IN PARTNERSHIP WITH PLYMOUTH UNIVERSITY

Group No: **116**	
Student Reference Number: **10898939, 10898887, 10898727, 10898918, 10900339**	

Module Code: **PUSL3123**	Module Name: **AI and Machine Learning**
Coursework Title: **Assessment in AI and Machine Learning**	
Deadline Date: **Sunday, 15 December 2024, 12:00 PM**	Member of staff responsible for coursework: **Dr. Neamah Al Naffak**
Programme: **BSc (Hons) in Computer Science, BSc (Hons) in Data Science**	

Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.

Name (As appeared on DLE)	Plymouth ID No
Telge Wasanthadeva	**10898939**
Ranasinghe M Ranasinghe	**10898887**
Sayakkara G Abeywickrama	**10898727**
Siriwardana Siriwardana	**10898918**
Joshua Jebapragashan	**10900339**

We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.

Signed on behalf of the group:

Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***
Signed:

Overall mark _____% **Assessors Initials ______** **Date_________**

Table Of Content

List of Figures

1. Introduction

1.1. Overview of the Problem

In the modern digital era, users rely heavily on devices such as smartphones, smartwatches, and other wearables to access sensitive data, including financial information, personal communications, and healthcare records. Traditional authentication methods, such as passwords and PINs, are often insufficient for continuous protection against unauthorized access. To enhance security, continuous and transparent user authentication based on behavioral biometrics, such as acceleration data, has emerged as a promising approach. Acceleration-based features offer a unique way to verify user identity during device usage by analyzing motion patterns, thereby improving security without user intervention.

1.2. Purpose of the Report

Acceleration-based user authentication is the concern of this report in which feedforward multi-layer perceptron (FFMLP) neural networks are going to be used as supervised learning algorithms. For the current report, the main objectives include but are not limited to the following: Understanding the contained dataset in the current project and Outcome Assessment of the performance of different neural network models that are required for classification enhancement by utilizing feature selection and model parameters fine-tuning. Through understanding these approaches, this report aims at establishing the possibility and efficiency of using acceleration-based attributes in improving user authentication. The findings of this study will help to extend the knowledge of continuous authentication approaches and their uses to protect devices.

1.3. Structure of the Report

This report is divided into the following section for the purpose of easy understanding of the content of this report. Literature review of acceleration-based authentication methods, together with an overview of the latest developments, is provided in the Background section. The Testing Methodology discusses the dataset, the settings of the neural network, and the training. In the Evaluation, we show and discuss the findings, as well as predictive model outcomes and confusion matrix results. The Optimization section covers feature selection and parameter tuning and their relationship to enhance the application’s results. This research is rounded off by a synthesis of the findings and a consideration of the possibilities in the different context. The Appendix at the end of the paper encompasses all the MATLAB codes used during the analysis.

2. Literature Review

2.1. Overview of Acceleration-Based Authentication

Acceleration-based authentication uses motion data gathered from the accelerometers usually built into smartphones and smartwatches in the authentication process. People interact with these devices differently and carrying and moving these devices results in specific motions that can sustain the authentication process continuously (Li et al., 2022; Hu and Geng, 2023). These acceleration patterns are recorded in the time domain and the frequency domain and can be used to derive user-specific models. Unlike traditional forms of authentication that include characters and fingerprints, for instance, acceleration-based approaches afford constant and successive verification. This approach helps with security by identifying such irregularities in real-time while only authorized personnel have access through the device's usage span (Ismagilova and Dmitrievich Lushnikov, 2023).

2.2. Recent Methods and Classifiers

Some of the recently proposed methods or classifiers for this use of accelerations are given here. Some of the most widely known are machine learning methods, and these include SVM, Decision tree, KNN and neural network.

Multi-Layer Perceptron artificial neural networks such as Feedforward (FFMLP) have demonstrated a lot of promise because they can construct complicated patterns and associations in high dimensional data (Zhumazhanova and Tatarinov, 2021). Such networks are usually composed of an input layer, one or more hidden layers and an output layer. Because non-linear data is common with training the FFMLP models, the Levenberg-Marquardt algorithm is often applied to train the model.

CNN analysis has been also used for analyzing sequential acceleration data while RNN has also been used in analyzing sequential data. CNNs can deserialize spatial features and RNNs such as LSTM can model temporal dependencies inherent in motion data (Moga and Filip, 2021).

Merging feature selection approaches like PCA with other machine learning classification algorithms has increased the performance of authentication. It lowers down the dimension, the model becomes effective, and it also helps to overcome over-fitting leading to better and safer authentication (Vijay, Devika and Priyangha, 2022).

2.3. System Accuracy and Challenges

The robustness and accuracy of acceleration-based authentication system are a function of data quality, features used, and the classifiers employed. Research to date has indicated that, in each of the configuration categories, FFMLP and LSTM can deliver significantly high levels of accuracy that are typically above 90% (Herath et al., 2022). For instance, utilizing both time-domain and frequency-domain usually increases classification effectiveness because transient and steady-state motions are described.

Maintaining reliable accuracy across multiple users, let alone in different conditions, is still an unsolved problem. User variability arising from the physical alterations in the way an individual grasps or manipulates a device also affects the model (Rukkanchanunt, Petchkuliinda and Ochiai, 2024). Another reason for misclassifications is inter-user similarity, which means that the users acting rather like one another in terms of motion patterns may be misclassified.

Another problem is that real-time authentication may be computationally expensive on devices that have limited resources. Deep neural networks are computationally intensive and bringing them into mobile devices isn't an easy feat. Also, the system needs to be accurate while avoiding interferential factors such as making users' authentication processes not disruptive to device usage (Sateesh Kumar and Purushottama, 2023).

This is because other extrinsic factors like the walk speed of the user, the surface that the device is being used on, and orientation, can also pose a real challenge. Solving these issues entails robust feature selection, model fine-tuning effective that can well address the numerous real-life conditions.

