	0.12	5.72	96.91
70	8.61	0.12	7.15	95.98
80	11.07	0.12	8.95	94.82

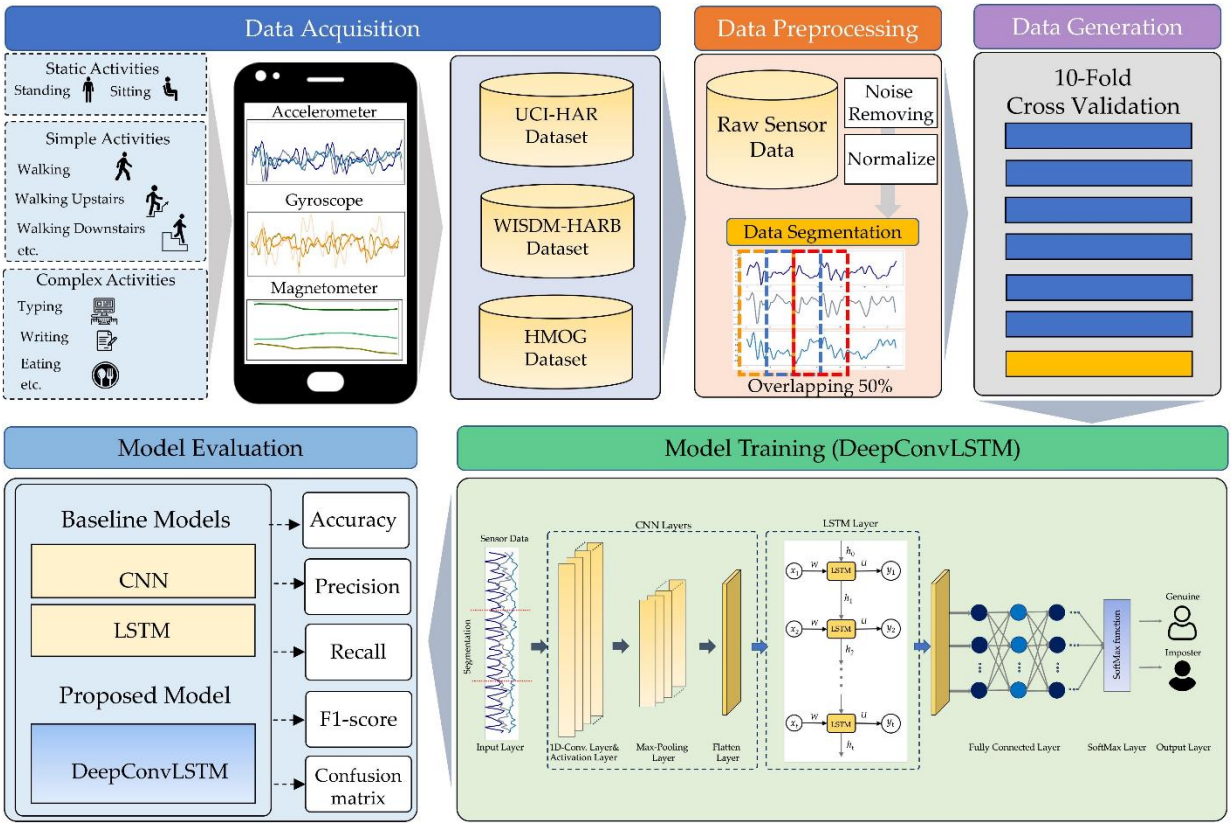

Data Acquisition
Accelerometer Data
Gyroscope Data
Magnetometer Data
Data Preprocessing
Feature Construction
Time Domain Data
Frequency Domain Data
Data Normalization
Multi-sensor Feature Fusion
Training
Testing
Feature Extraction
Representation Learning
Triplet Network Model Training
Feature Selection
PCA
Classifier Training
Elliptic Envelope
One-class SVM
Isolation Forest
Local Outlier Factor
Continuous Authentication Phase
Feature Extraction
Representation Learning
Trained Model
Feature Selection
PCA
Authentication
Trained Classifier
Legitimate User/Impostor

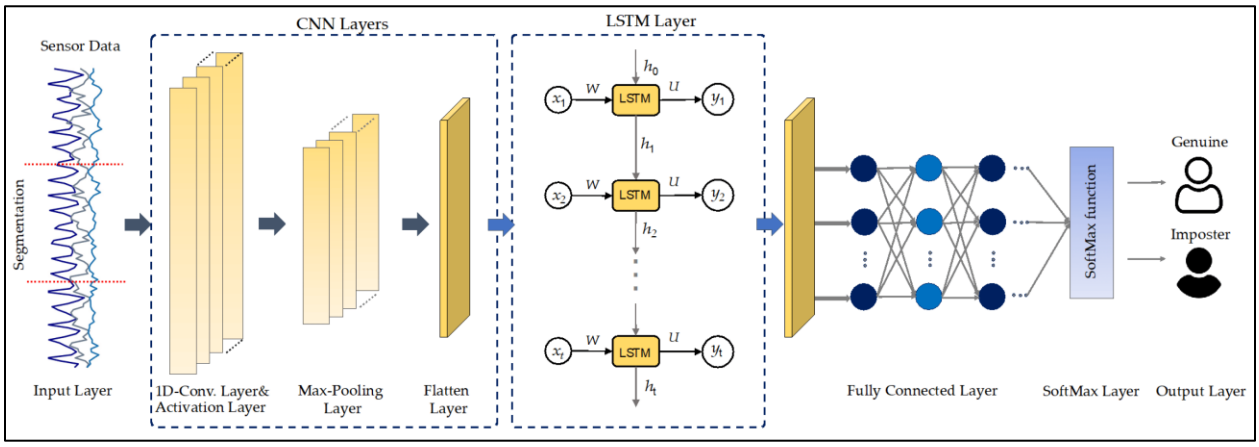

Positive Samples
Anchor Samples
Negative Samples
Shared Weights
Shared Weights
Net(p)
Net(a)
Net(n)
Triplet Loss
Con2D Layer
LeakyReLU Layer
MaxPooling2D Layer
Dropout Layer
GlobalMaxPooling2D Layer

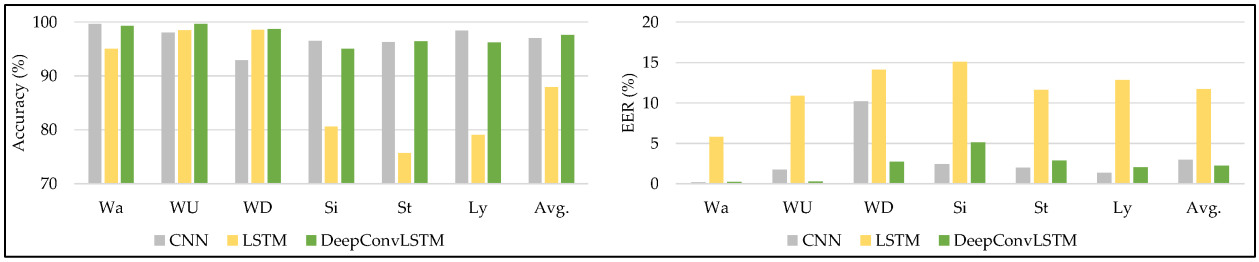

Accelerometer Data Sequence
Gyroscope Data Sequence
Magnetometer Data Sequence
Constructed Features
50×23
Mean Variance ⋯ Entropy Peak2
Fusion
50×23×3

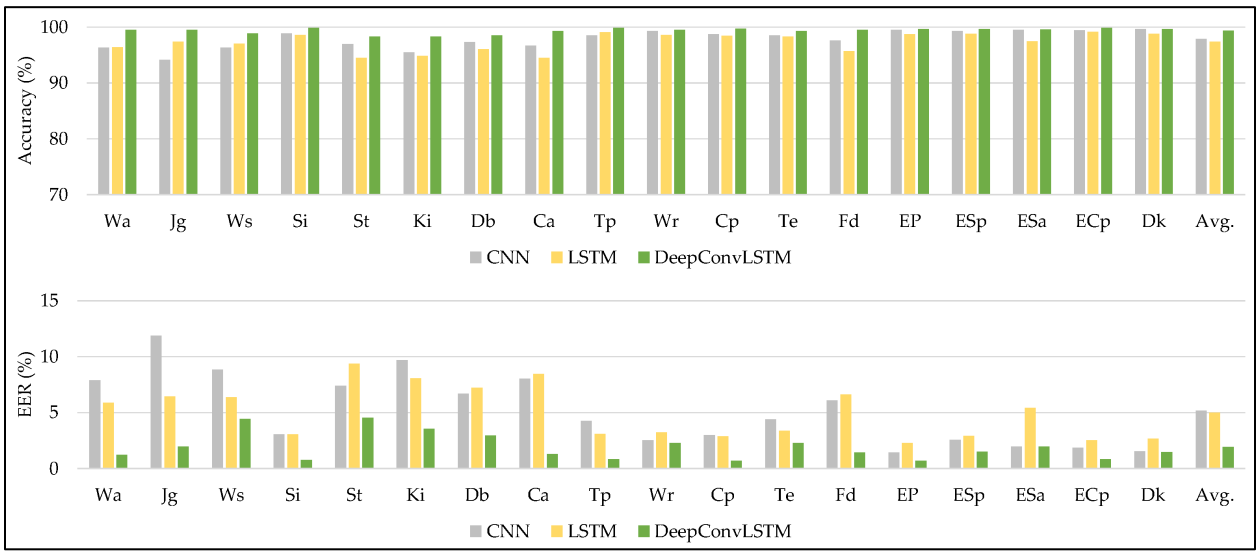
3. Deep Learning Approach for Continuous User Authentication Based on Activity Acceleration Patterns Using Multiple Sensor Data

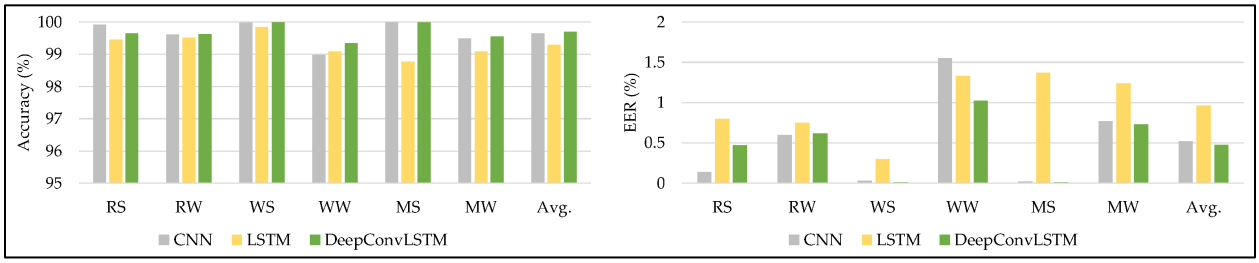
This study presents a novel continuous authentication system named **DeepAuthen**, which recognizes smartphone users through their physical activity patterns determined by the accelerometer, gyroscope, and magnetometer. Data sampling as training and testing was done using a **10-fold cross-validation** technique. They've used various deep learning classifiers, including their suggested deep learning model called **DeepConvLSTM** on the three benchmark datasets **UCI-HAR**, **WISDM-HARB**, and **HMOG**. This process was done using **Convolutional Layers** and a **LSTM** layer, and obtained an average **Accuracy** of 97%, 98.5%, 96.5% and average **EERs** of 2.5%, 2.2%, 0.5% for **UCI-HAR, WISDM-HARB**, and **HMOG** datasets respectively. (Sakorn Mekruksavanich, 2021)









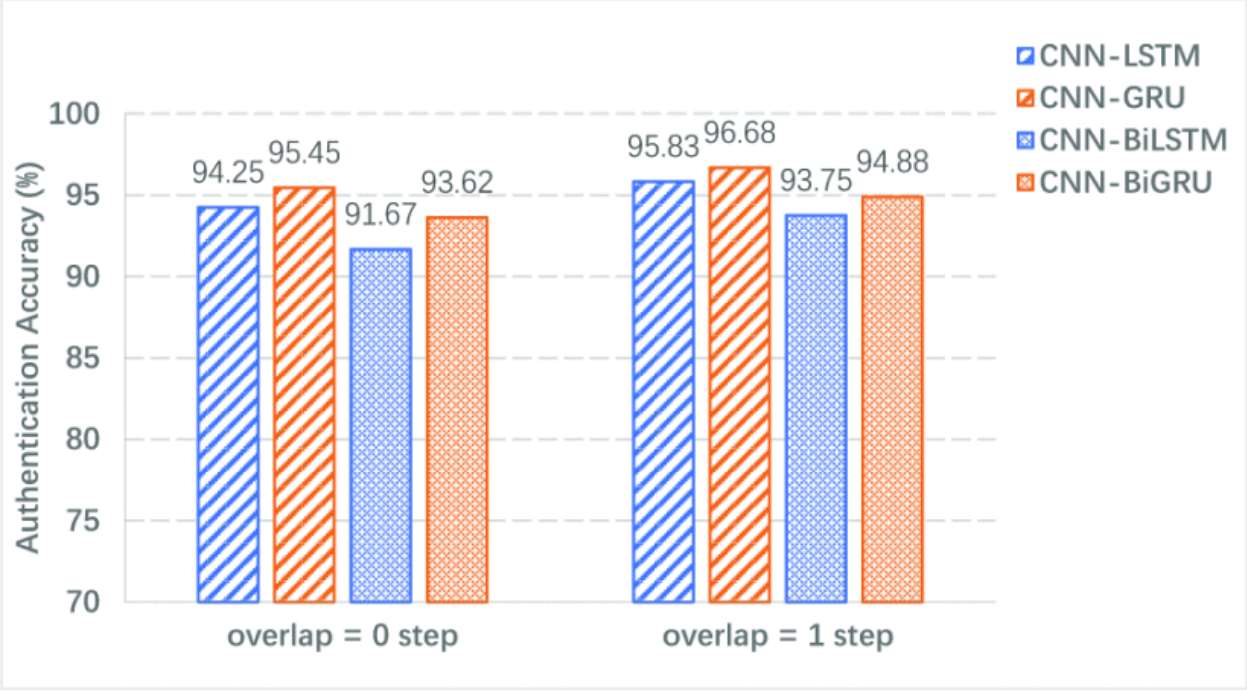


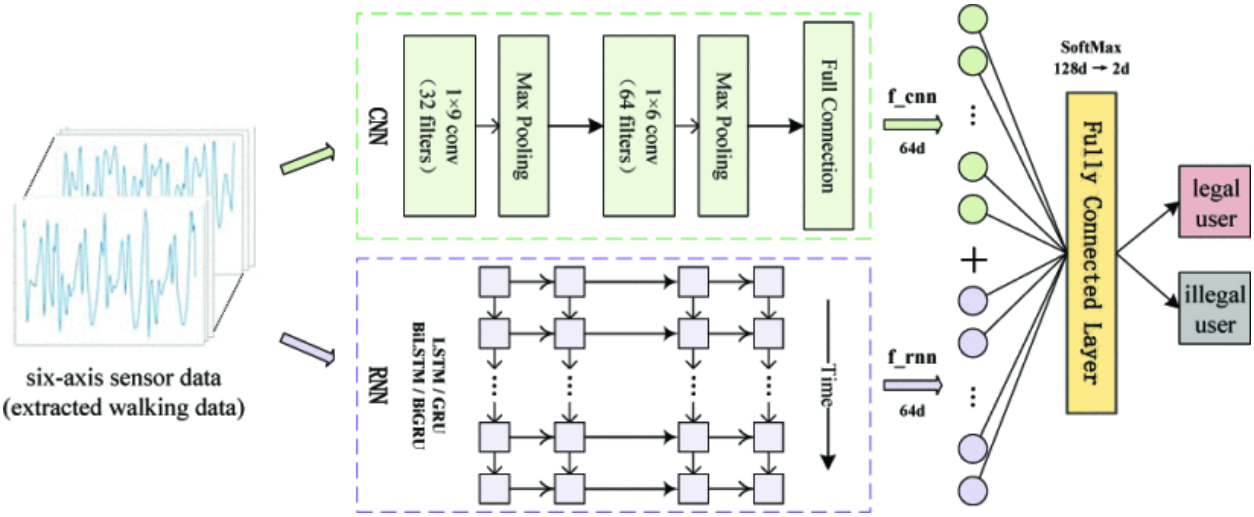
4. Smartphone Gait Authentication Based on Activity Recognition Task

