Fall Detection with Phone Sensor Accelerometer Data: A Comparative Study of Neural Network and Decision Tree Models

Abstract

The elderly population is frequently subjected to abandonment and isolation, either through living independently or in the company of others who are often absent from their home environment. This state of affairs is precarious and constant, for it exposes them to the possibility of falling and not receiving prompt assistance. Research has revealed that the effects of not receiving medical attention after a fall pose a greater danger to the elderly than the physical impact of the fall itself. Timely intervention after a fall incident can prevent fatal outcomes in the elderly population. In the current century, smartphones have become ubiquitous and permeated all demographic segments, from metropolitan cities to rural villages and from teenagers to senior citizens, resulting in widespread ownership. All standard smartphones are equipped with an accelerometer, which can be leveraged to capture acceleration data and, subsequently, to detect falls. This paper introduces two distinct methodologies for fall detection utilising accelerometer data, one based on a neural network and the other on decision trees. We present a comprehensive comparative analysis of both approaches to determine their strengths and limitations.

Introduction

Falls are a major cause of injury and hospitalization among the elderly population. Early detection of falls is critical to prevent severe outcomes such as fractures, hospitalization, and even death. In recent years, the use of smartphone sensors, particularly accelerometer data, has become increasingly popular in developing fall detection systems. The accelerometer data can be used to measure the changes in acceleration caused by a fall and distinguish them from other movements. In this study, we aim to investigate the performance of machine learning models, namely neural networks and decision trees, in detecting falls based on accelerometer data from standard smartphone sensors.

Several studies have previously explored the use of accelerometer data in fall detection systems. For instance, Harari et al. [1] developed a fall detection system using a smartphone's accelerometer data and machine learning algorithms, achieving an accuracy of 96.6%. Similarly, Sadreazami et al [2]. used accelerometer data from a smartphone to develop a fall detection system. These studies demonstrate the potential of using machine learning techniques with accelerometer data for fall detection.

In this study, we used publicly available accelerometer data from a sample of individuals to develop and evaluate our machine learning models. Our results demonstrate that both the neural network and decision tree models have high accuracy in detecting falls, with the neural network model achieving slightly better performance. The findings of this study have significant implications for fall detection technology, particularly in the widespread use of smartphone sensors, to improve the safety and well-being of elderly individuals.

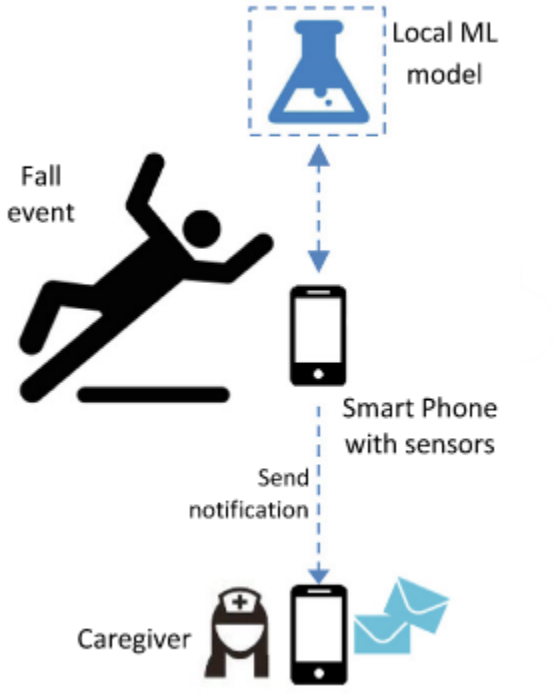


Fig 1 Architecture of Solution Proposed

In summary, this study highlights the importance of early detection of falls among the elderly population and the potential of using machine learning techniques with accelerometer data from standard smartphone sensors. The results of this study can help inform the development of future fall detection systems, particularly those designed for use in real-world settings. The research paper is structured as follows: Section 2 comprehensively reviews the existing literature on fall detection systems, highlighting their strengths and limitations. Section 3 details the dataset used in our study and provides insights into its features and characteristics. Section 4 outlines the methodology adopted in our research, encompassing the preprocessing of data, feature selection, and the implementation of machine learning models. Section 5 presents the experimental results obtained from our solution and critically evaluates the performance of the employed models. Finally, we conclude our study in Section 6, summarizing the key findings and implications for future research.