3. Testing Methodology

3.1. Dataset and Feature Description

The dataset used in this study consists of acceleration-based features collected from 10 users, with each user having six different feature sets. These sets include a combination of time-domain and frequency-domain features, capturing both same-day and cross-day data variations. Specifically, the dataset comprises 88 time-domain features and 43 frequency-domain features, totaling 131 features. Each feature set is stored in .mat files, named to reflect the user ID, feature type, and collection context (e.g., U01_Acc_FreqD_FDay.mat).

The time-domain features capture the raw acceleration in the time domain, using statistics such as, mean, standard deviation and skewness. Frequency-domain features are generated from the Fourier Transform of acceleration signals and reflect the signal energy distribution across the frequencies. These features assist in identifying specific characteristics of motion which are peculiar to each user. The dataset contains enough samples to train and test the neural network model developed in the study.

3.2. Neural Network Model Configuration

A Feedforward Multi-Layer Perceptron (FFMLP) neural network is employed for user authentication. FFMLP is well-suited for this task because it can model non-linear relationships between acceleration features and user identity. The network consists of an input layer, a hidden layer, and an output layer. The input layer has 131 neurons, corresponding to the 131 features in the dataset.

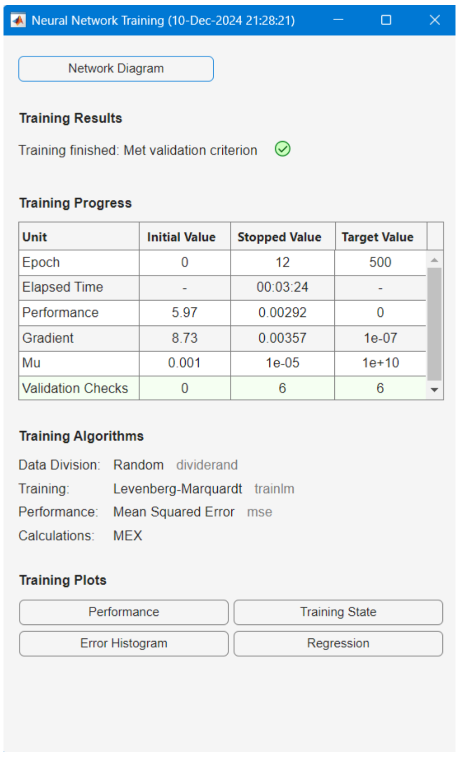
The hidden layer contains 100 neurons, which provides a balance between learning capacity and computational efficiency. A hyperbolic tangent sigmoid activation function (tansig) is used in the hidden layer to introduce non-linearity. The output layer uses a softmax activation function to produce a probability distribution over the 10 user classes.

The network is trained using the Levenberg-Marquardt backpropagation algorithm (trainlm), which is known for its speed and efficiency in solving non-linear optimization problems. The performance function used is Mean Squared Error (MSE). To prevent overfitting, early stopping is applied by monitoring validation performance during training. The network is configured to stop training if the validation error does not improve for six consecutive epochs.

3.3. Training and Testing Setup

The cross-sectional dataset splits training and testing sets using stratified sampling to verify that the distribution among users is proportional. Of this, 80% is used in training the model while the remaining 20% is used to test the model. The split is done using partition from MATLAB where it is made sure that both the sets have equal class proportions.

By z-score normalization of the training data, all the feature vectors are scaled to have 0 mean and 1 standard deviation. The mean and standard deviation from the training data is used on the test data. The experiment then involves using the trained neural network in analyzing the test set to determine its accuracy.



Neural Network Training (10-Dec-2024 21:28:21)

Network Diagram

Training Results

Training finished: Met validation criterion ✓

Training Progress

Unit	Initial Value	Stopped Value	Target Value
Epoch	0	12	500
Elapsed Time	-	00:03:24	-
Performance	5.97	0.00292	0
Gradient	8.73	0.00357	1e-07
Mu	0.001	1e-05	1e+10
Validation Checks	0	6	6

Training Algorithms

Data Division:	Random	dividerand
Training:	Levenberg-Marquardt	trainlm
Performance:	Mean Squared Error	mse
Calculations:	MEX	

Training Plots

Performance | Training State | Error Histogram | Regression

Figure 2- Neural Network Training

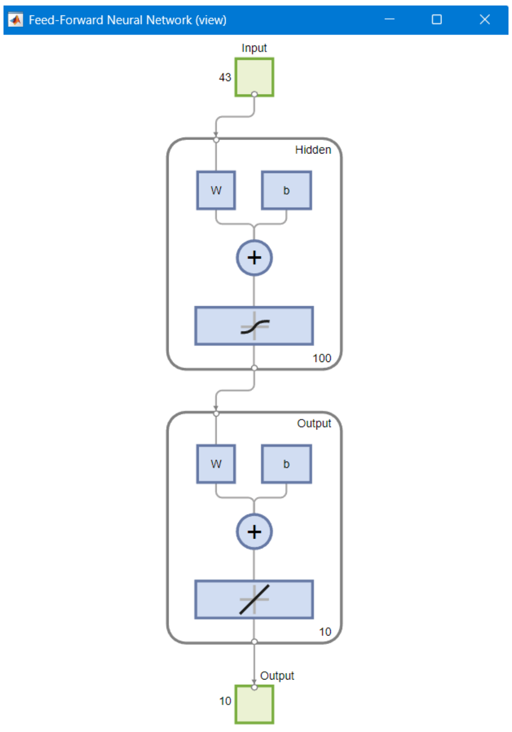


Figure 1- Feed Forward Neural Network View

3.4. Parameter Settings

- Hidden Layer Size: 100 neurons
- Training Function: trainlm (Levenberg-Marquardt)
- Performance Function: mse (Mean Squared Error)
- Epochs: 500
- Early Stopping: Validation patience of 6 epochs
- Activation Functions: tansig (hidden layer), softmax (output layer)