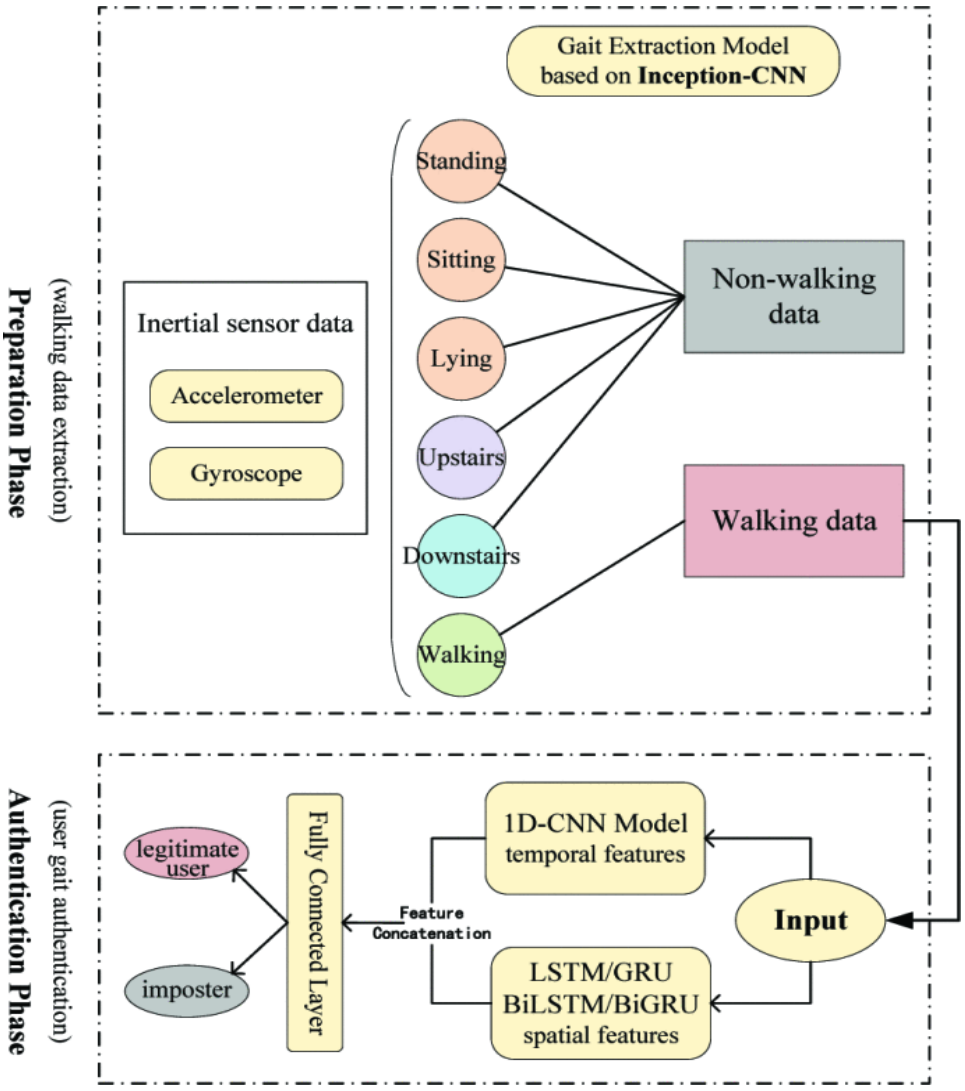
This study proposes an enhanced gait extraction model incorporating an upgraded **Inception-CNN** and a three-layer **1D-CNN** for acceleration-based activity recognition. The model extracts acceleration-based information related to walking from30 users. It employs a 10-fold cross-validation during training. A **CNN-RNN** hybrid model is constructed to compare **RNN** variants, including **LSTM, GRU, Bi-LSTM,** and **Bi-GRU**, on the **UCI-HAR** dataset. The model achieves 97.52% walking recognition accuracy, 96.68% user authentication accuracy, and an **EER** of 2.86%. (Cheng, et al., 2024)

Hybrid Model	Overlap = 0 step			Overlap = 1 step		
	FAR	FRR	EER	FAR	FRR	EER
CNN-LSTM	**6.64%**	2.26%	3.20%	5.88%	2.08%	2.97%
CNN-GRU	6.75%	**1.52%**	**3.11%**	5.97%	**2.01%**	**2.86%**
CNN-BiLSTM	8.81%	3.45%	4.96%	7.25%	3.37%	4.18%
CNN-BiGRU	8.22%	2.10%	4.03%	**5.26%**	2.14%	3.49%

Ref.	Gait Extraction Model	Extraction Acc.	Authentication Acc.	Authentication EER
[6]	Simple 1D-CNN	89.31%	87.43%	7.10%
[17]	Deep 1D-CNN	92.89%	93.86%	4.11%
[7]	LSTM	90.17%	91.88%	6.57%
[9]	DeepConvLSTM	95.08%	93.35%	4.88%
[10]	CNN-GRU	96.20%	95.39%	3.43%
Ours	**Improved Inception-CNN + three-layer 1D-CNN**	**97.52%**	**96.68%**	**2.86%**

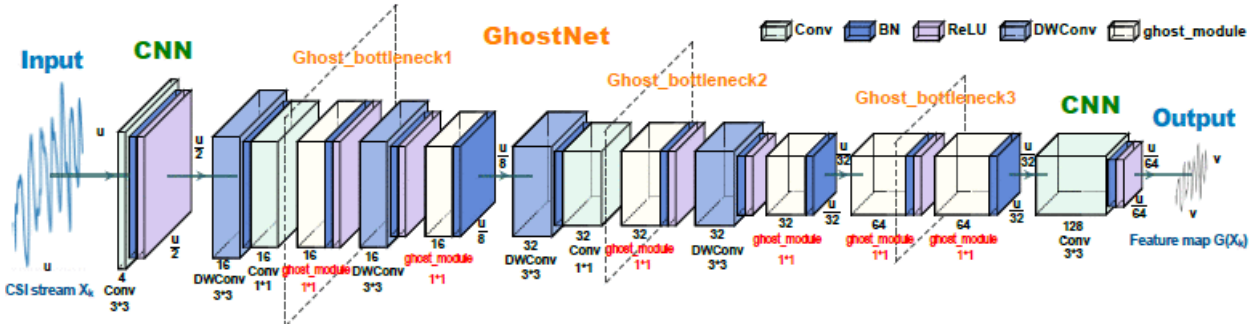


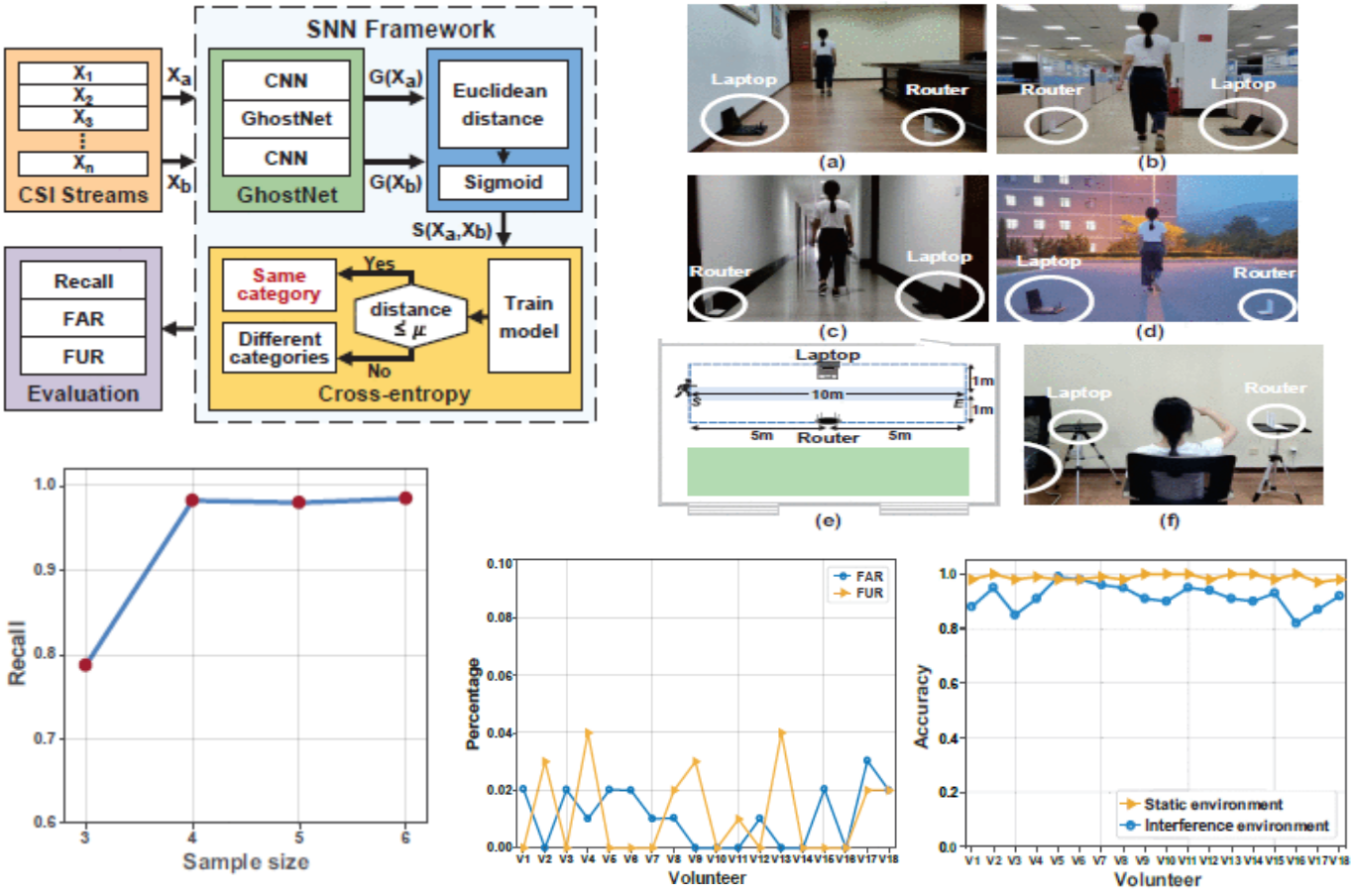




5. WiLCA : Mechanism for Accelerating Contactless User Authentication with Limited Data

The paper addresses a novel system called **WiLCA**, a lightweight wi-fi based contactless authenticator. It uses **Channel State Information (CSI)** stream selection method to capture human movement characteristics. Combination of both **Siamese Neural Network (SNN)** and **Ghost Network** to create an **Aggregation-SNN**. The **GhostNet (AGO)** model is used to enhance authentication speed. System obtained an average authentication accuracy of 98%, **FAR** of less than 3%, Recall rate of over 96% while reducing data size by at least 2.5X. (Lin, et al., 2022)





2.1. Reflections

Acceleration-based user authentication demonstrates promising recent advancements, leveraging gait biometrics using motion sensors, classifiers and deep learning techniques like SVM, CNN, RNN, SNN, Hybrid approaches and LSTM for acquire high accuracies above 96%. Integrating gait biometrics and liveness detection enhances the security. While recent methods achieve EERs as low as 5%, ongoing efforts address challenges like scalability, real-world applicability, and resistance to sophisticated attacks and risks for continuous user authentication.

3. Data

The collected acceleration data sequences of user's smartwatch were processed and converted from time-series acceleration data into samples. The data are broken down into 2 main domains as follows:

1. Time domain (88 features)
    - The acceleration data of x, y & z axis data sequences of each walk were converted into few statistical features like mean, standard deviation, etc. 88 of such features were generated for the time domain.
2. Frequency domain (43 features)
    - The frequency domain features were generated using the previously generated time domain data and Fast-Fourier Transform (FFT) was applied to get the frequency domain data.

In summary each sample of a user is a walk sequence of the user which were collected for 2 separate days from every user, and we've got 36 samples for each of the days collected for every user.

4. Testing Methodologies

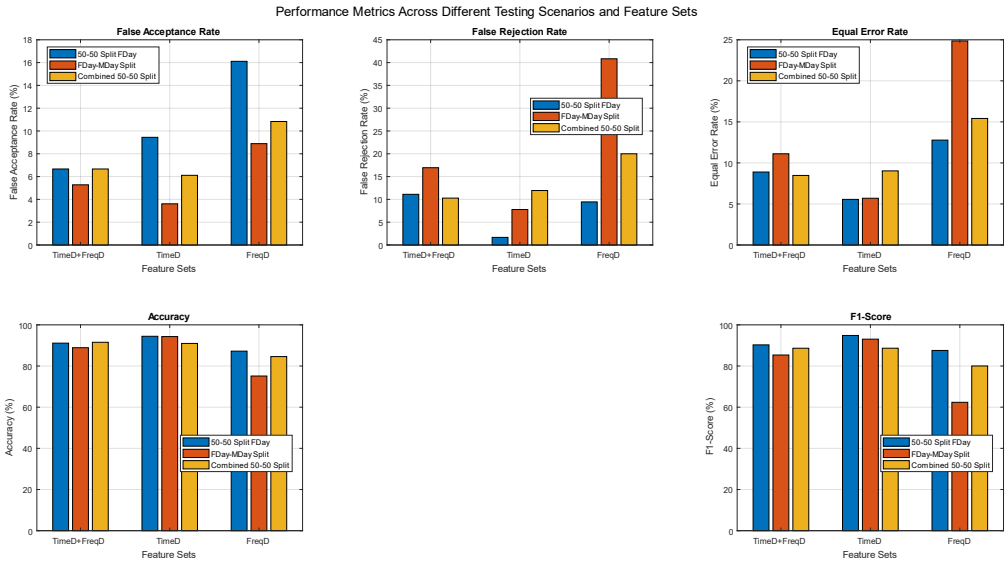
This section discusses the approaches used to evaluate the neural network models used for acceleration-based user authentication.

4.1. Data splitting

To get started, we compared different combinations of the days data to test how models will perform for each combination.

These train/test combinations were considered:

1. 50-50 Split **FDay**: **FDay** data splits 50% for training/testing.
2. **FDay-MDay** Split: **FDay** data for training & **MDay** data for testing.
3. Combined 50-50: Data from both days' splits 50% to training & testing.



The **FDay-MDay** split shows the weakest performance relative to the 50-50 Split of FDay & Combined 50-50 Split of FDay & MDay. This is most likely due to possible differences between the two days' data of users walking pattern, as there could be reasons like users who are having health concerns or any other issues which might affect their walking pattern. However, when it comes to the real-world scenario the FDay-MDay split is best for model evaluation, because models are trained once and used over time in most real-world applications. Therefore, we proceeded with FDay-MDay Split for evaluations.