Related Work

The problem of detecting falls has been extensively researched, with numerous proposed solutions falling into two distinct categories: those leveraging visual cues [8] and those utilizing non-visual or numerical data [3,4,5,6,7]. While each approach presents unique advantages and drawbacks, it is generally recognized that solutions relying on visual data demand more resources, such as the need for a continuously recording video device. Conversely, non-visual data-based solutions can rely on simpler devices such as an accelerometer commonly found in smartphones. This paper proposes a non-visual data-based solution for fall detection and thus focuses on reviewing relevant literature in the field of non-visual data-based fall detection methods.

In their paper, Wiprayoga et al. [3] introduced an innovative solution for identifying fall events and distinguishing them from routine daily activities, utilizing LSTM/RNN networks. While this approach represents a significant advancement within the field, its practical application in real-time settings remains limited. Specifically, the use of LSTMs can result in a relatively large footprint and protracted training and prediction times compared to standard neural networks or other established methods such as Decision Trees or Support Vector Machines. Given that the accurate and prompt detection of fall incidents is critical, with every second potentially

influencing the outcome, it is imperative to seek out a solution that can deliver low-latency performance while maintaining high efficacy in real-time applications.

In a recent study, Ramanathan et al. [4] proposed an alternative solution based on Residual Networks (ResNet) for fall detection. While ResNets offer promising theoretical performance, they may not be suitable for real-world deployment due to their heavy computational requirements. Fall detection models are often deployed on edge devices with limited computational power; thus, models with a large memory footprint are not preferred.

D. Mrozek et al. [5] have offered a distinct perspective in their study by taking into account the practical constraints of low-computation devices prevalent in real-world settings. Instead of simplifying their model to address these constraints, they chose to shift their computations to the cloud, a noteworthy innovation. However, this approach has limitations in that it is reliant on an internet connection to perform computations and detect falls.

A comprehensive review of the literature revealed a gap in the solutions available for fall detection, with none of the current approaches focusing on reducing model complexity. Instead, these models prioritize accuracy, often at the expense of significantly increasing model complexity. To address this gap, we have evaluated various approaches, including neural networks and decision trees, to propose a solution with reduced training and prediction times while maintaining high accuracy.

Datasets

For this study, we utilized the publicly available SmartPhone Human Fall Dataset [9], which comprises of accelerometer and gyroscope data collected from 11 individuals performing 13 activities, classified into 4 falls and 9 ADLs (Activities in Daily Life). The dataset's activity code, label, and category are described in the table below.

Activity Code	Activity Label	Category
FOL	Forward-lying	Fall Activity

FKL	Forward knee-lying	Fall Activity
SDL	Sideward-lying	Fall Activity
BSC	Back sitting chair	Fall Activity
STD	Standing	Activity in Daily Life
WAL	Walking	Activity in Daily Life
JOG	Jogging	Activity in Daily Life
JUM	Jumping	Activity in Daily Life
STU	Stairs up	Activity in Daily Life
STN	Stairs down	Activity in Daily Life
SCH	Sit chair	Activity in Daily Life
CSI	Car-step in	Activity in Daily Life
CSO	Car-step out	Activity in Daily Life

Each instance in the dataset represents a 6-second window that contains tri-axial accelerometer and gyroscope data. We extracted 13 features from the data points to effectively train our model for activity recognition.