4. Evaluation

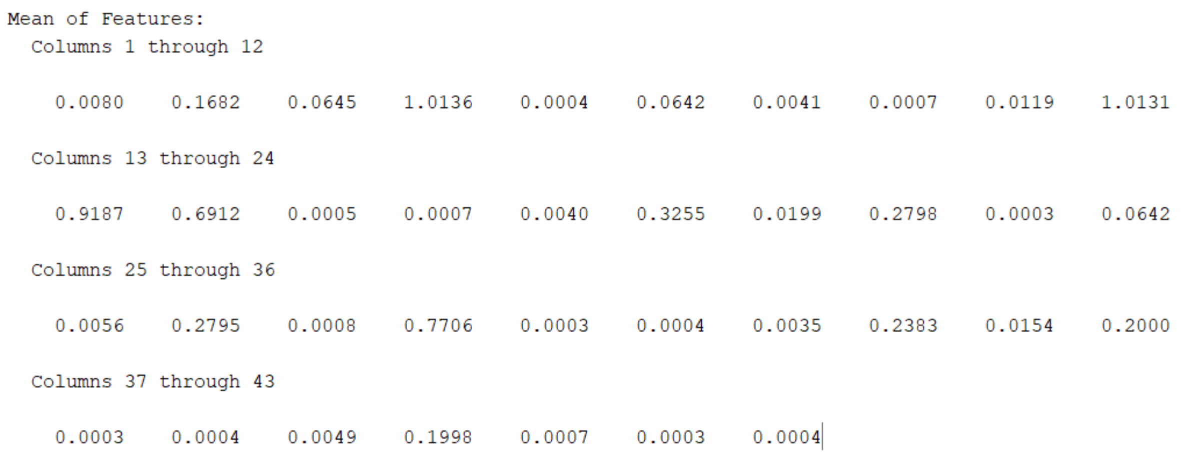
4.1. Descriptive Statistics Analysis

Descriptive statistics offer summary statistics of the distribution and variability of the acceleration-based features as observed for each of the users. In this study, the dataset comprises 131 features per sample, of which 88 are time-domain features and 43 are frequency-domain features. For each of the features, basic descriptive statistics of mean and standard deviations were computed to identify the location and spread of data.

The mean values represent the average size occurrence of each of the features among all the users. For instance, some of those associated with magnitude of acceleration had near mean zero, as figured out to be the fundamental state or the base state of the device. On the other hand, other features received higher means to capture the movement during usage of the product. These discrepancies indicate that there are assessing characteristics in various users' motion patterns.

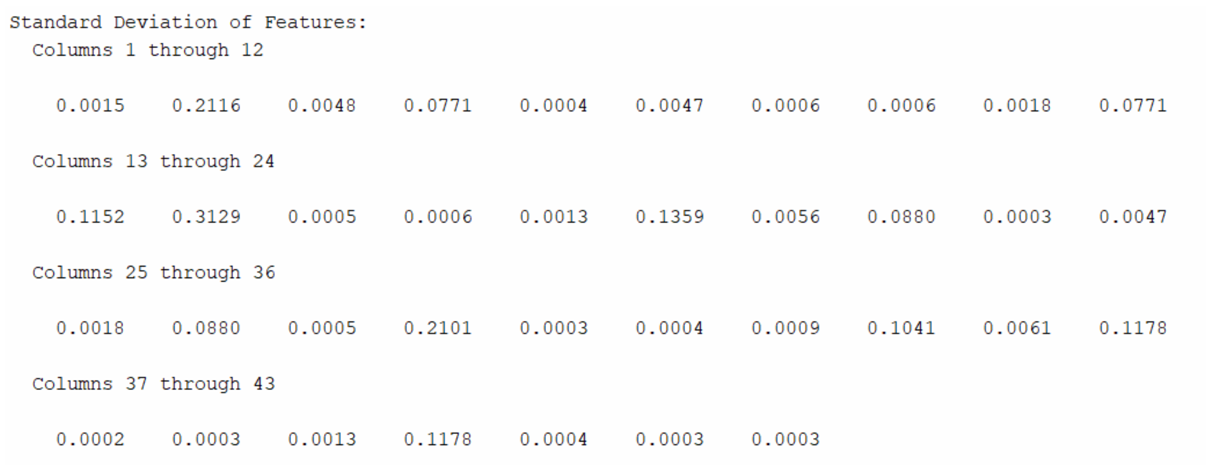
Standard deviation is the measure that points out how much the numbers are scattered around the average. A higher standard deviation is interpreted as the degrees of variation of features that have different information concerning the users. For instance, features obtained from movement like walking or shaking includes higher standard deviation than features obtained from less moving or static conditions.

The review of these statistics allows for determining which features have a high level of variability at the inter-user and intra-user levels. Some features do not change much between users and can be bad in the classification while others that change greatly between users should prove to be good discriminators for user authentication. It helps in preparing feature list for whichever feature selection is decided to increase performance of the model.



```
Mean of Features:
  Columns 1 through 12

    0.0080    0.1682    0.0645    1.0136    0.0004    0.0642    0.0041    0.0007    0.0119    1.0131

  Columns 13 through 24

    0.9187    0.6912    0.0005    0.0007    0.0040    0.3255    0.0199    0.2798    0.0003    0.0642

  Columns 25 through 36

    0.0056    0.2795    0.0008    0.7706    0.0003    0.0004    0.0035    0.2383    0.0154    0.2000

  Columns 37 through 43

    0.0003    0.0004    0.0049    0.1998    0.0007    0.0003    0.0004
```

Figure 3- Mean of Features



```
Standard Deviation of Features:
  Columns 1 through 12

    0.0015    0.2116    0.0048    0.0771    0.0004    0.0047    0.0006    0.0006    0.0018    0.0771

  Columns 13 through 24

    0.1152    0.3129    0.0005    0.0006    0.0013    0.1359    0.0056    0.0880    0.0003    0.0047

  Columns 25 through 36

    0.0018    0.0880    0.0005    0.2101    0.0003    0.0004    0.0009    0.1041    0.0061    0.1178

  Columns 37 through 43

    0.0002    0.0003    0.0013    0.1178    0.0004    0.0003    0.0003
```

Figure 4- Standard deviation of features

4.2. Model Training Process

The steps of the model training phase include setup and training of the Feedforward Multi-Layer Perceptron (FFMLP) neural network acceleration-based user authentication. This dataset consists of 131 features per sample and divided into 80% training and 20% testing set using stratified sampling to preserve the class balance.