4.2. Showing the findings and evaluation metrics

Here shown the findings and evaluations metrics for using 3 domains **TD ,FD , TDFD** with different training and testing combinations. Such as using only first day data. Combine both days data (**Fday** and **Mday**) using fist day for training and same day data for testing separately.

```
=== TimeD+FreqD Domain ===

Metric      50-50 Split FDay    FDay-MDay Split     Combined 50-50 Split
-------------------------------------------------------------------------

Accuracy     91.11%              88.89%              91.53%
Precision    93.16%              95.50%              91.74%
Recall       88.89%              83.06%              89.72%
Specificity  93.33%              94.72%              93.33%
F1Score      90.27%              85.35%              88.64%
MCC         0.8322              0.8061              0.8396
FAR           6.67%               5.28%               6.67%
FRR          11.11%              16.94%              10.28%
EER           8.89%              11.11%               8.47%
AUC         0.9111              0.8889              0.9153


=== TimeD Domain ===

Metric      50-50 Split FDay    FDay-MDay Split     Combined 50-50 Split
-------------------------------------------------------------------------

Accuracy     94.44%              94.31%              90.97%
Precision    91.80%              96.79%              94.22%
Recall       98.33%              92.22%              88.06%
Specificity  90.56%              96.39%              93.89%
F1Score      94.84%              93.05%              88.65%
MCC         0.8934              0.8988              0.8395
FAR           9.44%               3.61%               6.11%
FRR           1.67%               7.78%              11.94%
EER           5.56%               5.69%               9.03%
AUC         0.9444              0.9431              0.9097


=== FreqD Domain ===

Metric      50-50 Split FDay    FDay-MDay Split     Combined 50-50 Split
-------------------------------------------------------------------------

Accuracy     87.22%              75.14%              84.58%
Precision    86.70%              86.93%              88.99%
Recall       90.56%              59.17%              80.00%
Specificity  83.89%              91.11%              89.17%
F1Score      87.56%              62.37%              80.02%
MCC         0.7608              0.5515              0.7180
FAR          16.11%               8.89%              10.83%
FRR           9.44%              40.83%              20.00%
EER          12.78%              24.86%              15.42%
AUC         0.8722              0.7514              0.8458
```

Combination 1

=== TimeD + FreqD Domain ===

Metric	50-50 Split F_Day	F_Day-M_Day Split	Combined 50-50 Split
Accuracy	91.11%	88.89%	91.53%
Precision	93.16%	95.50%	91.74%
Recall	88.89%	83.06%	89.72%
Specificity	93.33%	94.72%	93.33%
F1Score	90.27%	85.35%	88.64%
MCC	0.8322	0.8061	0.8396
FAR	6.67%	5.28%	6.67%
FRR	11.11%	16.94%	10.28%
EER	8.89%	11.11%	8.47%
AUC	0.9111	0.8889	0.9153

Combination 2

=== TimeD Domain ===

Metric	50-50 Split F_Day	F_Day-M_Day Split	Combined 50-50 Split
Accuracy	94.44%	94.31%	90.97%
Precision	91.80%	96.79%	94.22%
Recall	98.33%	92.22%	88.06%
Specificity	90.56%	96.39%	93.89%
F1Score	94.84%	93.05%	88.65%
MCC	0.8934	0.8988	0.8395
FAR	9.44%	3.61%	6.11%
FRR	1.67%	7.78%	11.94%
EER	5.56%	5.69%	9.03%
AUC	0.9444	0.9431	0.9097

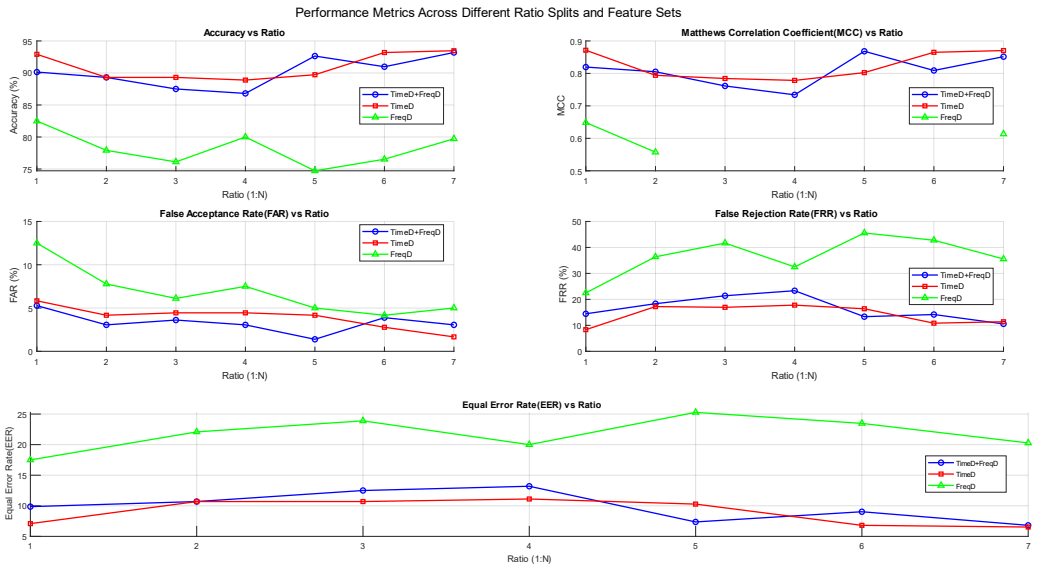
Combination 3

=== FreqD Domain ===

Metric	50-50 Split F_Day	F_Day-M_Day Split	Combined 50-50 Split
Accuracy	87.22%	75.14%	84.58%
Precision	86.70%	86.93%	88.99%
Recall	90.56%	59.17%	80.00%
Specificity	83.89%	91.11%	89.17%
F1Score	87.56%	62.37%	80.02%
MCC	0.7608	0.5515	0.7180
FAR	16.11%	8.89%	10.83%
FRR	9.44%	40.83%	20.00%
EER	12.78%	24.86%	15.42%
AUC	0.8722	0.7514	0.8458

4.3. Data ratio / percentage selection for creating training and testing sets

Next, it was needed to find a good ratio for legitimate user & imposter user data for a well-balanced model performance. We considered using the **Target : Imposter** ratios within the range **1:1 – 1:7**, and for all domains. In **1:N** ratio **1** means labelled legitimate user samples, and **N** means labelled imposter samples.



As per the results shown in above graph, a **1:5** ratio shows best performance while for frequency domain has shown overall weak performance but has contributed towards the reduction of FAR. Considering these facts, we used the **TimeD + FreqD dataset** for this system.

For the testing set, We used total **36** legitimate user samples from **Fday** file for training and the remaining **224** samples are imposters, out of them we select them by the ratio, and it selects equal number of samples from all the remaining imposters randomly. As example **1:1** means 36 **legitimate user** samples and 36 **imposter samples** (**4** random samples from remaining **9** imposters **4 * 9** = **36**) then we create the training template by combining all the selected samples according to ratio.

Ratio table (Data Sampling)

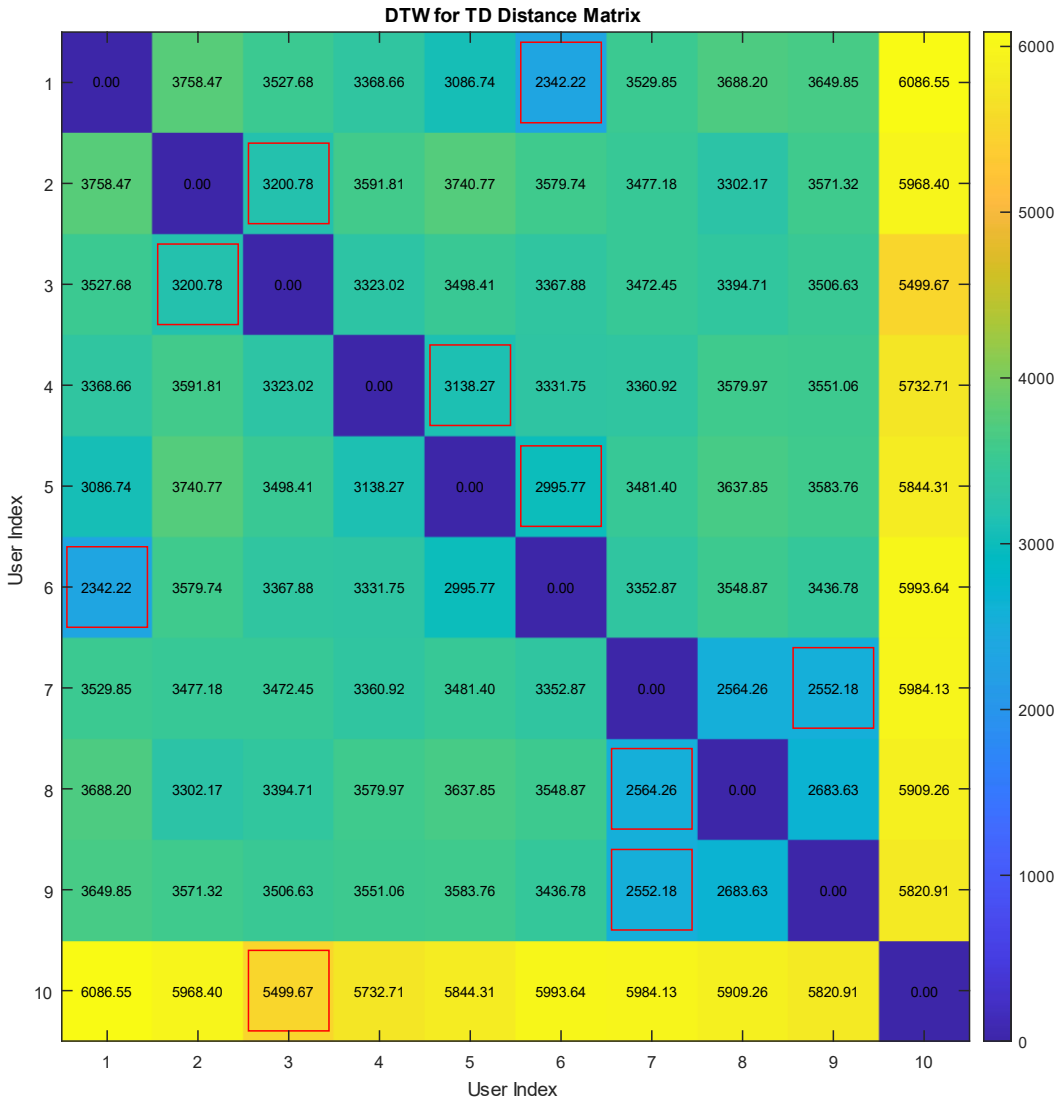
This table give full idea about how we select the data samples from each legitimate and 9 imposters.

Ratio	Total legitimates (1)	Total imposters (9)	N from each	Calculation
1:1	36	36	4	4 * 9 = 36
1:2	36	72	8	8 * 9 = 72
1:3	36	108	12	12 * 9 = 108
1:4	36	144	16	16 * 9 = 144
1:5	36	180	20	20 * 9 = 180
1:6	36	216	24	24 * 9 = 216
1:7	36	252	28	28 * 9 = 252

The tested outputs are shown for each ratio. We select **1:5** as the optimal ratio that give outputs best values such as low EER , FAR and FRR in each 3 data set combinations (TD, FD, TD_FD). As we can observe in the above graph the best domains with better values are **Time_Domain** and **Time_Frequency_Domain.** Among that 2 domains we select **Time_Frequency_Domain** as it contains more features , 88 from time domain and 43 from frequency domain and another point is frequency domain data perform well and give low **FAR** rate as shown in the graph.

4.4. Cross-validation

We used the **Leave-One-User-Out (LOUO)** method, which tests the model on all users, including unseen ones, for reliable evaluation. To identify the user left out, **Dynamic Time Warping (DTW)** was used to compare user similarity. DTW measures the distance between two temporal sequences with varying lengths or speeds. DTW is a technique usually used with time-series data, but due to it's features of matching data despite the lengths or speed, we considered using this technique, and before applying DTW, we concatenated the samples inversely making the data suitable for the method.



The user most similar to the target user (based on DTW) was selected as the leave-out for each test. To compare models, evaluations were conducted with and without LOUO, because if we have lots of data, it could help distinguish legitimate and imposter users. Meanwhile we also notice a significant difference in distance of user 10 with other users, it's likely because user 10's data has a huge variance between samples, which we will explore in the further parts of this report.

4.5. Initial parameter settings

The initial model was configured with the following parameters:

- **Dropout Rate (0.3):** Deactivates 30% of neurons during training to reduce overfitting.
- **L2 Regularization (1e-4):** Balances penalizing large weights with retaining flexibility.
- **Performance Goal (1e-5):** Ensures precise model performance with a low error threshold.
- **Minimum Gradient (1e-6):** Prevents training stalls caused by small updates.
- **Training Algorithm (trainscg):** Scaled conjugate gradient for moderate datasets.
- **Early Stopping (10 epochs):** Stops training when validation performance stagnates for 10 epochs.
- **Hidden Layer (131 Neurons):** Matches the number of input features.
    - **Activation Functions:** logsig for the hidden layer to capture non-linear relationships and tansig for the output layer, suitable for cross-entropy loss.
- **Loss Function:** Cross-entropy loss for effective prediction error measurement.
- **Learning Rate (0.01):** Balances convergence speed and stability.
- **Max Epochs (500):** Allows sufficient training while relying on early stopping to avoid overfitting.

4.6. Evaluation Metrics

The following metrics were used to assess model performance:

- Primary Metrics: False Acceptance Rate (FAR), False Rejection Rate (FRR), and Equal Error Rate (EER).
- Secondary Metrics: Accuracy, Precision, Recall, and a similarity score system for each target user's model.

EER, FAR, and FRR were prioritized for their reliability in assessing model efficiency.

Equal Error Rate (EER)

It's the point where FAR equals FRR. EER is used as a single value metric to evaluate the overall performance of a biometric system. Lower EER indicates better performance. The threshold at which $FAR = FRR$ is determined through iterative threshold adjustments.

$$FAR(t) = FRR(t)$$

False Acceptance Rate (FAR)

Measures the proportion of imposter users incorrectly accepted as legitimate. A lower FAR indicates better security against unauthorized users.

$$FAR = \frac{FP}{TN + FP}$$

False Rejection Rate (FRR)

Measures the proportion of legitimate users incorrectly classified as imposters. A lower FRR indicates better usability.

$$FRR = \frac{FN}{TP + FN}$$

Similarity Score

A measure of the mean of the probabilities of each user's samples for each target user's model.

t : Target User

$SS_{(u,\,t)}$: Mean Similarity Score for the evaluated user u in target user t 's Model.

N_u: Number of samples from the evaluated user.

M_t: Neural network model trained for the target user

u_i: A sample from the tested user.

$P(M_t, u_i)$: Probability output by the model M for the i -th sample u_i belonging to the target user t .

$SS_{(u,t)}$: Mean Similarity Score for the evaluated user u in target user t 's Model.

$$SS_{(u,t)} = \frac{1}{N}\sum_{i=1}^{N_u} P(M_t, u_i)$$

Also, a median was also of the similarities with a variation was calculated and plotted on the similarity score plot. The **M** indicates the Median of the similarity scores with its corresponding variance.

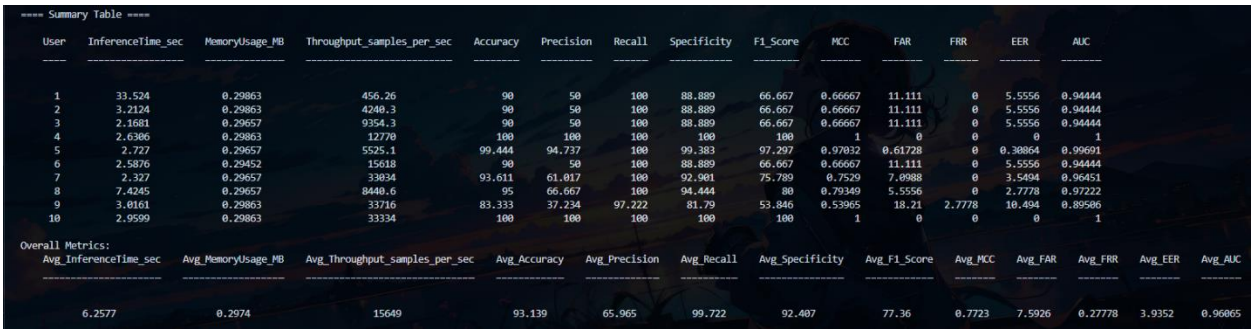
$MS_{(u,t)}$: The middle value of the sorted probabilities $P(M_t, u_i)$ for all i . If N_u is even, the median is the average of the two central values.

$$MS_{(u,t)} = \frac{\min\big(P(M_t, u)\big) + \max\big(P(M_t, u)\big)}{2}$$

5. Evaluation

5.1. Initial model performance (before optimization)

According to the analysis findings we selected 'TimeD_FreqD' combined domain for further use and trained the neural network using 'TimeD_FreqD_FDay' and tested using 'TimeD_FreqD_MDay'. the outputs of evaluation metrics for Initial model (before optimization) are shown below. Average classifier **accuracy** of 93.139%, average **FAR** of 7.5926%, average **FRR** of 0.27778%, average **EER** of 3.9352% are the main evaluation metrics that we can get for initial model. Deep description added as a graph and table.



```
==== Summary Table ====
 User  InferenceTime_sec  MemoryUsage_MB  Throughput_samples_per_sec  Accuracy  Precision  Recall  Specificity  F1_Score  MCC      FAR      FRR  EER      AUC
 ____  _________________  ______________  __________________________  ________  _________  ______  ___________  ________  _______  _______  ___  _______  _______

   1        33.524            0.29863              456.26               90        50       100      88.889      66.667   0.66667   11.111    0    5.5556   0.94444
   2        3.2124            0.29863              4240.3               90        50       100      88.889      66.667   0.66667   11.111    0    5.5556   0.94444
   3        2.1681            0.29657              9354.3               90        50       100      88.889      66.667   0.66667   11.111    0    5.5556   0.94444
   4        2.6306            0.29863              12770                100       100      100      100         100      1         0         0    0        1
   5         2.727            0.29657              5525.1               99.444    94.737   100      99.383      97.297   0.97032   0.61728   0    0.30864  0.99691
   6        2.5876            0.29452              15618                90        50       100      88.889      66.667   0.66667   11.111    0    5.5556   0.94444
   7         2.327            0.29657              33034                93.611    61.017   100      92.901      75.789   0.7529    7.0988    0    3.5494   0.96451
   8        7.4245            0.29657              8440.6               95        66.667   100      94.444      80       0.79349   5.5556    0    2.7778   0.97222
   9        3.0161            0.29863              33716                83.333    37.234   97.222   81.79       53.846   0.53965   18.21     2.7778   10.494   0.89586
  10        2.9599            0.29863              33334                100       100      100      100         100      1         0         0    0        1

Overall Metrics:
 Avg_InferenceTime_sec  Avg_MemoryUsage_MB  Avg_Throughput_samples_per_sec  Avg_Accuracy  Avg_Precision  Avg_Recall  Avg_Specificity  Avg_F1_Score  Avg_MCC  Avg_FAR  Avg_FRR  Avg_EER  Avg_AUC
 _____________________  __________________  ______________________________  ____________  _____________  __________  _______________  ____________  _______  _______  _______  _______  _______

       6.2577                0.2974                  15649                    93.139        65.965      99.722        92.407          77.36       0.7723   7.5926   0.27778  3.9352   0.96065
```