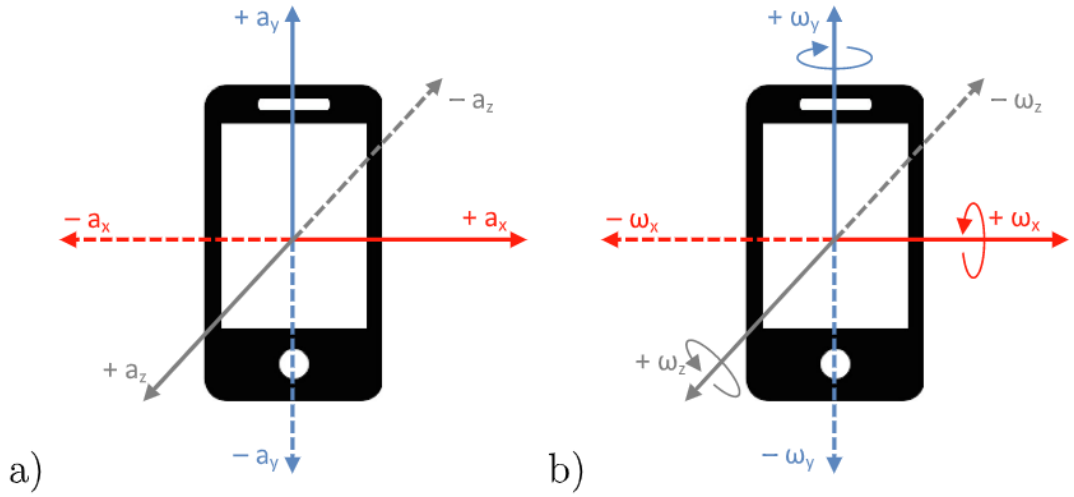


Fig. 2. Acceleration and angular velocity provided by triaxial accelerometer and gyroscope mounted in a smartphone.

The feature set used for activity recognition includes acc_max, which represents the maximum acceleration magnitude during the 4th second of the window; acc_kurtosis, which denotes the kurtosis of acceleration for the entire window; acc_skewness, which represents the skewness of acceleration for the entire window; gyro_max, which denotes the maximum gyroscope magnitude during the 4th second of the given window; gyro_kurtosis, which denotes the kurtosis of gyroscope for the entire window; gyro_skewness, which denotes the skewness of gyroscope for the entire window; lin_max, which represents the maximum linear acceleration (i.e., acceleration excluding gravity) during the 4th second of the given window; post_lin_max, which denotes the maximum linear acceleration during the 6th second; post_gyro_max, which denotes the maximum gyroscope magnitude during the 6th second. Additionally, the fall parameter is included in the feature set, with a value of 1 if the activity is a fall activity and 0 otherwise. Lastly, each activity is labelled accordingly in the 'label' column.

Methodology

We propose a two-phase approach in our paper. We aim to compare both the traditional machine learning algorithms like decision trees and the nuanced neural networks. As mentioned earlier, the objective is to keep the model footprint as low as possible without compromising performance.

The methodology for both these approaches is different and explained below.

Decision Trees

Decision Trees are a traditional supervised machine learning algorithm that combines numerous probability trees. Each tree branch utilizes a combination of parameters to bifurcate data points into two categories, enabling effective binary classification. This process is reiterated across multiple branches and trees to create a model that can perform accurate binary classification. However, traditional algorithms are not proficient at feature extraction; therefore, we manually identify essential features for model training and exclude the others (covered in the next section).

Decision Trees' performance relies heavily on its hyperparameters, which include n_estimators, max_depth, min_samples_split, min_samples_leaf, and max_features. To guarantee optimal model performance, it is necessary to optimize these parameters. We first initiate a breadth-first grid search to identify the optimized parameter range for the model. This is followed by a depth-first search to determine the best model parameters. The primary objective is to narrow down the parameter range using breadth-first search and identify the optimal values within that range using depth-first search.

Feature Extraction

We utilise a multifaceted approach to identify the salient features from the dataset of 13 extracted features. Initially, we compute the correlation matrix to ascertain the interrelationship between each feature and the target variable. We retain features exhibiting strong correlation coefficients with the target variable, whereas features with a low absolute value of correlation coefficient are selected for potential removal from the feature set. Subsequently, we employ feature-pair correlation analysis to identify features with a high degree of mutual correlation and remove one to reduce data dimensionality, thereby improving performance. Additionally, we calculate the mutual information scores (MIS) for all features with the target variable. The MIS and

correlation coefficient results consistently highlight that the features gyro_max and lin_max are less relevant, and thus, they are removed from the feature set. Moreover, we observe that gyro_skewness and gyro_kurtosis display a high positive correlation, and we opt to exclude gyro_skewness without any preference for either. After discarding these superfluous features, we effectively streamline the model's feature set to 10, resulting in more accurate classification.