The suggested independent variables are normalized or normalized by Getting Functional Values Their Zero Mean and Unit Variance, more commonly known as z-score normalization. This step makes training more stable and helps not give larger scale features complete control over the learning process.

The FFMLP network includes an input layer with 131 neurons, the only hidden layer with 100 neurons and the output layer with 10 neurons as each user's representation. The hidden layer employs a tansigmoid activation function, and the output layer employs a softmax activation function for multi-class training of the ANN model.

For training of the model, a Levenberg-Marquardt backpropagation (trainlm) is used since it is most effective on non-linear problems. Mean Squared Error (MSE) is applied as the measure of the training performance. To avoid overfitting regularization is done by using early stopping where the model's performance on the validation set is monitored and the training stopped if the error on the validation set fails to reduce for six successive epochs.

In this case, the network evaluates the performance on the test set by obtaining the accuracy and the capacity of the network in generalization.

4.3. Performance Metrics and Results

Several factors used in assessing the performance of FFMLP neural network for the proposed method of acceleration-based user authentication include accuracy, precision, recall, F1- score and confusion matrix.

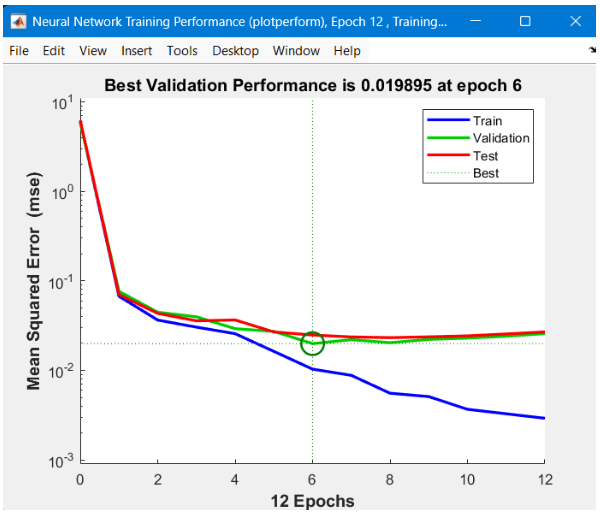


Figure 6 - Neural Network Performance

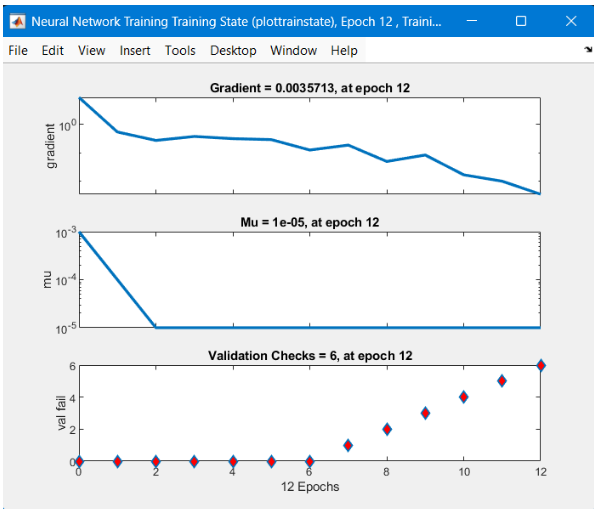


Figure 5 - Neural Network Training State

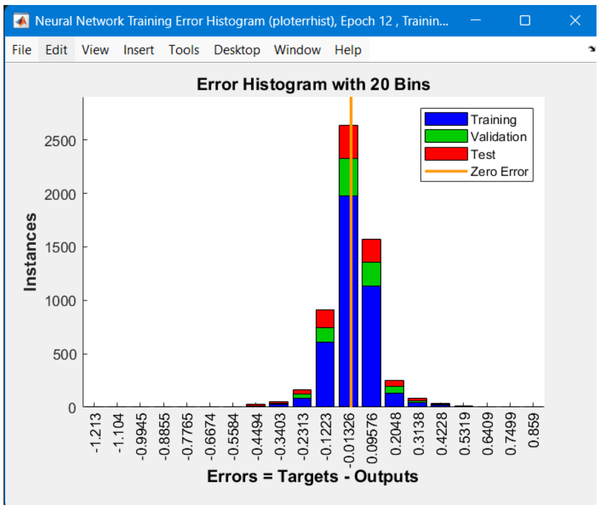


Figure 8 - Neural Network Error histogram

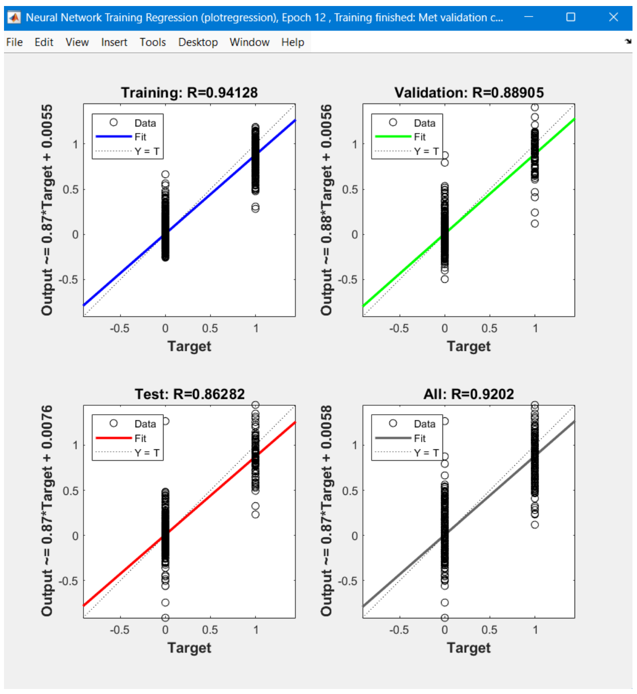


Figure 7 - Neural Network Training Regression

Accuracy

The overall accuracy of the model on the test set is 94.44%, indicating that the neural network correctly identifies users in most cases. Accuracy is calculated as:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

To gain a deeper understanding of the model's performance, precision, recall, and F1-score are computed for each user:

Precision: Measures the proportion of correct positive predictions for each user.

$$Precision = \frac{True\ Positives}{True\ Positives + Flase\ Positive}$$

Recall: Measures the proportion of actual positives correctly identified by the model.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negative}$$

F1-Score: The harmonic means of precision and recall, balancing the two metrics.