1. Evaluation metrics for initial model

user	Accuracy	AUC	Precision	Recall	FAR	FRR	EER
1	90	0.94444	50	100	11.111	0	5.5556
2	90	0.94444	50	100	11.111	0	5.5556
3	90	0.94444	50	100	11.111	0	5.5556
4	100	1	100	100	0	0	0
5	99.444	0.99691	94.737	100	0.61728	0	0.30864
6	90	0.94444	50	100	11.111	0	5.5556
7	93.611	0.96451	61.017	100	7.0988	0	3.5494
8	95	0.97222	66.667	100	5.5556	0	2.7778
9	83.333	0.89506	37.234	97.222	18.21	2.7778	10.494
10	100	1	100	100	0	0	0

2. Neural network features and parameters

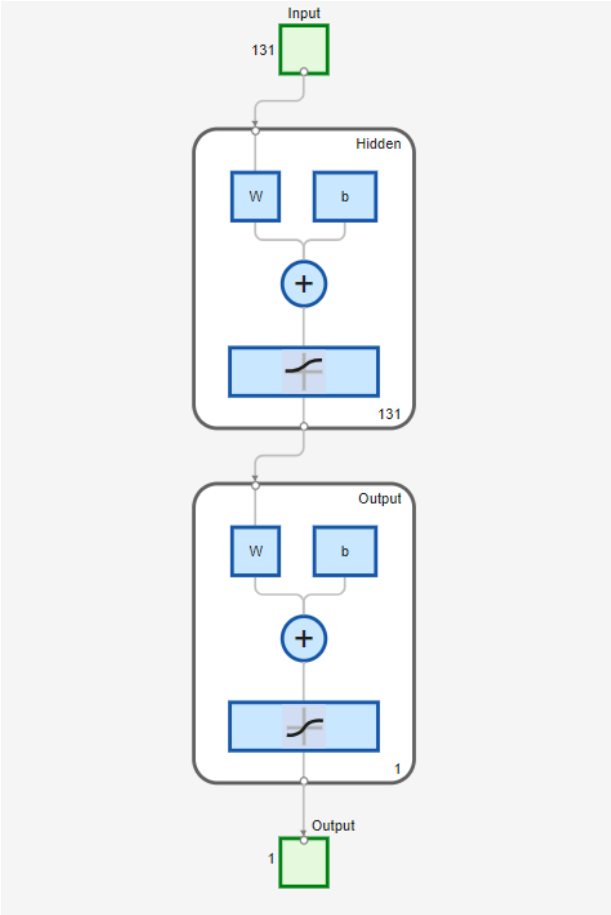
```
==== Neural Network Architecture ====

Input Layer: 131 neurons
Hidden Layer 1: 131 neurons (logsig)
Output Layer:  neuron (tansig)
Training Algorithm: trainscg
Performance Function: Cross-Entropy
L2 Regularization: 1.000000e-04
Max Epochs: 500

==== Performance Benchmarks ====

Average Training Time: 6.2577 seconds (±9.6995)
Average Memory Usage: 0.30 MB (±0.00)
Average Throughput: 15648.88 samples/second (±12930.44)
```

4. Neural network diagram for initial model



Network Diagram

Training Results

Training finished: Met performance criterion

Training Progress

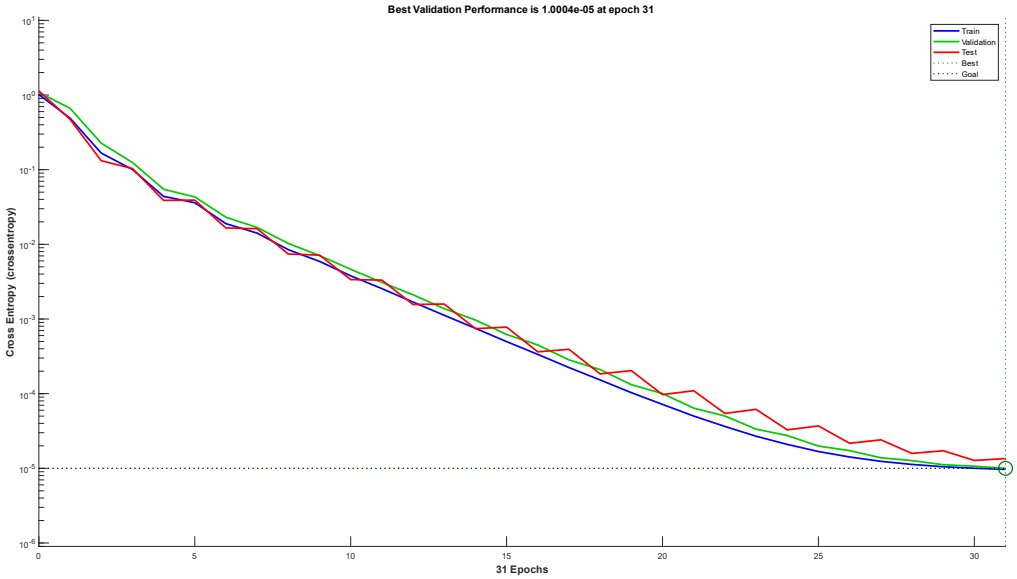
Unit	Initial Value	Stopped Value	Target Value
Epoch	0	31	500
Elapsed Time	-	00:00:01	-
Performance	1.02	9.69e-06	1e-05
Gradient	6.06	3.6e-06	1e-06
Validation Checks	0	0	10

Training Algorithms

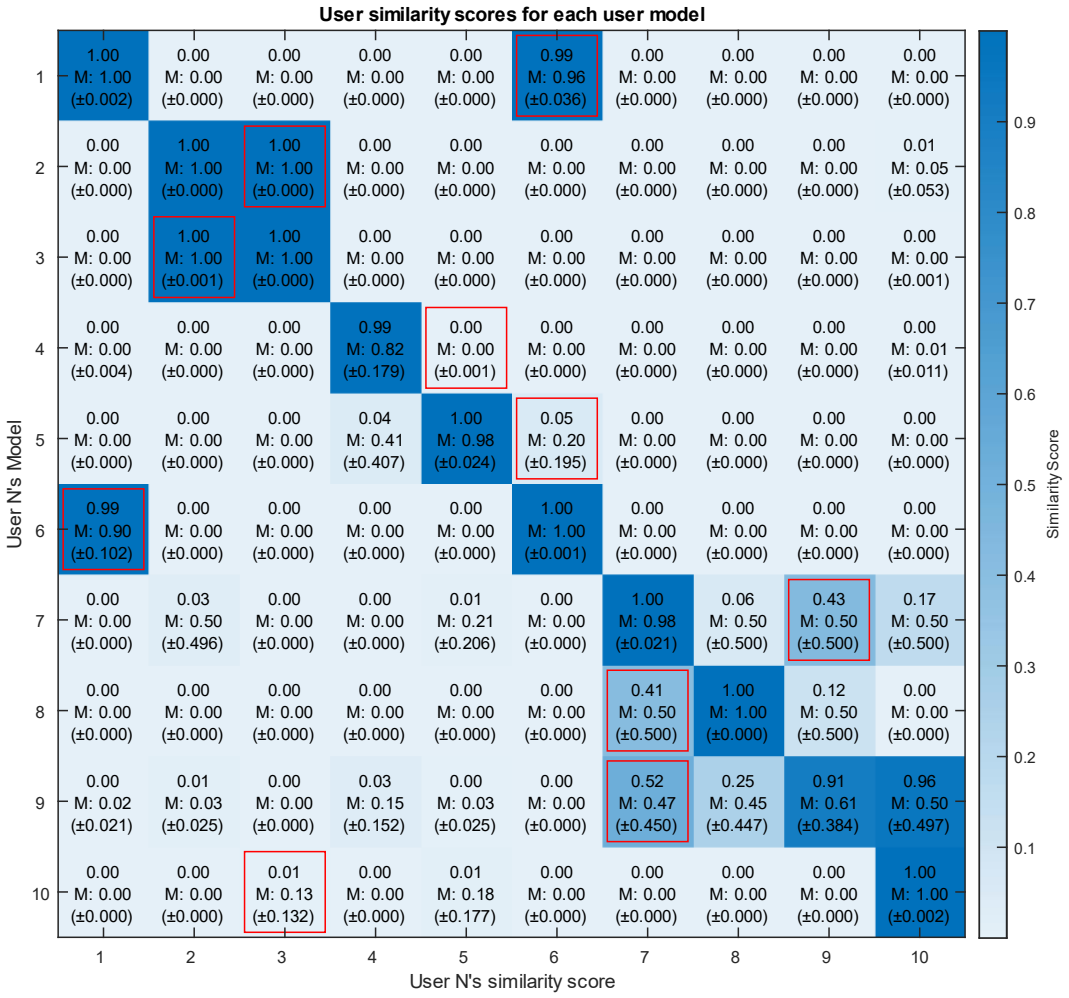
Data Division: Random dividerand
Training: Scaled Conjugate Gradient trainscg
Performance: Cross Entropy crossentropy
Calculations: MEX

Training Plots

Performance | Training State
Error Histogram | Regression

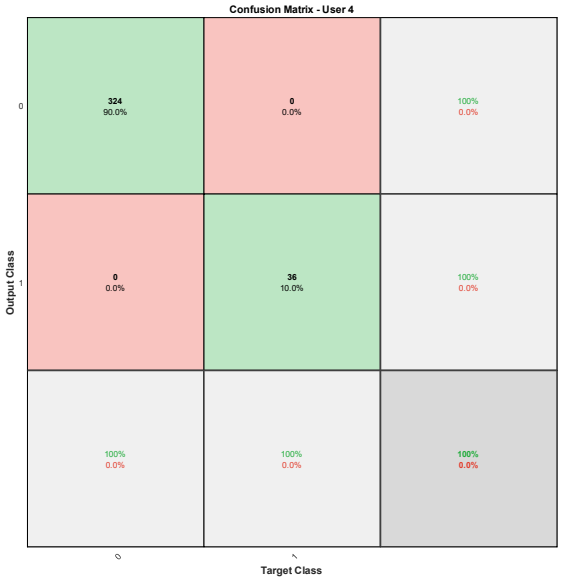


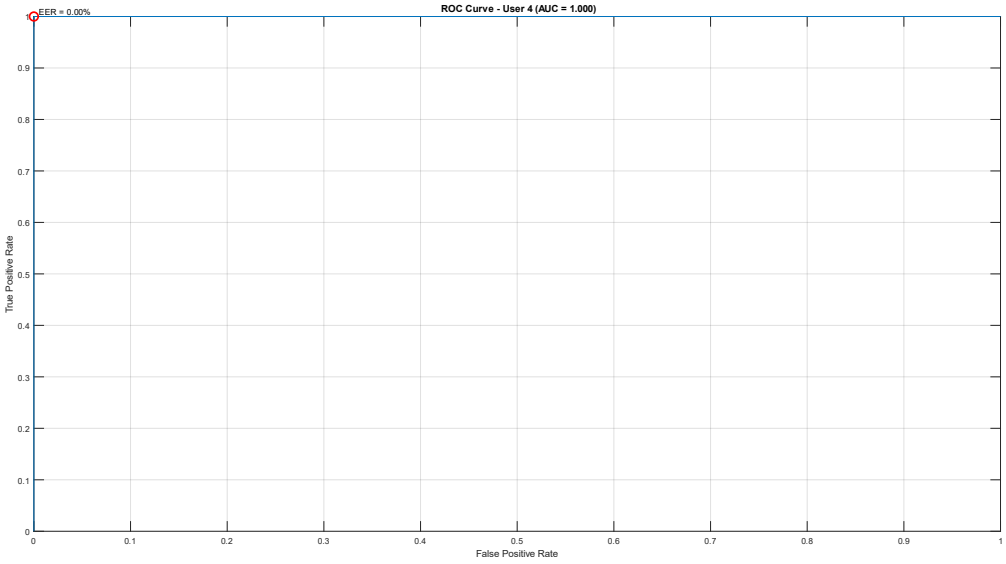
5. Similarity scores for users in initial model (before optimization)

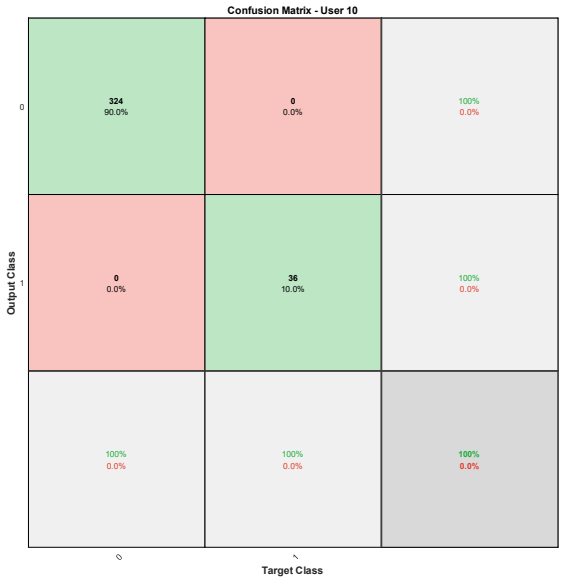


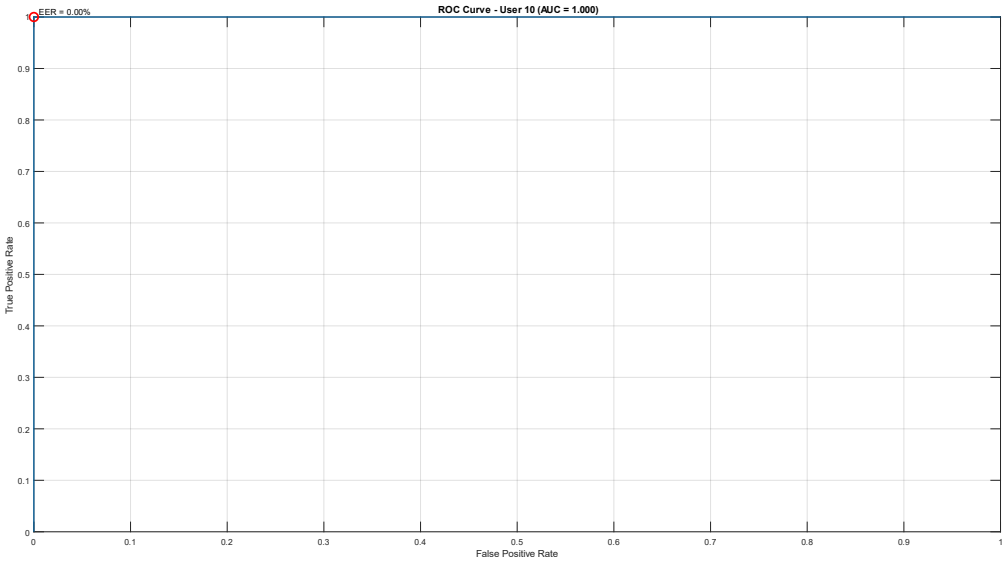
We calculated the similarity scores for every user by testing them each other. All the legitimate users get high similarity scores when they tested against their own testing set. Other imposters get low scores when they tested against legitimate user. As mentioned earlier we integrated **LOUO** (leave one user out) method for randomly remove one user from reference template and train the neural networks, then testing the model with all the 10 users' data. The leave out user marked in the graph using a red color square for each testing iteration. We can see that anomaly when testing **legitimate user 1, 2, 3** with **imposter 6, 3, 2** . The random user we dropped got higher similarity score so in some cases they also accepted as legitimate user. We handled that anomaly in the model optimization with higher accuracy.