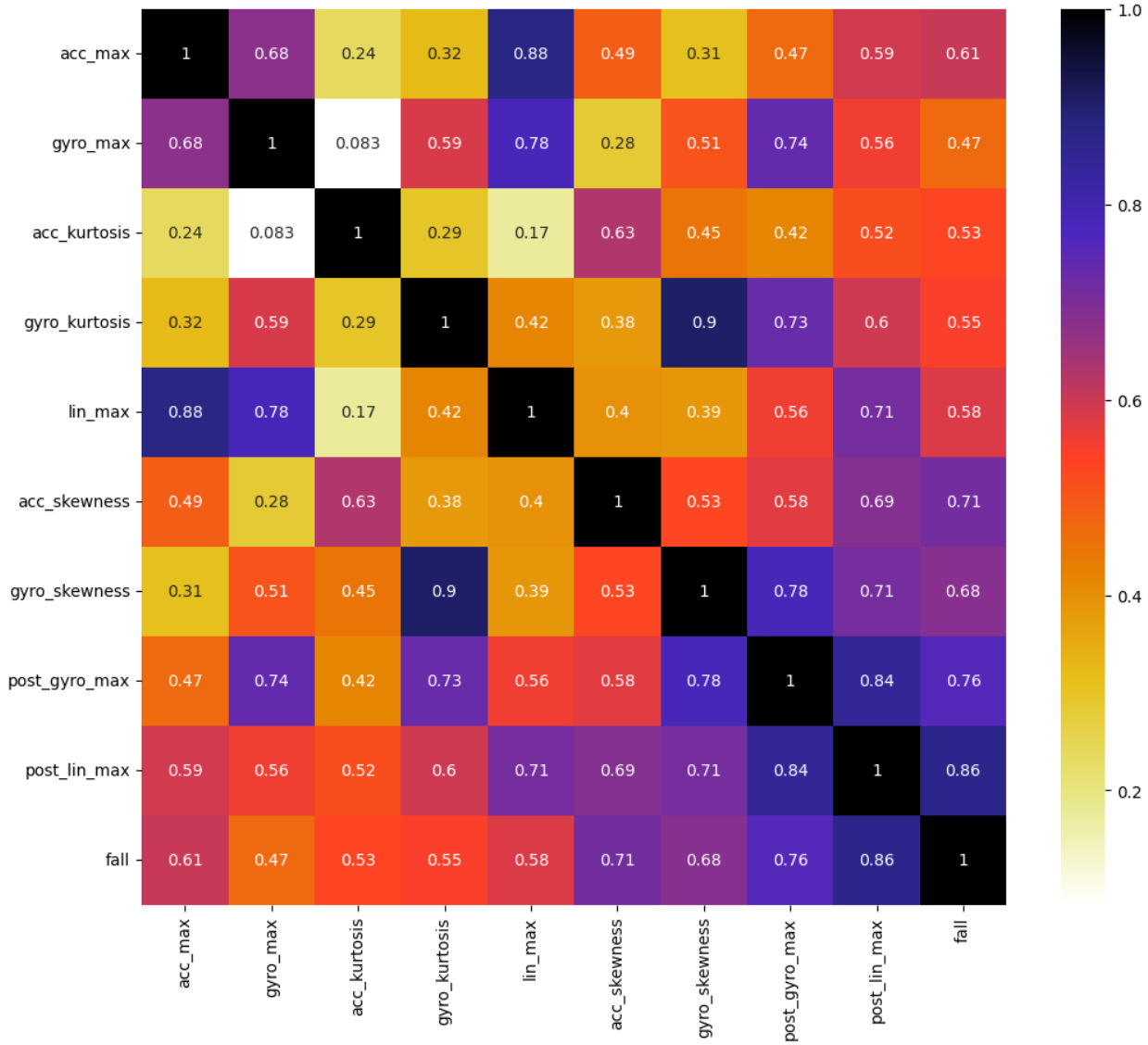


Fig-3 Correlation Matrix (Features and Target Variable)

Fig-4 Swarm Plot (Main Features)