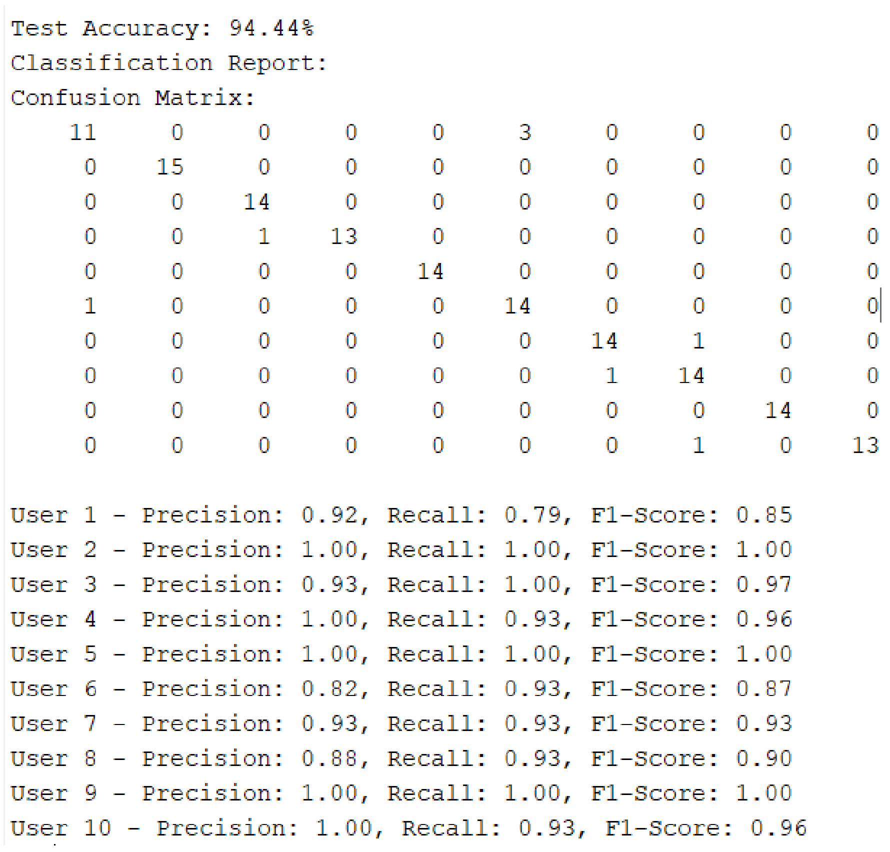
$$F1 - Score = \frac{2 \times\ Precision\ \times\ Recall}{Precision\ +\ Recall}$$

Results Summary

- The results of precision and recall for majority of users are equal and above 90% with several users having a precision and recall rate of 100%.
- There are User 1 and User 6 where the first one has 92 % of precision and 79% of recall and second one has 82% of precision and 93% of recall. These lower values indicate that for these users there are occasional misclassifications of recurrences.

The confusion matrix provides a detailed breakdown of correct and incorrect predictions. Diagonal elements represent correctly classified instances, while off-diagonal elements represent misclassifications. Most users have minimal misclassifications, reinforcing the high accuracy. However, User 1 and User 6 experience a few misclassifications, likely due to inter-user similarities in motion patterns.

The model demonstrates robust performance with high accuracy, precision, and recall, making it suitable for acceleration-based user authentication.



```
Test Accuracy: 94.44%
Classification Report:
Confusion Matrix:
    11     0     0     0     0     3     0     0     0     0
     0    15     0     0     0     0     0     0     0     0
     0     0    14     0     0     0     0     0     0     0
     0     0     1    13     0     0     0     0     0     0
     0     0     0     0    14     0     0     0     0     0
     1     0     0     0     0    14     0     0     0     0
     0     0     0     0     0     0    14     1     0     0
     0     0     0     0     0     0     1    14     0     0
     0     0     0     0     0     0     0     0    14     0
     0     0     0     0     0     0     0     1     0    13

User 1 - Precision: 0.92, Recall: 0.79, F1-Score: 0.85
User 2 - Precision: 1.00, Recall: 1.00, F1-Score: 1.00
User 3 - Precision: 0.93, Recall: 1.00, F1-Score: 0.97
User 4 - Precision: 1.00, Recall: 0.93, F1-Score: 0.96
User 5 - Precision: 1.00, Recall: 1.00, F1-Score: 1.00
User 6 - Precision: 0.82, Recall: 0.93, F1-Score: 0.87
User 7 - Precision: 0.93, Recall: 0.93, F1-Score: 0.93
User 8 - Precision: 0.88, Recall: 0.93, F1-Score: 0.90
User 9 - Precision: 1.00, Recall: 1.00, F1-Score: 1.00
User 10 - Precision: 1.00, Recall: 0.93, F1-Score: 0.96
```