6. Best sample outputs among users (confusion matrix and ROC curves with EER value)







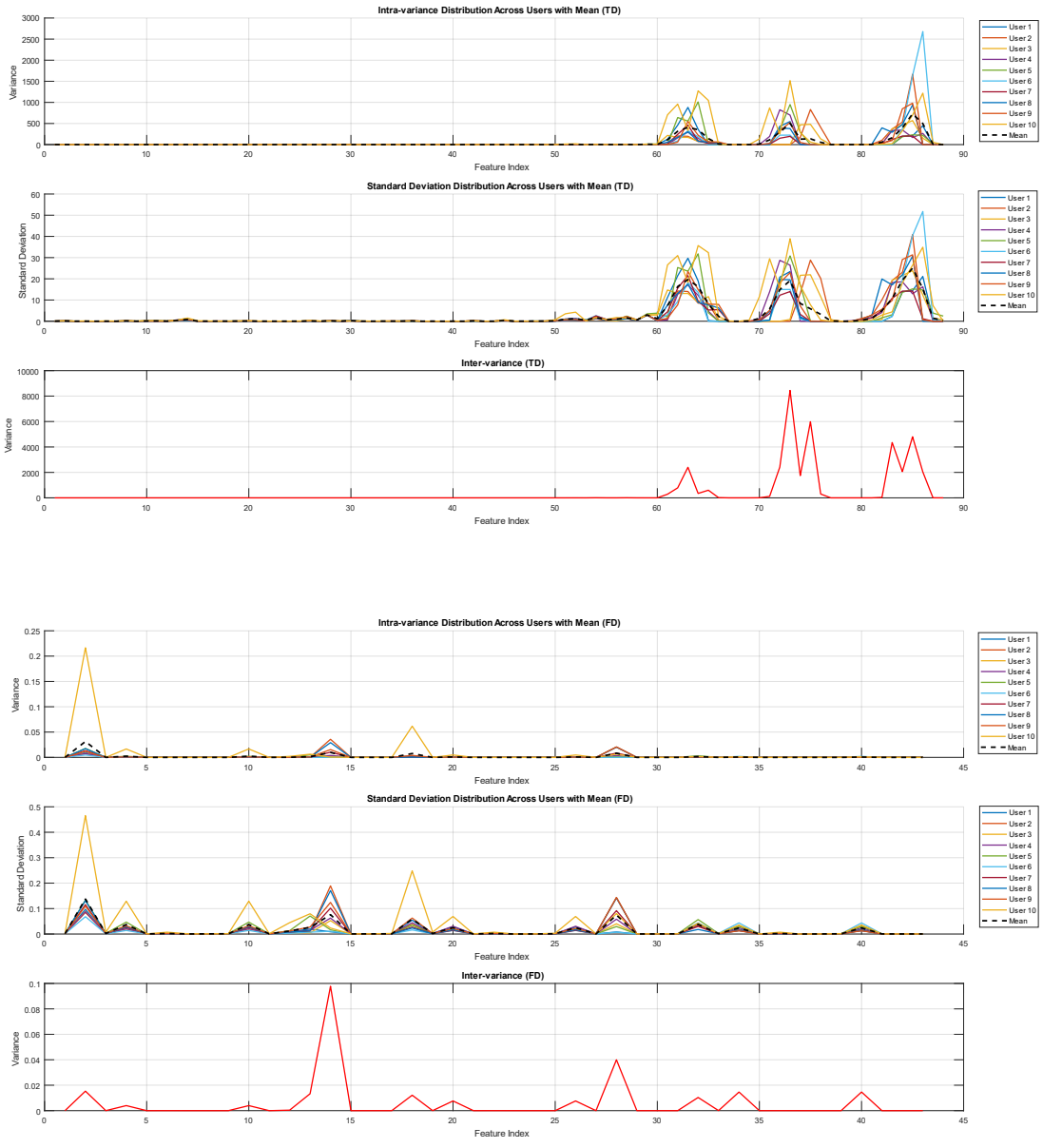


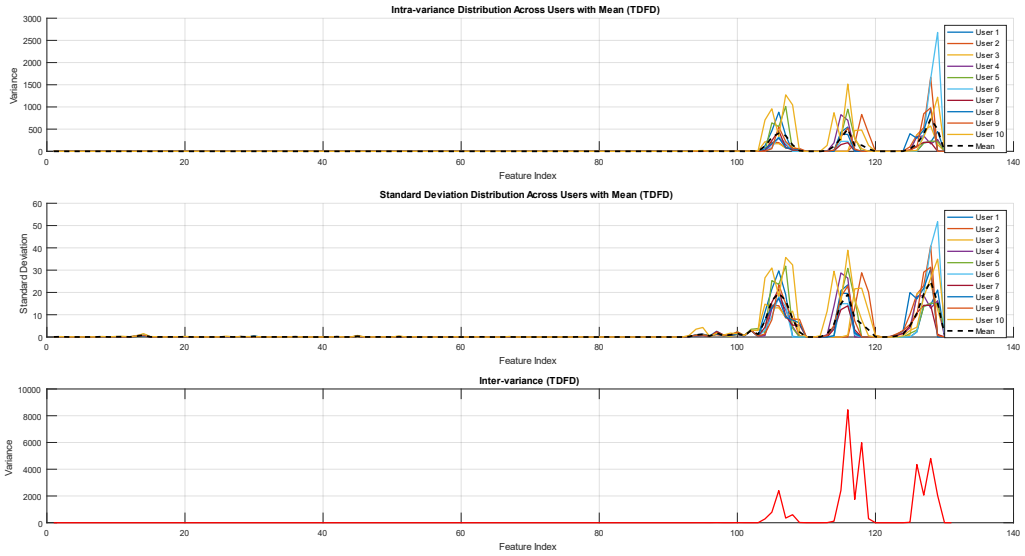
Here we shown the results using confusion matrix and Roc curve for user 4 and user 10 who got the best accuracy rates in initial neural network results.

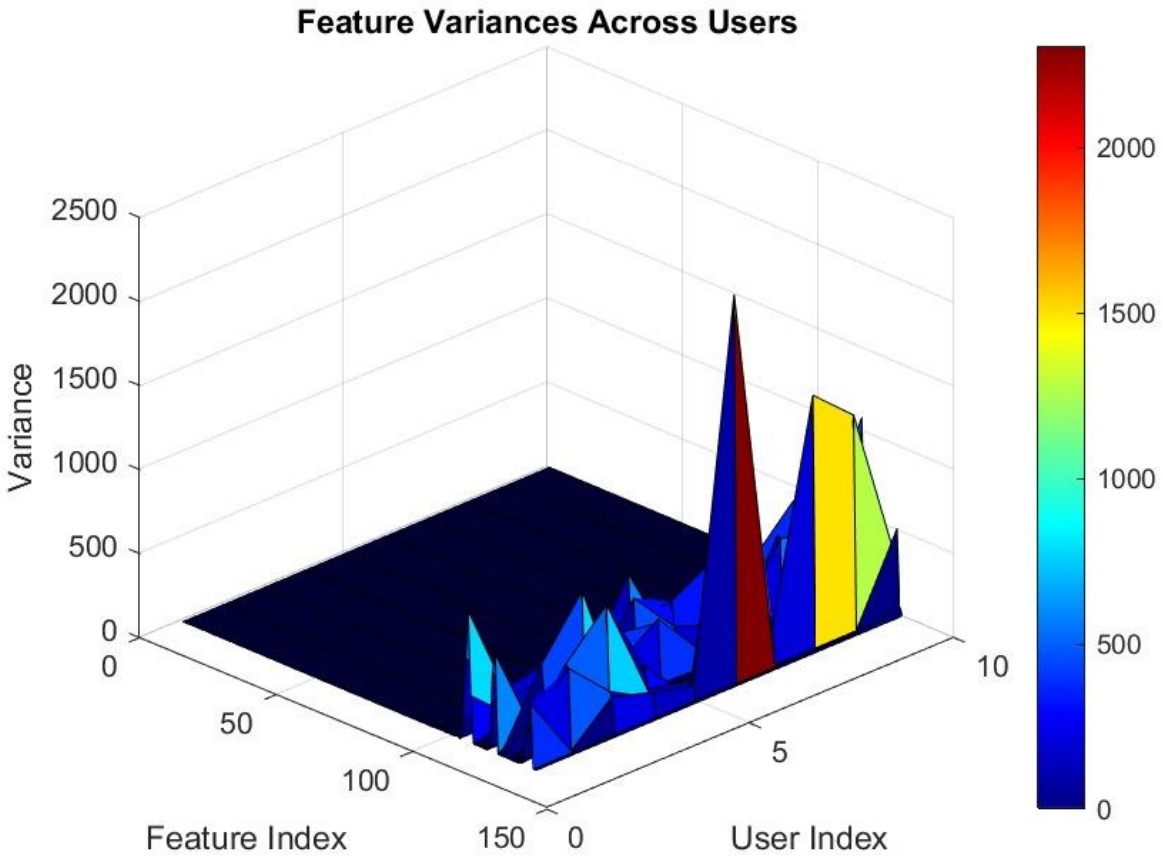
5.2. Intra-Variance and Inter-varience observations and findings

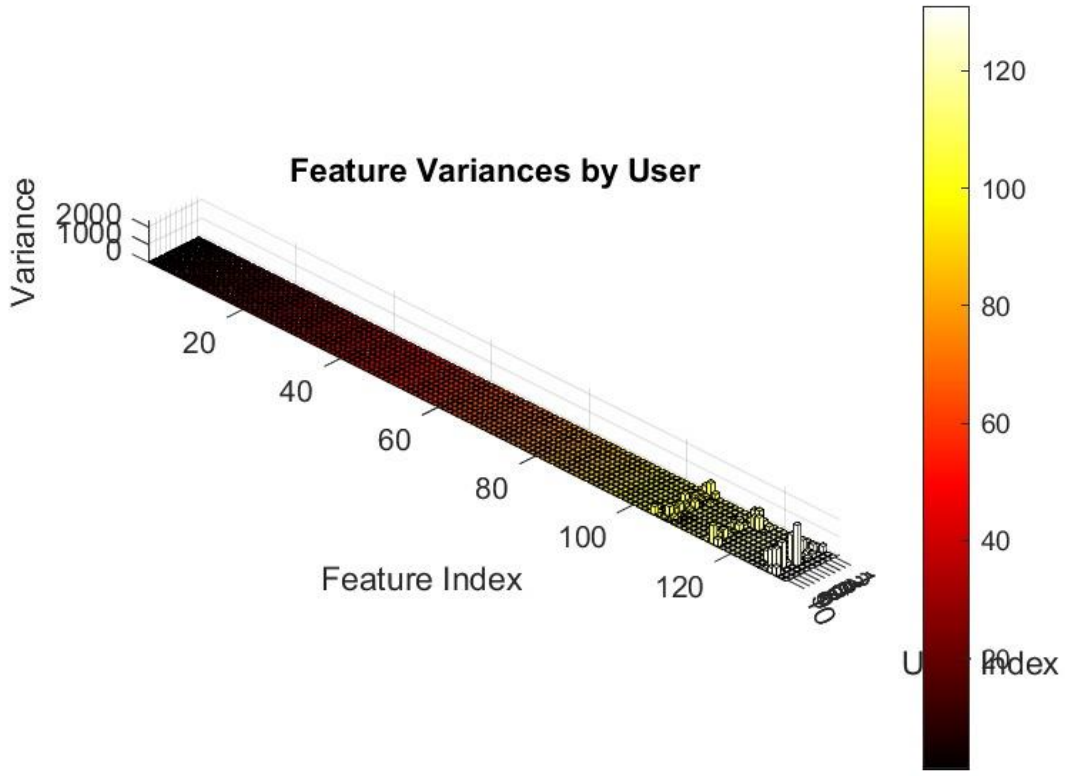
Intra-variance refers to the variability in motion patterns within a single user across multiple samples. Lower intra-variance indicates consistent behavior, making authentication more reliable.

Inter-variance measures the differences in motion patterns between different users. High inter-variance ensures that users have distinct motion characteristics, which is crucial for accurately differentiating between individuals in authentication systems. Both metrics are critical for proper integration of the authentication model.




Intra-variance Distribution Across Users with Mean (TDFD)
Variance
Feature Index
User 1
User 2
User 3
User 4
User 5
User 6
User 7
User 8
User 9
User 10
Mean
Standard Deviation Distribution Across Users with Mean (TDFD)
Standard Deviation
Feature Index
Inter-variance (TDFD)
Variance
Feature Index


Feature Variances Across Users
Variance
Feature Index
User Index



Observations,

TD Variance: Variability is concentrated between **feature indices 60–90**, with sharp peaks indicating significant differences within and between groups. Shows significant variance starting around feature index 60, peaking between 60 and 90

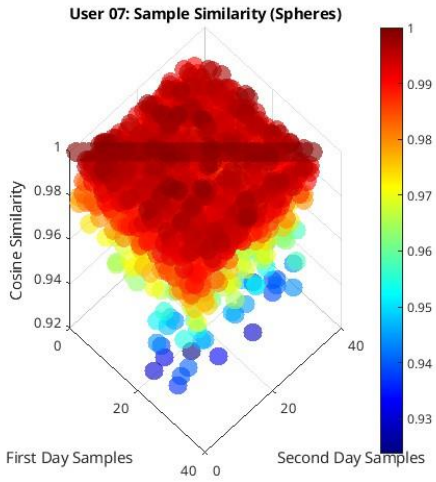
TDFD Variance: Variability is broader, covering **feature indices 100–130**. This suggests more widespread and significant variance across features. Higher Inter-variance compared to TD.

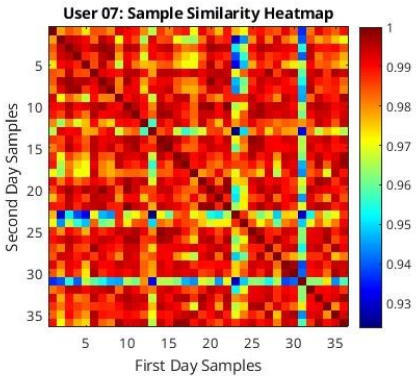
FD Variance: Variability is minimal and localized around **feature indices 10–20**, with lower values overall. Making FD more stable but less diverse. Has lower Inter-variance overall.

During ANOVA analysis (mentioned later), features with low intra-variance and high inter-variance will produce low p-values and high F-statistics. These characteristics are chosen for the authentication system because they have a high ability to distinguish between real users and fraudsters.

5.3. Similarity of samples between the two days for users

1. Anomally finding of user seven (7)

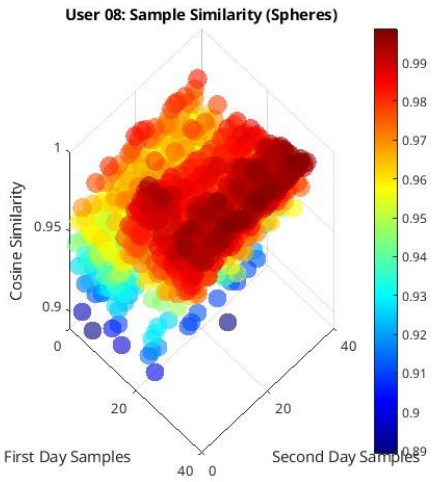


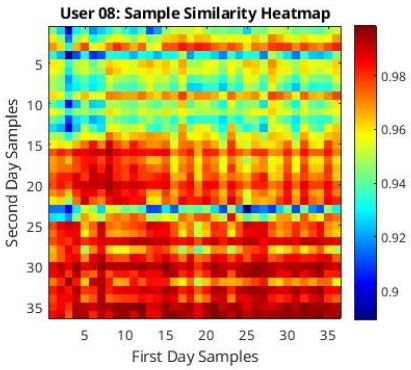


As we have separated the day's data for the training set & testing set, we reviewed the similarities of the samples on the 2 days, to make sure that the walking pattern of the users stay consistent throughout the days. For this, we used cosine similarity for the sample vectors to find the similarity within samples. Cosine similarity is a method of finding similarity of given feature vectors. While inspecting cosine similarities for samples of the two days, we found out that **User 7** has a cosine similarity of 1.0 (which means the samples are same). This could be a intended or unintended data duplication of the user, because it's almost impossible for a user to mimic acceleration data for all the 36 sample.