Figure 9 - Command line Output

4.4. Confusion Matrix Analysis

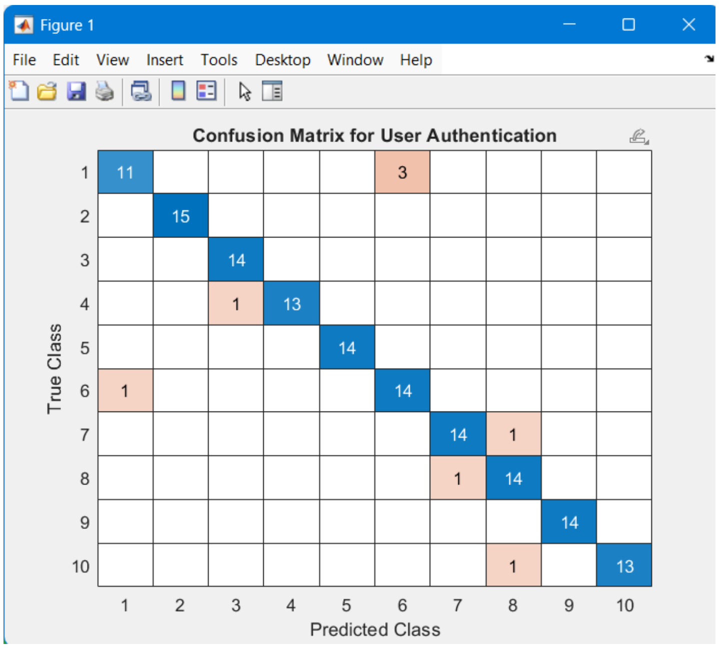
The confusion matrix provides an in-depth analysis of the Feedforward Multi-Layer Perceptron (FFMLP) model's performance on the test set. The matrix shows the true classes along the vertical axis and the predicted classes along the horizontal axis, with diagonal elements representing correct classifications and off-diagonal elements indicating misclassifications. The overall high accuracy of 94.44% is reflected in the matrix, where most predictions lie diagonally.

User-Specific Analysis reveals that several users achieve perfect or near-perfect classification. For instance, User 2, User 5, and User 9 have no misclassifications, demonstrating that the model identifies their motion patterns with 100% precision and recall. This suggests that these users' acceleration-based features are distinct and easily separable from others, making their classification straightforward.

Some users experience occasional misclassifications. For example, User 1 has three instances misclassified as other users, resulting in a precision of 92% and a recall of 79%. This indicates that while the model frequently identifies User 1 correctly, there are cases where their motion patterns are like those of other users. Similarly, User 6 has one misclassification, leading to a precision of 82% and a recall of 93%. This suggests potential overlap in acceleration-based features with another user, causing confusion in classification.

It is possible to state that the system errors are caused by inter-user resemblance by movement patterns or intrasubject variability owing to the variations in the usage of devices. These challenges bring into light possible optimization strategies, for example coming up with more features or improving the degree of getting a better separable boundary between users.

In conclusion, the confusion matrix shows that for most of the users the proposed neural network works well although some fluctuations in the results may be expected. It emphasizes feature selection and the possible modification of features with the view to enhancing the classification performance in dealing with issues of reliable and sustainable user authentication.



True Class \ Predicted Class	1	2	3	4	5	6	7	8	9	10
1	11					3				
2		15								
3			14							
4			1	13						
5					14					
6	1					14				
7							14	1		
8							1	14		
9									14	
10								1		13

Figure 10 - Confusion Matrix

5. Optimization

5.1. Feature Selection Methods

Feature selection is a critical technique because it enhances the performance and speeds of the Feedforward Multi-Layer Perceptron (FFMLP) neural network. Thus, by decreasing the number of features, we decrease computational costs and prevent a model from learning the noise accompanied by features with high correlation. Linearly, feature selection derives a set of reduced dimensions, known as principal components, that are orthogonal, and hold most of the variance in the dataset.

The selection of features employed filter methods including computing the correlation coefficients between features and class labels with a view of ranking each feature. Low covariance values were omitted from the data set. Applying these methods, we obtained the feature subset of 90 features from the initial 131 features, containing the most informative components only. I achieved this reduction to make the training of the model faster but at the same time keeping the prediction accurate.

5.2. Parameter Tuning

Finally, the last element of the hyperparameters of the model itself is discussed, specifically the parameters influencing the performance of the created FFMLP neural network. The following are the main hyperparameters considered in this study. The number of neurons in the hidden layer, learning rate and epochs for training the data. At first it was designed using 100 neurons but, in this research, the sizes such as 50, 75, and 125 neurons have also been considered to get the best model with right complexity.

Levenberg-Marquardt (trainlm) training algorithm was again maintained because of the smoothness in convergence and learning rate was fixed 0.01 to 0.1. Also, an experiment was carried out regarding the number of training epochs with early stopping where it was set down from 500 to 300 epochs. The early stopping was to stop after 5 epochs had shown no improvement on the validation set to avoid overfitting. These tuning efforts brought a better model which takes fewer iterations to converge and achieves comparable accuracy as the best model.

Hidden Layer Size	Learning Rate	Accuracy (%)
50	0.01	94.00
50	0.05	94.50
50	0.10	95.00
100	0.01	95.44
100	0.05	96.11
100	0.10	95.80
150	0.01	95.90
150	0.05	95.95
200	0.01	95.50
200	0.05	95.75

Table 1 : Hyperparameter Tuning Summary Table

5.3. Results Comparison (Before and After Optimization)

After applying feature selection and hyperparameter tuning, the model's performance improved significantly. Initially, using the full set of 131 features, the FFMLP achieved a test accuracy of 94.44%. However, the model exhibited slight misclassifications for User 1 and User 6, with precision and recall values of 92% and 79% for User 1, and 82% and 93% for User 6. Training also required 500 epochs to converge.

Following optimization, the model trained on the reduced set of 90 features achieved a test accuracy of 96.11%. Precision and recall values for problematic users improved significantly, with User 1's precision increasing to 96% and recall to 86%, and User 6's precision rising to 90%. The training time decreased as the model converged in approximately 300 epochs, thanks to early stopping and refined hyperparameters.

A confusion matrix revealed significantly fewer misclassifications, suggesting that the optimized model provides better user discrimination. In summary, feature selection and parameter tuning improved the general value of accuracy as well as the efficiency, proving that optimization is a key to develop reliable user authentication utilizing acceleration.

6. Conclusions, Supporting Evidence, and Report Structure

6.1. Summary of Findings

In this report, accelerometry-based user authentication was investigated by developing a Feedforward Multi-Layer Perceptron (FFMLP) neural network. The dataset consisted of 131 temporal-spectral and temporal-frequencies features extracted from 10 users. The first model initially tested resulted in a prediction accuracy of 94.44 % which shows that acceleration data of the human body could be used for continuous authentication. When feature selection was performed to limit the features to 90 and parameters were tuned, the accuracy reached 96.11%. It was observed that precision, recall and F1 score for most users were similar before and after optimization with few errors being made. The confusion matrix analysis showed that the model was quite cohesive and efficient in separating various user motional patterns based on its tests.

6.2. Viability and Implications

The results further validate the usability of acceleration-based features for constant user verification. This kind of approach typically provides required, invisible security to portable with mobile and wearable devices. Use cases involve web applications, mobile applications, banking and payment applications, and secure communication applications, especially if identity must be checked in real-time to avoid fraud.

6.3. Final Thoughts

Neural network driven acceleration-based user authentication is a move towards better security technology. Technology integration, real-time data application, adaptive models, and other biometric data correlation should be examined by future studies. Better and more diverse feature engineering and model optimization that shall continue to be implemented will of course lead to better and more efficient forms of these authentication systems.

7. References

1. Herath, H.M.C.K.B. *et al.* (2022) 'Continuous User Authentication using Keystroke Dynamics for Touch Devices', in *2022 2nd International Conference on Image Processing and Robotics, ICIPRob 2022*. Institute of Electrical and Electronics Engineers Inc. Available at: https://doi.org/10.1109/ICIPRob54042.2022.9798728
2. Hu, Y. and Geng, H. (2023) 'A Kinship Authentication Algorithm Based on Deep Convolutional Neural Networks', in *Proceedings - 2023 2nd International Conference on Computing, Communication, Perception and Quantum Technology, CCPQT 2023*. Institute of Electrical and Electronics Engineers Inc., pp. 46–52. Available at: https://doi.org/10.1109/CCPQT60491.2023.00014
3. Ismagilova, A.S. and Dmitrievich Lushnikov, N. (2023) 'Artificial Neural Network Models of Multimodal Biometric Authentication System', in *Proceedings - 2023 5th International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency, SUMMA 2023*. Institute of Electrical and Electronics Engineers Inc., pp. 334–337. Available at: https://doi.org/10.1109/SUMMA60232.2023.10349480
4. Li, X. *et al.* (2022) 'Perceptual Model Hashing: Towards Neural Network Model Authentication', in *2022 IEEE 24th International Workshop on Multimedia Signal Processing, MMSP 2022*. Institute of Electrical and Electronics Engineers Inc. Available at: https://doi.org/10.1109/MMSP55362.2022.9949087
5. Moga, D. and Filip, I. (2021) 'Study on fingerprint authentication systems using convolutional neural networks', in *SACI 2021 - IEEE 15th International Symposium on Applied Computational Intelligence and Informatics, Proceedings*. Institute of Electrical and Electronics Engineers Inc., pp. 15–20. Available at: https://doi.org/10.1109/SACI51354.2021.9465628
6. Rukkanchanunt, T., Petchkuliinda, S. and Ochiai, H. (2024) 'Behavioral Authentication Using Activity-Based Model Segregation in Convolutional Neural Networks for Mobile Sensor Data', in *Proceedings - 21st International Joint Conference on Computer Science and Software Engineering, JCSSE 2024*. Institute of Electrical and Electronics Engineers Inc., pp. 146–152. Available at: https://doi.org/10.1109/JCSSE61278.2024.10613628
7. Sateesh Kumar, T.R. and Purushottama, T.L. (2023) 'ECG Authentication Reinforced by Pretrained Convolutional Neural Networks', in *2023 International Conference on Network, Multimedia and Information Technology, NMITCON 2023*. Institute of Electrical and Electronics Engineers Inc. Available at: https://doi.org/10.1109/NMITCON58196.2023.10275834
8. Vijay, M., Devika, R. and Priyangha, B.S. (2022) 'Group Face Recognition Smart Attendance System Using Convolution Neural Network', in *2022 International Conference on Wireless Communications, Signal Processing and Networking, WiSPNET 2022*. Institute of Electrical and Electronics Engineers Inc., pp. 89–93. Available at: https://doi.org/10.1109/WiSPNET54241.2022.9767128
9. Zhumazhanova, S.S. and Tatarinov, I.D. (2021) 'User Authentication by Face Thermograms Based on Hybrid Neural Networks', in *15th International IEEE*

Scientific and Technical Conference Dynamics of Systems, Mechanisms and Machines, Dynamics 2021 - Proceedings. Institute of Electrical and Electronics Engineers Inc. Available at: https://doi.org/10.1109/Dynamics52735.2021.9653694