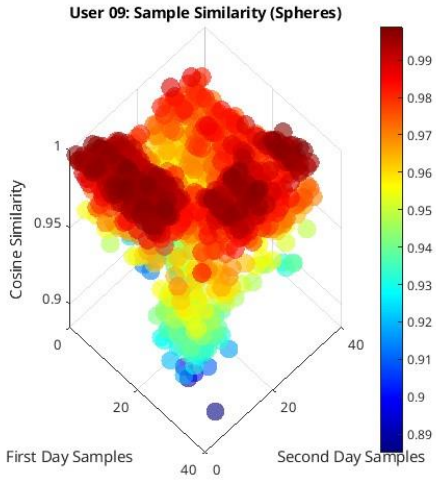
2. For other users

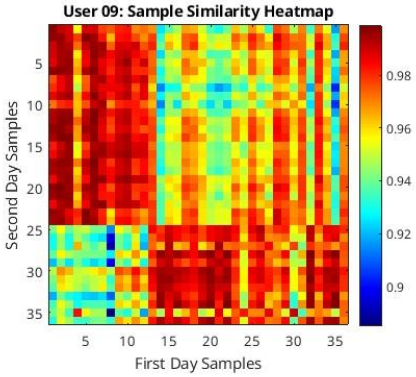
The rest of users did not any anomalies through this analysis. All the users have a fair similarity between their own samples of the and sometimes a bit difference in few samples between the 2 days, which could happen as the user's walking pattern could have slight changes over time, either due to health concerns or any other unexpected scenarios. This shows how users may have the differences between their own samples over time and this triggers a possibility for the model to fail to authenticate. Also, this shows a requirement for a model which can re-train itself for users to adapt with the change of user's walking patterns over time. Unfortunately, we will not be focusing on this fact in this analysis, we will be only focusing on getting a model working for the testing approach we've discussed about.





For User 8

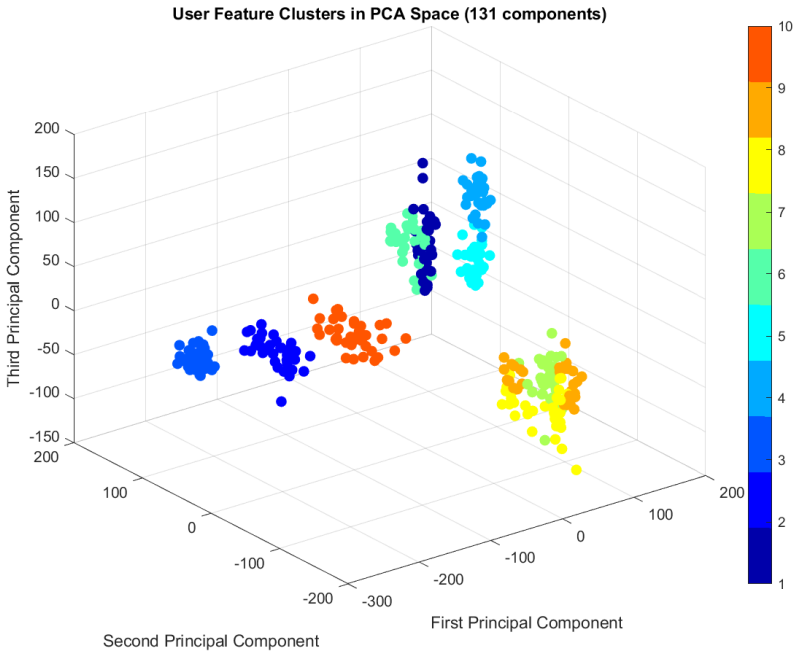


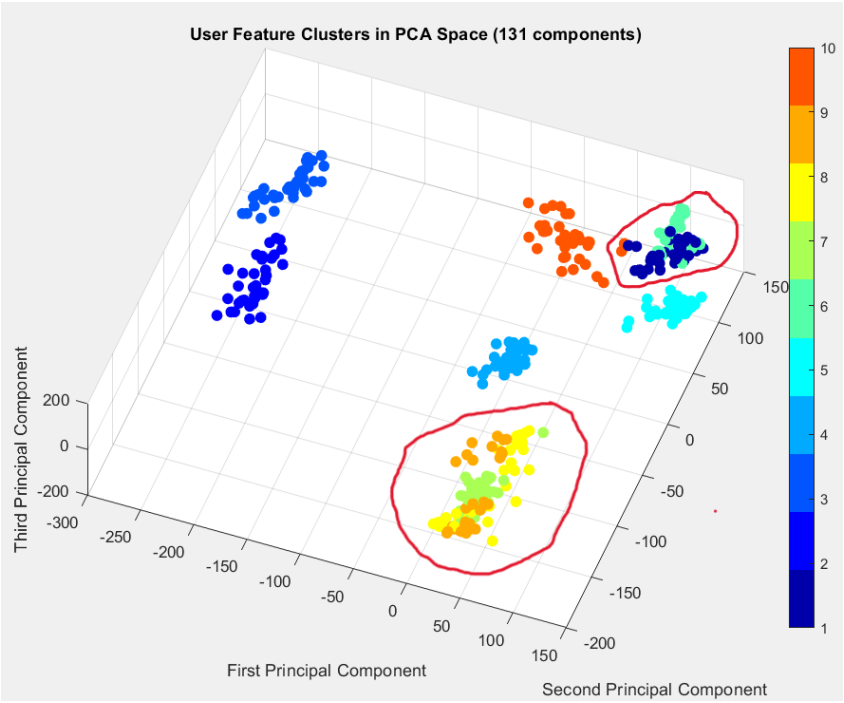


For User 9

5.4. PCA analysis and it's findings

Principal Component Analysis (PCA) is a dimensionality reduction method that is used to reduce the complexity of huge datasets while preserving as much variance as possible. The original features are converted into a new collection of uncorrelated variables known as principal components, which are then arranged according to how much of the variance they can account for. This method can result in enhancing model performance, decreases redundancy, and increases computing efficiency by choosing the best components, particularly in high-dimensional data.





Finding: We really can see that some users can't be separately identified as the real-world perspective. This high intra-cluster similarity likely contributed to the low precision .

6. Optimization

Our optimization efforts were driven by the need to improve classification performance through effective feature selection and neural network adjustments.

6.1. Initial Optimization

We systematically explored various feature selection techniques to identify impactful features as follows:

- **ANOVA**
- **ANOVA + Reliable Ratio + PCA**
- **ANOVA + Mutual Information (MI)**
- **Recursive Feature Elimination (RFE)**

  Trying those methods were affected for some performance metrices but as an average it didn't take many improvements

Finally, we could find that **ANOVA + MI + Steepest Gradient** was the most effective method for the feature selection optimization this phase. As we discovered. But it was also did not impact for our model's precision . When find out why this happened , we could discover that the features, and the Behaviors among the users are slightly similar when considering few users.

The feature selection process combines **ANOVA**, **Mutual Information (MI)**, and **Steepest Gradient** methods to identify the most discriminative features for user authentication. Each method contributes uniquely:

1. **ANOVA**:
   - Evaluates statistical differences between target (genuine) and imposter feature distributions.
   - Features with lower p-values (e.g., <0.05) are considered significant.
   - Improves class separation by focusing on meaningful features.

$$f = \frac{SSw/dfw}{SSb/dfb}$$

2. **Mutual Information (MI)**:
   - Measures the dependency between features and the target variable.
   - Features with higher MI scores provide more relevant information for classification.
   - Formula:

$$MI(X;Y) = \sum_{x \in X}^{n} \sum_{y \in Y}^{n} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)}\right)$$

3. **Steepest Gradient**:
    - Uses neural network gradients to rank features based on their impact on model predictions.
    - Higher gradient values indicate more influential features.
    - Formula:

$$\textbf{Gradients} = \frac{\partial L}{IW \partial L}$$

4. **Hybrid Approach**:

    - Combines normalized scores from all three methods using weighted importance (e.g., 40% ANOVA, 30% MI, 30% Steepest Gradient).
    - Selects the top 75% of features to balance dimensionality reduction and classification accuracy.

This hybrid method improves model performance by reducing redundant features, minimizing overfitting, and enhancing genuine / imposter separation for accurate user authentication with high rates of metrics.

we mentioned below the Performance metrics to each user,

6.2. Optimization results using Hybrid Approach (ANOVA + MI + SG)

Type 1 - Including leave one user out approach

1.1. Performance metrics with (LOUO) for each user

= Summary Table ====

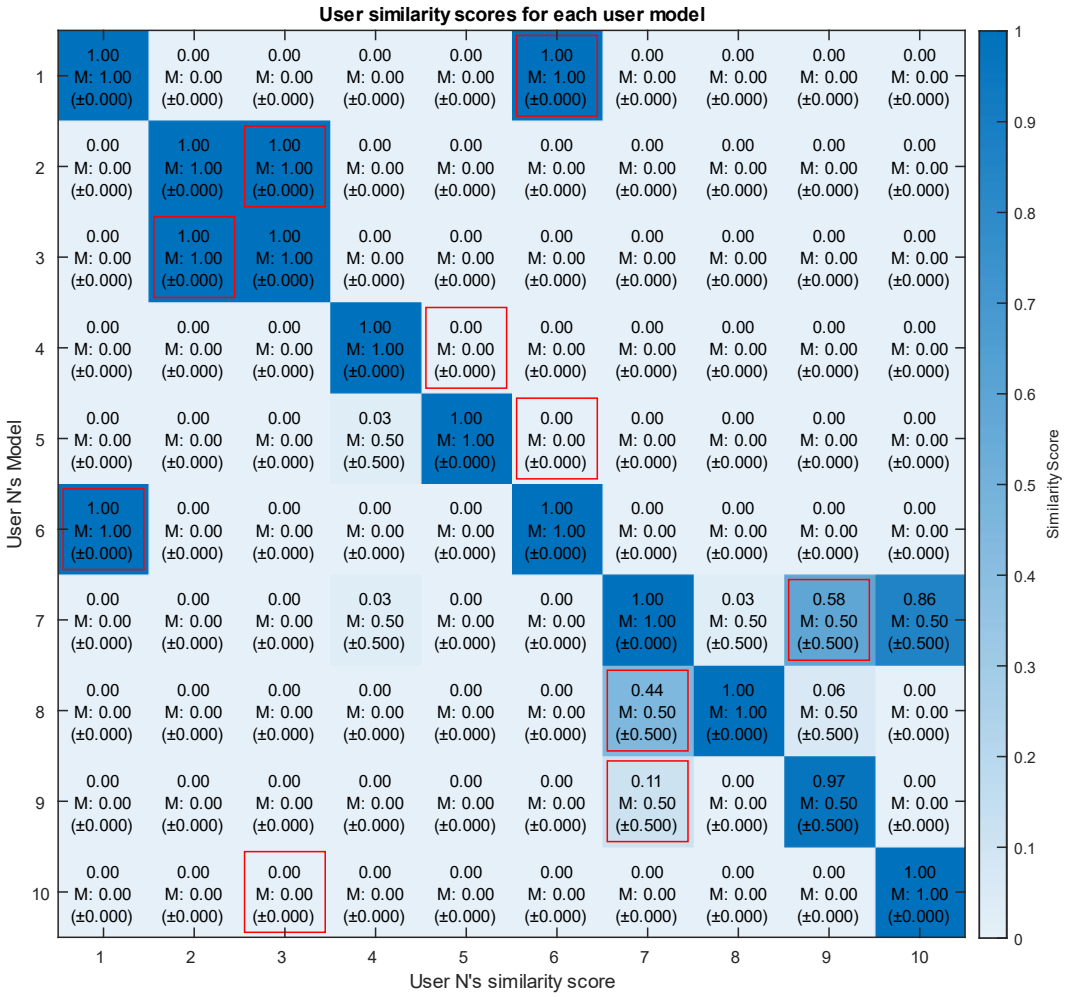
User	InferenceTime_sec	MemoryUsage_MB	Throughput_samples_per_sec	Accuracy	OverallPrecision	Recall	Specificity	F1_Score	MCC	FAR	FRR	EER	AUC
1	3.1237	0.24056	6969	90	50	100	88.889	66.667	0.66667	11.111	0	5.5556	0.94444
2	2.5284	0.24056	18402	90	50	100	88.889	66.667	0.66667	11.111	0	5.5556	0.94444
3	2.4617	0.24056	26862	90	50	100	88.889	66.667	0.66667	11.111	0	5.5556	0.94444
4	4.5948	0.24056	26533	100	100	100	100	100	1	0	0	0	1
5	3.7275	0.24056	12759	99.722	97.297	100	99.691	98.63	0.98487	0.30864	0	0.15432	0.99846
6	28.137	0.24056	1200.9	90	50	100	88.889	66.667	0.66667	11.111	0	5.5556	0.94444
7	66.776	0.24056	20872	85	40	100	83.333	57.143	0.57735	16.667	0	8.3333	0.91667
8	53.009	0.24056	42216	95	66.667	100	94.444	80	0.79349	5.5556	0	2.7778	0.97222
9	30.078	0.24056	55138	98.611	89.744	97.222	98.765	93.333	0.92652	1.2346	2.7778	2.0062	0.97994
10	50.136	0.24056	50084	100	100	100	100	100	1	0	0	0	1

rall Metrics:

Avg_InferenceTime_sec	Avg_MemoryUsage_MB	Avg_Throughput_samples_per_sec	Avg_Accuracy	Avg_OverallPrecision	Avg_Recall	Avg_Specificity	Avg_F1_Score	Avg_MCC	Avg_FAR	Avg_FRR	Avg_EER
24.457	0.24056	26104	93.833	69.371	99.722	93.179	79.577	0.79489	6.821	0.27778	3.5494

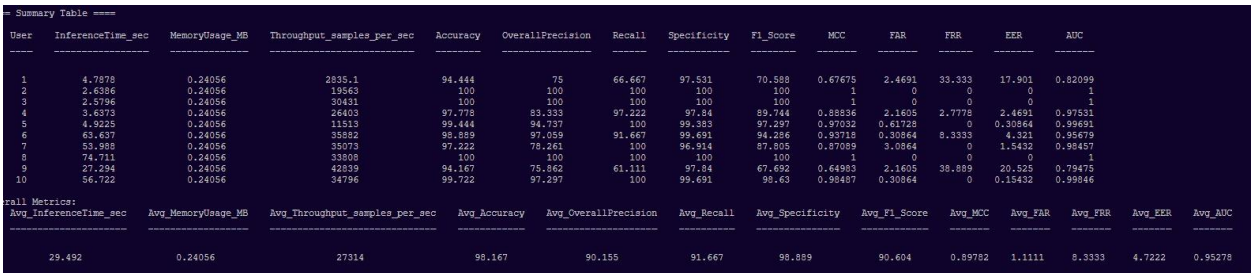
user	Accuracy	AUC	Precision	Recall	FAR	FRR	EER
1	90	0.94444	50	100	11.111	0	5.5556
2	90	0.94444	50	100	11.111	0	5.5556
3	90	0.94444	50	100	11.111	0	5.5556
4	100	1	100	100	0	0	0
5	99.722	0.99846	97.297	100	0.30864	0	0.15432
6	90	0.94444	50	100	11.111	0	5.5556
7	85	0.91667	40	100	16.667	0	8.3333
8	95	0.97222	66.667	100	5.5556	0	2.7778
9	98.611	0.97994	89.744	97.222	1.2346	2.7778	2.0062
10	100	1	100	100	0	0	0