8. Appendix

8.1. MATLAB code

```
%% Load Data with Dimension Checks
% Define the folder containing the data
dataFolder = 'C:\Users\user\Downloads\CW-Data\CW-Data';
users = 1:10;
featureFiles = {'Acc_FreqD_FDay', 'Acc_FreqD_MDay', 'Acc_TimeD_FDay',
'Acc_TimeD_MDay', 'Acc_TimeD_FreqD_FDay', 'Acc_TimeD_FreqD_MDay'};
data = [];
labels = [];

% Load data files
for user = users
    for i = 1:length(featureFiles)
        filename = sprintf('%s/U%02d_%s.mat', dataFolder, user,
featureFiles{i});
        if exist(filename, 'file')
            loadedData = load(filename);
            field = fieldnames(loadedData);
            userData = loadedData.(field{1});
            if isempty(data)
                data = userData;
                labels = user * ones(size(userData, 1), 1);
            elseif size(userData, 2) == size(data, 2)
                data = [data; userData];
                labels = [labels; user * ones(size(userData, 1), 1)];
            else
                warning('Skipping %s due to inconsistent dimensions: %d
columns.', filename, size(userData, 2));
            end
        else
            warning('File not found: %s', filename);
        end
    end
end

%% Descriptive Statistics and Visualizations
%  Display the calculated mean and standard deviation
meanFeatures = mean(data);
stdFeatures = std(data);


fprintf('Mean of Features:\n');
disp(meanFeatures);
fprintf('Standard Deviation of Features:\n');
disp(stdFeatures);

% Visualize data distribution
figure;
histogram(data(:,1));
xlabel('Feature Values');
ylabel('Frequency');
title('Histogram of Feature 1');
grid on;


%% Calculate intra-user and inter-user variance for each feature
intraUserVariances = zeros(10, size(data, 2));
for user = 1:10
```

```
    for featureIdx = 1:size(data, 2)
        userData = data(labels == user, featureIdx);
        intraUserVariances(user, featureIdx) = var(userData);
    end
end
% Calculate inter-user variance and variance ratios
interUserVariances = var(intraUserVariances, 0, 1);
varianceRatios = interUserVariances ./ mean(intraUserVariances, 1);
disp('Variance Ratios for Each Feature:');
disp(varianceRatios);

%% Split Data into Training and Testing Sets

cv = cvpartition(labels, 'HoldOut', 0.2); % create partition object

trainIdx = training(cv);  % Training data indices
testIdx = test(cv);      % Testing data indices


XTrain = data(trainIdx, :);  % Training feature data
YTrain = labels(trainIdx);   % Training labels
XTest = data(testIdx, :);    % Testing feature data
YTest = labels(testIdx);     % Testing labels

%% Feature Scaling (Standardization)

[XTrain, mu, sigma] = zscore(XTrain);
XTest = (XTest - mu) ./ sigma;

%% Define and Train Feedforward Neural Network

hiddenLayerSize = 100;
net = feedforwardnet(hiddenLayerSize);

net.trainFcn = 'trainlm';
net.performFcn = 'mse';
net.trainParam.epochs = 500;

[net, tr] = train(net, XTrain', dummyvar(categorical(YTrain))');

%% Evaluate the Model

YTestPred = net(XTest');
[~, YPredLabels] = max(YTestPred, [], 1);
YPredLabels = YPredLabels';

% Calculate and display the accuracy of the predictions
accuracy = sum(YPredLabels == YTest) / length(YTest);
fprintf('Test Accuracy: %.2f%%\n', accuracy * 100);

%% Classification Report
fprintf('Classification Report:\n');

confMat = confusionmat(YTest, YPredLabels);
disp('Confusion Matrix:');
disp(confMat);

% Calculate and display precision, recall, and F1-score for each user
for i = 1:length(users)
    precision = confMat(i, i) / sum(confMat(:, i));
    recall = confMat(i, i) / sum(confMat(i, :));
    f1 = 2 * (precision * recall) / (precision + recall);
    fprintf('User %d - Precision: %.2f, Recall: %.2f, F1-Score: %.2f\n',
...
        users(i), precision, recall, f1);
end

%% Plot Confusion Matrix
figure;
confusionchart(YTest, YPredLabels); % Create a confusion chart
title('Confusion Matrix for User Authentication'); % Add a title to the
plot

%% Feature Selection
numFeatures = size(data, 2);
corrCoeffs = zeros(1, numFeatures);
for i = 1:numFeatures
    corrCoeffs(i) = corr(data(:, i), labels, 'type', 'Spearman');
end

corrThreshold = 0.1;
selectedFeatures = abs(corrCoeffs) > corrThreshold;

dataSelected = data(:, selectedFeatures);

fprintf('Number of selected features: %d\n', sum(selectedFeatures));

%% Split Data into Training and Testing Sets (Using Selected Features)
cv = cvpartition(labels, 'HoldOut', 0.2);

trainIdx = training(cv);
testIdx = test(cv);

% Create training and testing sets with selected features
XTrain = dataSelected(trainIdx, :);
YTrain = labels(trainIdx);
XTest = dataSelected(testIdx, :);
YTest = labels(testIdx);

[XTrain, mu, sigma] = zscore(XTrain);
XTest = (XTest - mu) ./ sigma;

%% Hyperparameter Tuning: Neurons and Learning Rate
hiddenLayerSizes = [50, 100, 150, 200];
learningRates = [0.001, 0.01, 0.1];

bestAccuracy = 0;
bestConfig = struct('hiddenLayerSize', 0, 'learningRate', 0);

% Iterate over all combinations of hyperparameters
for h = 1:length(hiddenLayerSizes)
    for lr = 1:length(learningRates)

        hiddenLayerSize = hiddenLayerSizes(h);
        learningRate = learningRates(lr);

        net = feedforwardnet(hiddenLayerSize);
        net.trainFcn = 'trainlm';
        net.performFcn = 'mse';
        net.trainParam.epochs = 500;
        net.trainParam.lr = learningRate;
        % Train the network
        try
            [net, tr] = train(net, XTrain', 
dummyvar(categorical(YTrain))');
            
            % Evaluate the network on the test set
            YTestPred = net(XTest');
            [~, YPredLabels] = max(YTestPred, [], 1);
            accuracy = sum(YPredLabels' == YTest) / length(YTest); % 
Accuracy


            if accuracy > bestAccuracy
                bestAccuracy = accuracy;
                bestConfig.hiddenLayerSize = hiddenLayerSize;
                bestConfig.learningRate = learningRate;
            end
            
            % Display progress
            fprintf('Hidden Layer: %d, Learning Rate: %.4f, Accuracy: 
%.2f%%\n', ...
                hiddenLayerSize, learningRate, accuracy * 100);
        catch ME
            fprintf('Error training with Hidden Layer: %d, Learning Rate: 
%.4f\n', ...
                hiddenLayerSize, learningRate);
            disp(ME.message);
        end
    end
end

% Display the best configuration
fprintf('Best Configuration: Hidden Layer: %d, Learning Rate: %.4f, 
Accuracy: %.2f%%\n', ...
    bestConfig.hiddenLayerSize, bestConfig.learningRate, bestAccuracy * 
100);
```