1.2. Similarity scores for users with (LOUO)



Type 2 - Excluding leave one user out approach

2.1. Performance metrics without LOUO for each user



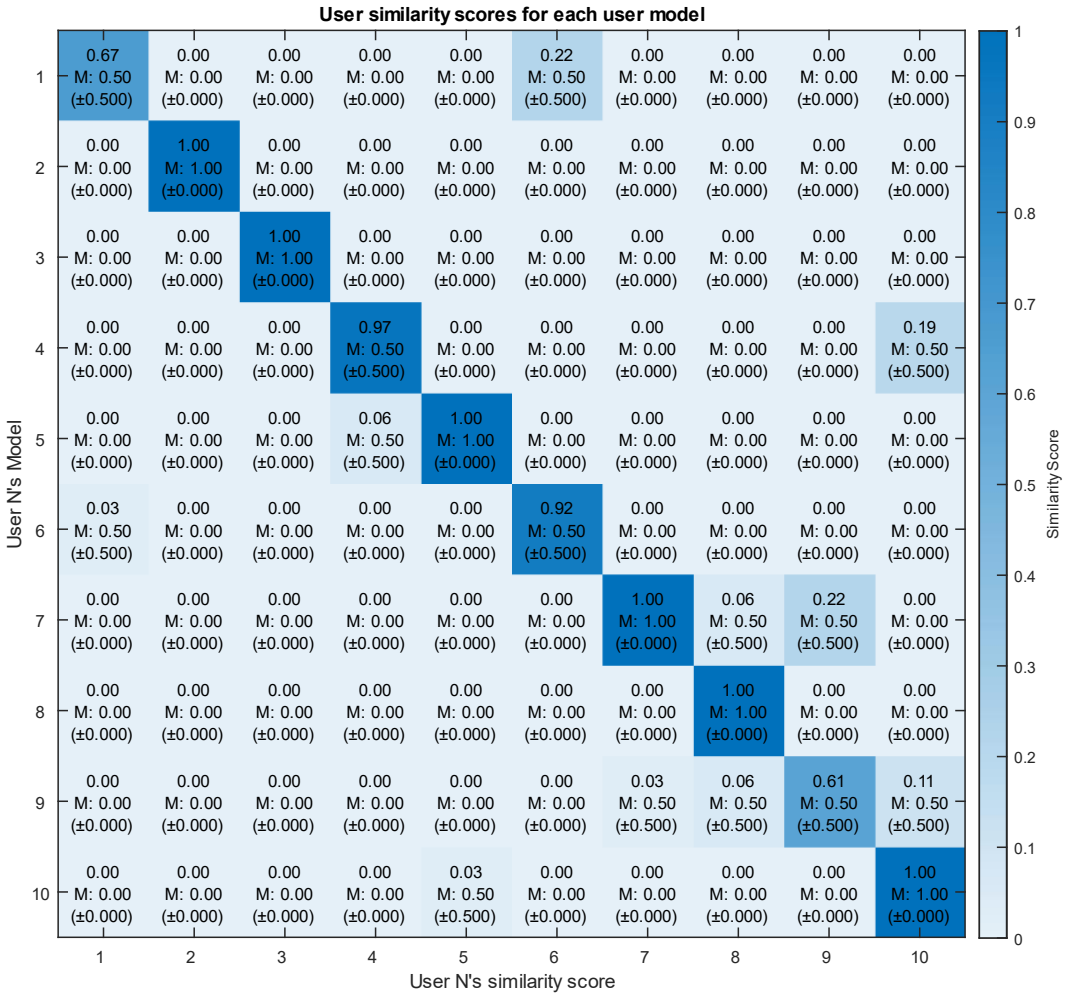
```
= Summary Table ====
 User   InferenceTime_sec   MemoryUsage_MB   Throughput_samples_per_sec   Accuracy   OverallPrecision   Recall   Specificity   F1_Score   MCC       FAR      FRR      EER      AUC
 ----   -----------------   --------------   --------------------------   --------   ----------------   ------   -----------   --------   -------   ------   ------   ------   -------
    1             4.7878          0.24056                       2835.1     94.444                 75   66.667        97.531     70.588   0.67675   2.4691   33.333   17.901   0.82099
    2             2.6386          0.24056                        19563        100                100      100           100        100         1        0        0        0         1
    3             2.5796          0.24056                        30431        100                100      100           100        100         1        0        0        0         1
    4             3.6373          0.24056                        26403     97.778             83.333   97.222         97.84     89.744   0.88836   2.1605   2.7778   2.4691   0.97531
    5             4.9225          0.24056                        11513     99.444             94.737      100        99.383     97.297   0.97032   0.61728        0   0.30864   0.99691
    6             63.637          0.24056                        35882     98.889             97.059   91.667        99.691     94.286   0.93718   0.30864   8.3333    4.321   0.95679
    7             53.988          0.24056                        35073     97.222             78.261      100        96.914     87.805   0.87089    3.0864        0   1.5432   0.98457
    8             74.711          0.24056                        33808        100                100      100           100        100         1        0        0        0         1
    9             27.294          0.24056                        42839     94.167             75.862   61.111         97.84     67.692   0.64983   2.1605   38.889   20.525   0.79475
   10             56.722          0.24056                        34796     99.722             97.297      100        99.691      98.63   0.98487   0.30864        0   0.15432   0.99846

rall Metrics:
 Avg_InferenceTime_sec   Avg_MemoryUsage_MB   Avg_Throughput_samples_per_sec   Avg_Accuracy   Avg_OverallPrecision   Avg_Recall   Avg_Specificity   Avg_F1_Score   Avg_MCC   Avg_FAR   Avg_FRR   Avg_EER   Avg_AUC
 ---------------------   ------------------   ------------------------------   ------------   --------------------   ----------   ---------------   ------------   -------   -------   -------   -------   -------

               29.492              0.24056                            27314         98.167                 90.155       91.667            98.889         90.604   0.89782    1.1111    8.3333    4.7222   0.95278
```

user	Accuracy	AUC	Precision	Recall	FAR	FRR	EER
1	90.444	0.82098	75	86.687	2.4891	33.333	17.931
2	100	1	100	100	0	0	0
3	100	1	100	100	0	0	0
4	97.778	0.97531	83.333	97.222	2.1805	2.7378	2.4691
5	99.444	0.99891	94.737	100	0.81725	0	0.30864
6	98.888	0.98879	97.058	91.687	0.30864	8.3333	4.321
7	97.222	0.98457	78.261	100	3.0864	0	1.5432
8	100	1	100	100	0	0	0
9	94.167	0.97475	75.862	81.111	2.1805	38.889	20.525
10	99.722	0.99546	97.297	100	0.30864	0	0.15432

2.2. Similarity scores for users without (LOUO)



6.3. Advanced Optimization Using GA and SVM

1.Genetic Algorithm Using Support Vector Machines

This is a method inspired by natural evolution that optimizes solutions by mimicking the process of natural selection. It treats features as "chromosomes" like that structure the optimum features of like in a genetic form

In this case, the quality of each “chromosome” is measured using the accuracy of a Support Vector Machine (SVM). SVM is a supervised learning model that finds the best boundary to separate data. The GA generates different feature sets, SVM identify the best one to performed. So, after running the model we could gain this information.

```
Training model for User 1...
+++++ ============================================================================ +++++


Single objective optimization:
131 Variables

Options:
CreationFcn:       @gacreationuniform
CrossoverFcn:      @crossoverscattered
SelectionFcn:      @selectionstochunif
MutationFcn:       @mutationadaptfeasible


                                Best           Mean      Stall

Generation      Func-count       f(x)           f(x)    Generations
    1              100         0.01389        0.02483        0
    2              150         0.01111        0.02422        0
    3              200         0.01389        0.02406        1
    4              250         0.01389        0.02317        2
    5              300         0.01389        0.02333        0
    6              350         0.01111        0.02444        0
    7              400         0.01111        0.02372        1
    8              450         0.01111        0.02278        2
    9              500         0.01111        0.02233        0
   10              550         0.01111        0.02233        1
   11              600        0.008333        0.02167        0
   12              650         0.01389        0.02406        1
   13              700         0.01389        0.02256        0
   14              750         0.01111        0.02217        0
   15              800         0.01389         0.0225        1
   16              850         0.01111        0.02222        0
   17              900         0.01389        0.02233        1
   18              950         0.01111        0.02222        0
   19             1000         0.01111        0.02167        1
   20             1050         0.01111        0.02133        2
```

GA Options:

Function selections on GA & SVM and their intro,

- **CreationFcn: @gacreationuniform** - Creates the initial population uniformly.
- **CrossoverFcn: @crossoverscattered** - Performs scattered crossover to combine feature subsets.
- **SelectionFcn: @selectionstochunif** - Uses stochastic uniform selection to choose the best subsets.
- **MutationFcn: @mutationadaptfeasible** - Applies adaptive feasible mutation to introduce variability.

Observations,

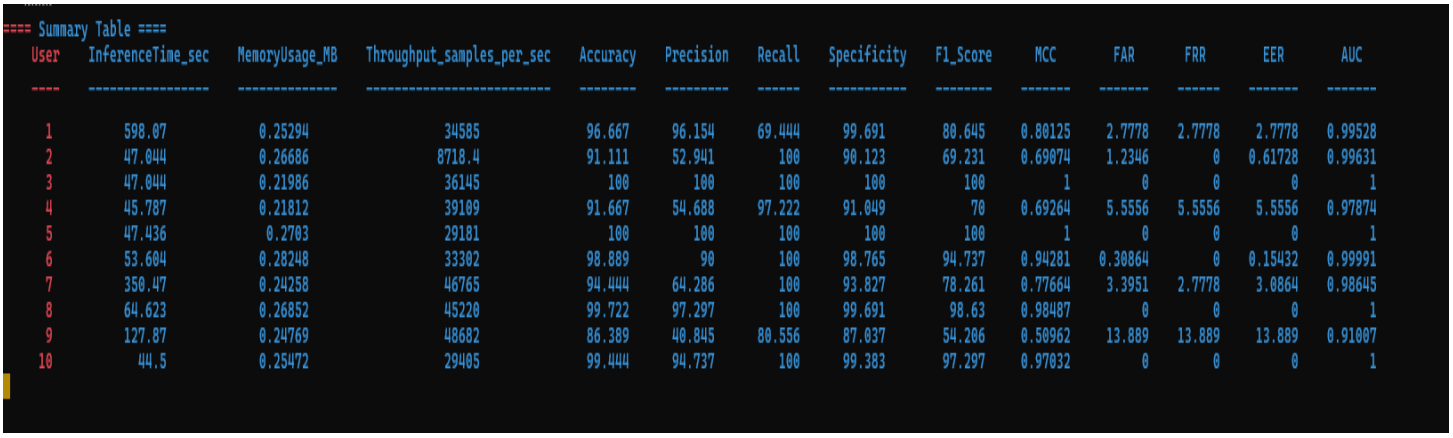
- Best f(x) starts at 0.01389 and improves to 0.008333 by generation 11.
- Mean f(x) decreases over generations, indicating overall population improvement.
- Stall shows generations without improvement in the best value.
- Generations indicates the number of generations since the last improvement.

After Using this Methods for the Feature Selection , this is the Summary of users' Performance matrices, For Both (With LOUO and without LOUO)

6.4. Optimization results Using GA and SVM approach

Type 1 - Including leave one user out approach

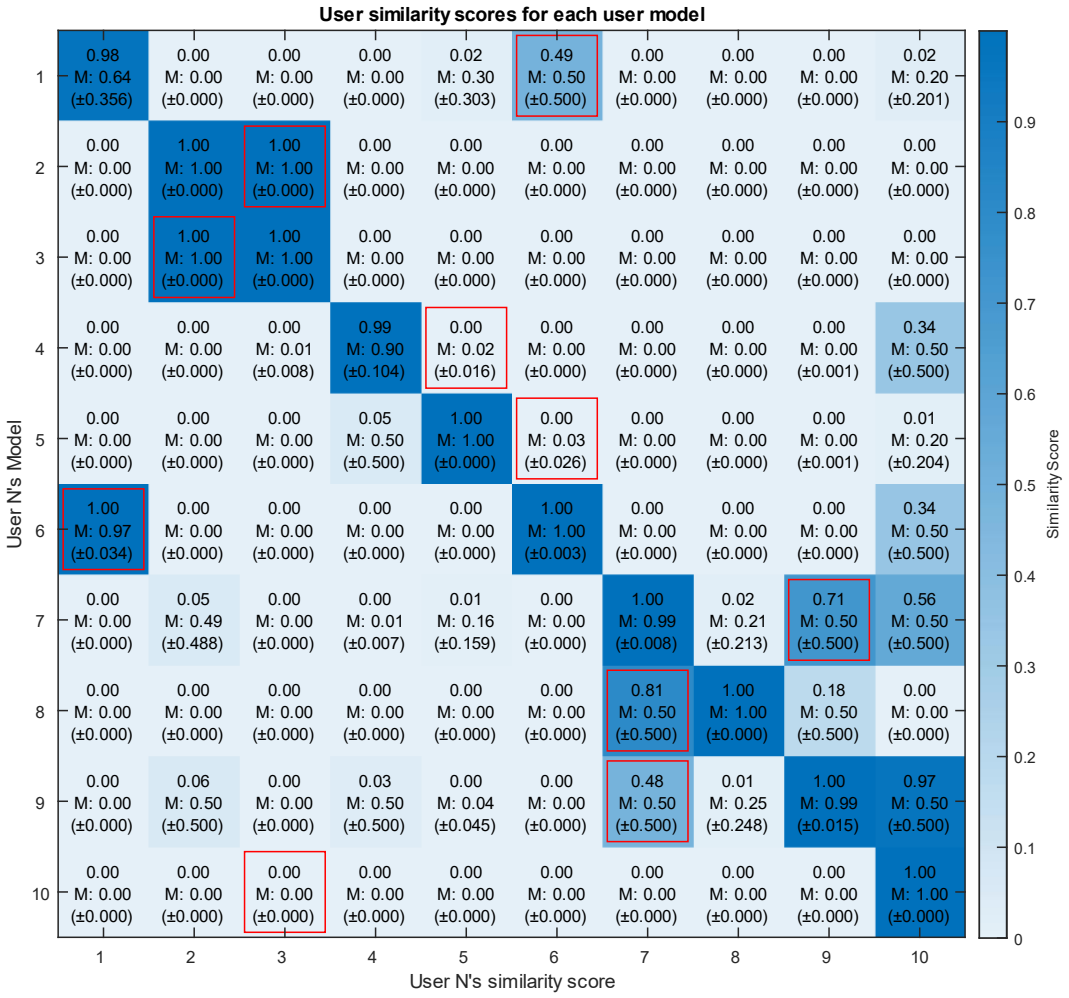
1.1. Results With LOUO for each user



```
==== Summary Table ====
 User  InferenceTime_sec  MemoryUsage_MB  Throughput_samples_per_sec  Accuracy  Precision  Recall  Specificity  F1_Score  MCC      FAR     FRR     EER     AUC
 ----  -----------------  --------------  --------------------------  --------  ---------  ------  -----------  --------  -------  ------  ------  ------  -------
    1             598.07         0.25294                       34585    96.667     96.154  69.444       99.691    80.645  0.80125  2.7778  2.7778  2.7778  0.99528
    2             47.044         0.26686                      8718.4    91.111     52.941     100       90.123    69.231  0.69074  1.2346       0  0.61728  0.99631
    3             47.044         0.21986                       36145       100        100     100          100       100        1       0       0       0        1
    4             45.787         0.21812                       39109    91.667     54.688  97.222       91.049        70  0.69264  5.5556  5.5556  5.5556  0.97874
    5             47.436          0.2703                       29181       100        100     100          100       100        1       0       0       0        1
    6             53.604         0.28248                       33302    98.889         90     100       98.765    94.737  0.94281  0.30864       0  0.15432  0.99991
    7             350.47         0.24258                       46765    94.444     64.286     100       93.827    78.261  0.77664   3.3951  2.7778   3.0864  0.98645
    8             64.623         0.26852                       45220    99.722     97.297     100       99.691     98.63  0.98487        0       0       0        1
    9             127.87         0.24769                       48682    86.389     40.845  80.556       87.037    54.206  0.50962   13.889  13.889  13.889  0.91007
   10               44.5         0.25472                       29405    99.444     94.737     100       99.383    97.297  0.97032        0       0       0        1
```

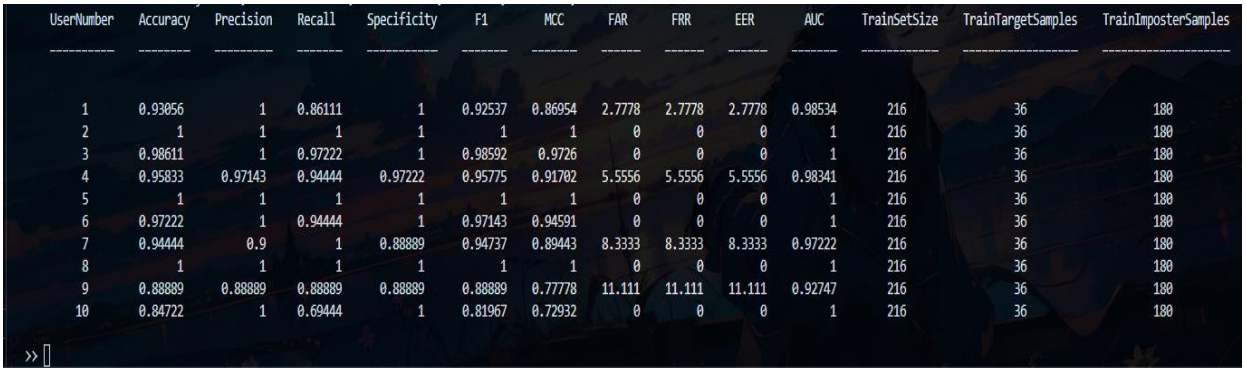
user	Accuracy	AUC	Precision	Recall	FAR	FRR	EER
1	96.667	0.99528	96.154	69.444	2.7778	2.7778	2.7778
2	91.111	0.99631	52.941	100	1.2346	0	0.61728
3	100	1	100	100	0	0	0
4	91.667	0.97874	54.688	97.222	5.5556	5.5556	5.5556
5	100	1	100	100	0	0	0
6	98.889	0.99991	90	100	0.38864	0	0.15432
7	94.444	0.98645	64.286	100	3.3951	2.7778	3.8864
8	99.722	1	97.2297	100	0	0	0
9	86.389	0.91887	49.845	88.556	13.889	13.889	13.889
10	99.444	1	94.737	100	0	0	0

1.2. Similarity scores for users with (LOUO)



Type 2 - Excluding leave one user out approach

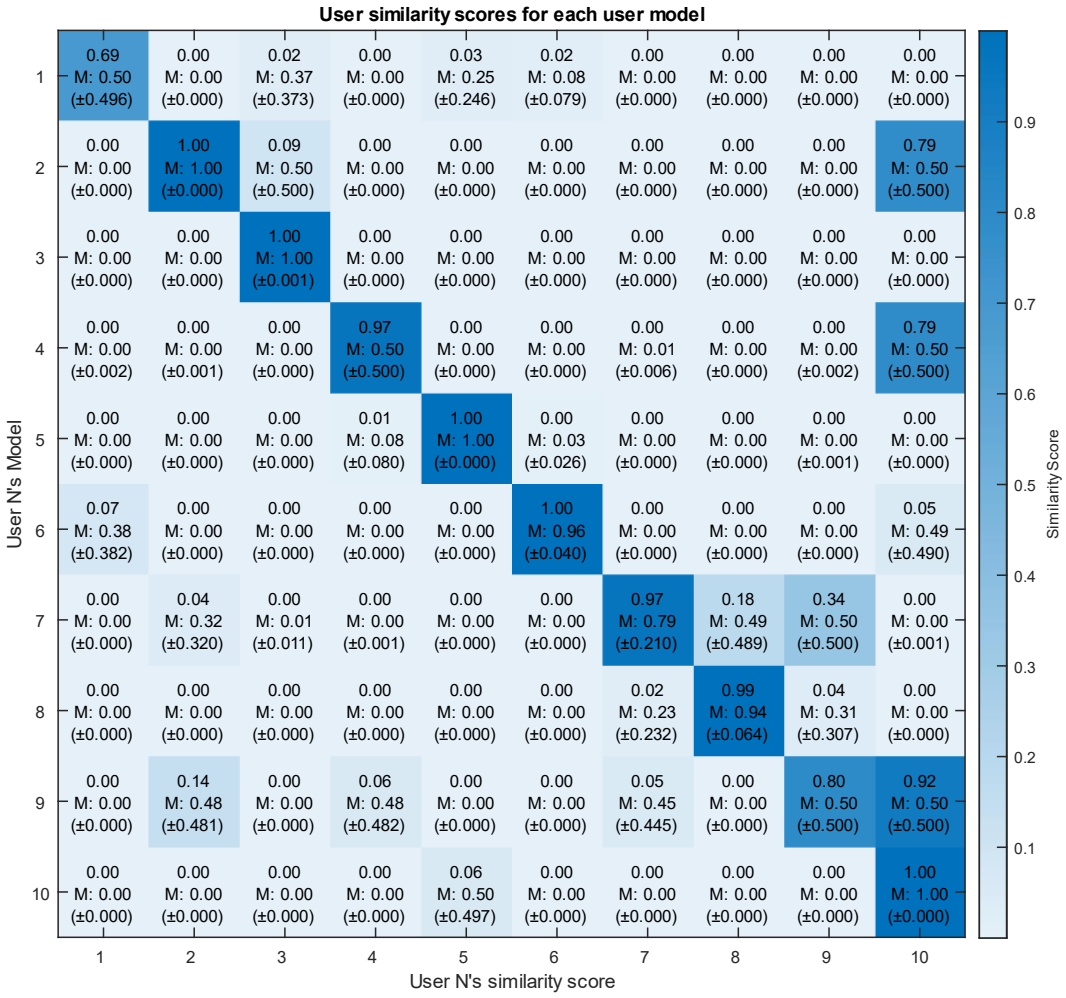
2. Results without LOUO for each user



UserNumber	Accuracy	Precision	Recall	Specificity	F1	MCC	FAR	FRR	EER	AUC	TrainSetSize	TrainTargetSamples	TrainImposterSamples
1	0.93056	1	0.86111	1	0.92537	0.86954	2.7778	2.7778	2.7778	0.98534	216	36	180
2	1	1	1	1	1	1	0	0	0	1	216	36	180
3	0.98611	1	0.97222	1	0.98592	0.9726	0	0	0	1	216	36	180
4	0.95833	0.97143	0.94444	0.97222	0.95775	0.91702	5.5556	5.5556	5.5556	0.98341	216	36	180
5	1	1	1	1	1	1	0	0	0	1	216	36	180
6	0.97222	1	0.94444	1	0.97143	0.94591	0	0	0	1	216	36	180
7	0.94444	0.9	1	0.88889	0.94737	0.89443	8.3333	8.3333	8.3333	0.97222	216	36	180
8	1	1	1	1	1	1	0	0	0	1	216	36	180
9	0.88889	0.88889	0.88889	0.88889	0.88889	0.77778	11.111	11.111	11.111	0.92747	216	36	180
10	0.84722	1	0.69444	1	0.81967	0.72932	0	0	0	1	216	36	180

user	Accuracy	AUC	Precision	Recall	FAR	FRR	EER
1	93.056	0.98534	100	86.111	2.7778	2.7778	2.7778
2	100	1	100	100	0	0	0
3	98.611	1	100	97.222	0	0	0
4	95.833	0.98341	97.143	94.444	5.5556	5.5556	5.5556
5	100	1	100	100	0	0	0
6	97.222	1	100	94.444	0	0	0
7	94.444	0.97222	90	100	8.3333	8.3333	8.3333
8	100	1	100	100	0	0	0
9	88.889	0.92747	88.889	88.889	11.111	11.111	11.111
10	84.722	1	100	69.444	0	0	0

2. Similarity scores for users without (LOUO)



Findings: This advanced combination successfully addressed user similarity issues, leading to noticeable improvements in feature selection and classification.

6.5. Comparison Table for Initial VS Optimized models

Model	Accuracy	precision	recall	specifiy	FAR	FRR	EER
Initial Model	93.306	64.302	95.833	93.025	6.975	4.1667	5.571
ANOVA+MI+SGM (without LOUO)	98.167	90.155	91.667	98.889	1.111	8.333	4.7222
ANOVA+MI+SGM (with LOUO)	93.833	69.371	99.722	93.179	6.821	0.277	3.5494
GA+SVM (Without LOUO)	95.833	79.095	94.722	90.864	2.716	2.5	2.608
GA+SVM (With LOUO)	91.694	61.366	99.167	90.864	5.277	5.246	0.97743

1. Initial Model

- **Why Metrics Are Moderate**:
  - No advanced feature selection or optimization.
  - The neural network depends on raw data and default parameters, which can be affected to performance.
  - Overfitting in the features could cause lower precision and higher error rates.

2. ANOVA+MI+SGM (Without LOUO)

- **Why Metrics Increased**:
  - **ANOVA** (Analysis of Variance) and **Mutual Information (MI)** selected the most important features, removing redundant or irrelevant ones.
  - **SGM (Smoothing)** helped reduce noise in the features, improving accuracy and model stability.
  - **Without LOUO**: Testing was done without strict leave-one-user-out validation, leading to **higher accuracy** but slightly reduced generalizability.
- **Impact**:
  - Precision and accuracy increased significantly, but recall dropped slightly due to overfitting on NO-LOUO validation.

3. ANOVA+MI+SGM (With LOUO)

- **Why Metrics Changed**:
  - Applying LOUO validation tested to see really model's ability to generalize to unseen users.
  - LOUO is stricter and more realistic because the model trains without seeing one user's data.
  - This increased **recall** (99.722%) so the model became more sensitive to identifying actual positives but slightly lowered accuracy because of the challenge for generalizing to unseen data.
- **Impact**:
  - Lower **EER** (3.5494) indicates better balance between FAR and FRR.
  - Higher recall shows the model effectively identified genuine users.

4. GA+SVM (Without LOUO)

- **Why Metrics Improved**:
  - **GA (Genetic Algorithm)** optimized model parameters, selecting the best combination of hyperparameters for improved performance.
  - **SVM (Support Vector Machine)** provided strong generalization for classification.
  - Without LOUO, the model accessed to all users' data during training and leading to very high precision (**79.095%**) and recall.
- **Impact**:
  - Lower **FAR (2.716)** and **FRR (2.5)** indicate strong performance on the training data.
  - Overfitting could explain the lower generalization seen without LOUO.

5. GA+SVM (With LOUO)

- **Why Metrics Dropped Slightly**:
  - LOUO validation made the model generalize to unseen users, increasing **FRR (5.246)** and lowering precision (**61.366%**).
  - GA and SVM had to adapt to more challenging validation, and optimization struggled to perfectly generalize for all users.
- **Impact**:
  - Accuracy dropped to **91.694%**, but recall remained high (**99.167%**).
  - **EER (0.9774)** is the lowest across all models, indicating an excellent trade-off between FAR and FRR.

Key Observations,

1. **LOUO Validation**:
    a. Results with LOUO show slightly lower accuracy but **better generalization** to new users.
    b. This highlights the importance of strict validation for realistic deployment.
2. **Feature Selection (ANOVA+MI)**:
    a. Selecting relevant features improved accuracy and reduced noise, boosting overall performance.
3. **Optimization (GA+SVM)**:
    a. Hyperparameter tuning using GA and combining SVM improved both precision and recall, particularly without LOUO.
4. **EER and Trade-offs**:
    a. Models with the lowest EER (e.g., GA+SVM with LOUO) achieved the best balance between False Acceptance (FAR) and False Rejection (FRR).

Conclusion For optimized models,

- The **best performing models** used feature selection, optimization, and stricter validation (LOUO).
- LOUO models had lower accuracy but generalized better, making them more suitable for real-world use.
- Optimization techniques like GA+SVM significantly enhanced precision, recall, and error trade-offs compared to the baseline model.

6.6. Neural Network Tuning

We adjusted the neural network to maximize performance by:

- Modifying layer configurations to improve learning efficiency.
- Changing learning rates with new methods such as random search and activation functions to enhance stability and accuracy.

Achievements: While tuning improved the model performance, some users' highly similar acceleration patterns continued to pose challenges, preventing a breakthrough in test statistics.

After trying to optimize our application . We failed many times and could not understand why . To find out something ,did similarity score graph and also Created a 3D cluster plot based on **PCA** to inspect that. And found some valuable insights that are already added in the evaluation section.

Using Thumb Rule to Determine Hidden Layer Size

1. **Absence of Feature-Specific Guidance**
    a. With the provided challenge, we lacked feature names to prioritize features and ientify the behavoirs and actual features of user.
    b. The thumb rule, which suggests setting the hidden layer size as an average of input and output sizes, provided a practical starting point for architecture design.

2. **Heuristic-Based Approach**
    a. The thumb rule

    $$HIdden\ Layer\ Size = \overline{Input\ Size + Output\ Size} / 2$$

    is widely accepted for high-dimensional data when no prior knowledge is available.
    b. For our dataset with **131 input features** and **1 output**, the initial hidden layer size was calculated as: 131 , 66.
    c. This allowed us to establish a baseline configuration efficiently.

3. **Empirical Validation of the Rule**
    a. So, the thumb rule provides a good starting point, it must be validated through experimentation.
    b. We systematically adjusted the hidden layer size (e.g. 131 / 66 neurons) to identify the optimal architecture, using validation accuracy and recall as performance metrics.

4. **Balancing Model Complexity**
    a. The thumb rule balances model capacity:

5. **Justified Simplicity**
    a. With the output size being **1**, the problem was a **binary classification task**.
    b. The thumb rule supported a simple architecture, so it was easy to score without overcomplicating the network.

6.7. Neural Network Configuration Comparison

Comparison of Initial & Tuned Parameters are given below.

Initial Parameters:

- Architecture: Single hidden layer with 131 neurons.
- Activation Functions: logsig for hidden layer and tansig for output.
- Training: Learning rate of 0.01, 500 epochs, L2 regularization (1e-4).

Tuned Parameters:

- Architecture: Two hidden layers with 131 and 65 neurons.
- Activation Functions: logsig for hidden layers, satlins for output layer.
- Training: Mini-batch size of 32, reduced epochs to 300, added cross-validation, and random search for hyperparameter optimization.

2. Impact of Tuning:

We have used the tuned neural network in our optimization models to enhance representational power, speed up convergence, and improve generalization, effectively reducing overfitting while boosting model performance.

1. **Improved Representational Power**
    a. Two hidden layers enabled better learning of complex patterns.
    b. Second layer (66 neurons) compressed features, focusing on essential patterns.
2. **Better Generalization**
    a. Cross-validation reduced overfitting issue and ensured performance on unseen data.
    b. L2 regularization penalized large weights, enhancing model robustness.
3. **Faster Convergence**
    a. Mini-batch training (batch size 32) and sped up learning with stable gradient updates.
    b. Reduced epochs (300) saved training time without sacrificing accuracy.
4. **Higher Performance**
    a. Random search for hyperparameter optimization improved accuracy and recall.
    b. Optimized architecture captured patterns more effectively.
5. **Overfitting Mitigation**
    a. Regularization, cross-validation, and tuned epochs prevented memorization of training data.
6. **Efficiency in Learning**
    a. Balanced architecture ensured efficient learning with fewer parameters.
    b. Activation functions (**`logsig`** and **`satlins`**) suited the task for smooth training.

7. Conclusion

In this project, we initially employed a feedforward neural network with a single hidden layer for user authentication using acceleration data. This simple architecture served as a solid foundation, allowing us to build and evaluate the model effectively.

Through experimentation, we determined the optimal training and testing split ratio for our dataset, which helped improve model performance and generalization. We found that the most effective split ratio 1:5, ensuring that the model was both well-trained and sufficiently evaluated on unseen data.

For optimization, we applied various feature selection techniques such as ANOVA, RFE, Relational Ratio, and Mutual Information (MI). Additionally, advanced methods like Genetic Algorithms (GA) and Support Vector Machines (SVM) were used to further enhance feature relevance and reduce overfitting. These optimization strategies significantly boosted the model's performance.

By combining the feedforward network with effective feature selection and training strategies, we successfully addressed the user authentication problem, achieving strong accuracy and robustness in the final model.

8. References

Al-Naffakh, N., Clarke, N., Dowland, P. & Li, F., 2017. *IEEE explore.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/7856695
[Accessed 12 2024].

Cheng, D. et al., 2024. *IEEE explore.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/10733706
[Accessed 12 2024].

Hu, M., Zhang, K., You, R. & Tu, B., 2022. *IEEE explore.* [Online]
Available at: https://ieeexplore.ieee.org/document/9937042
[Accessed 12 2024].

Lin, C. et al., 2022. *IEEE explore.* [Online]
Available at: https://ieeexplore.ieee.org/abstract/document/9918594
[Accessed 12 2024].

Sakorn Mekruksavanich, A. J., 2021. *MDPI.* [Online]
Available at: https://www.mdpi.com/1424-8220/21/22/7519
[Accessed 12 2024].

Wei Song, H. J. et al., 2022. *IEEE exolore.* [Online]
Available at: https://ieeexplore.ieee.org/document/9996544?denied=
[Accessed 2024].

9. Appendix

All the MATLAB scripts and graphs are available in the below GitHub repository.

Git hub link - https://github.com/scssandanayake/AI-ML-Coursework