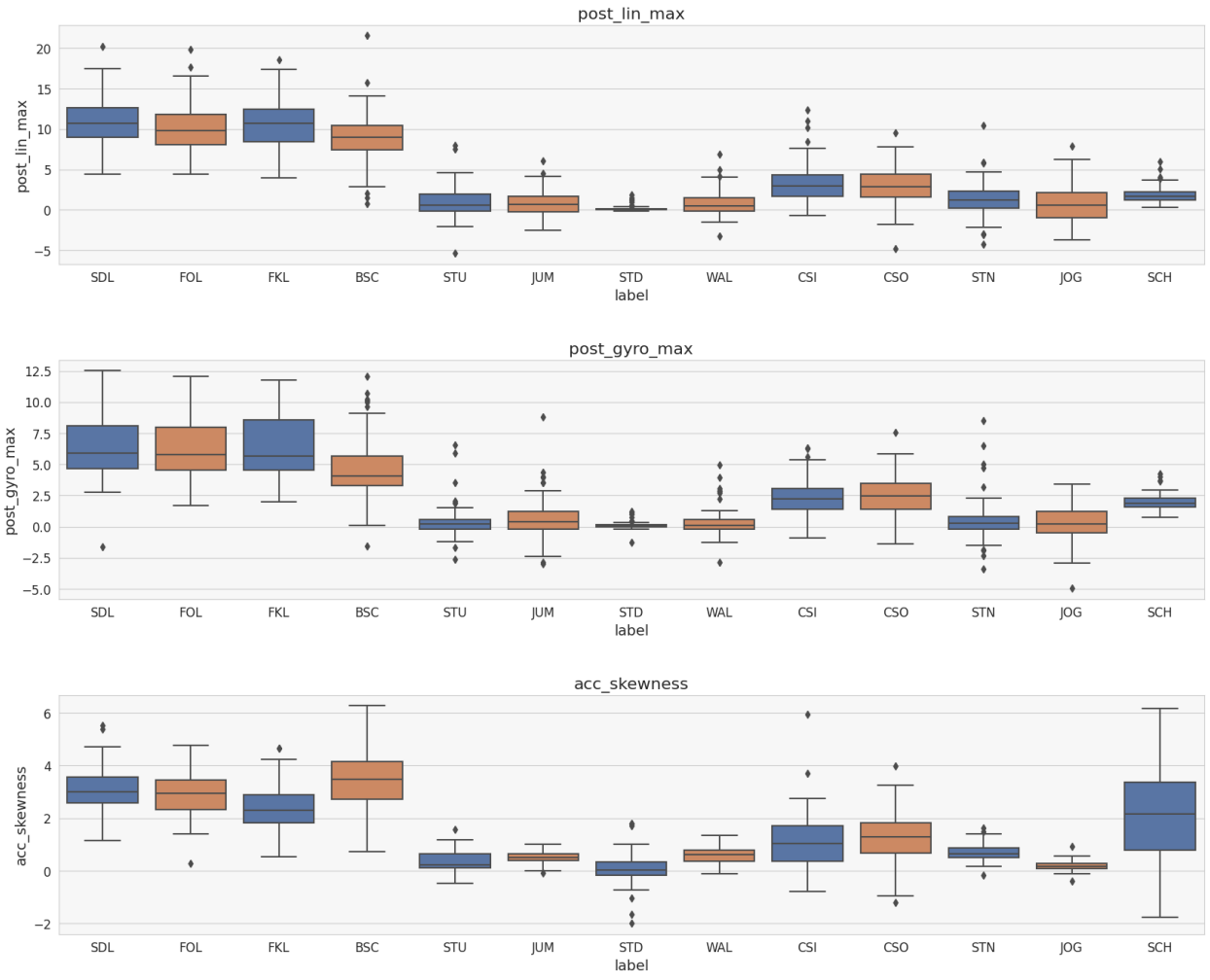


Fig-5 Box Plot Distribution (Main Features)

Based on the swarm plot and box plot presented in Figure 4 and Figure 5, we observe a notable differentiation in the distribution of features (that have high correlation coefficients with the target variable) between the activities categorized as falls (first four) and those associated with daily life (last nine). This observation indicates that the model is likely to perform accurate classification, instilling confidence in its efficacy.

Neural Network

Unlike traditional machine learning algorithms, neural networks have superior feature extraction capability. Thus, we assigned the responsibility of feature extraction to the neural network and inputted all 13 features. With the aim of developing an efficient yet effective model, we

constructed a neural network with a sequential architecture. This architecture consisted of four dense layers and three dropout layers. The first dense layer comprised 256 neurons, which was followed by a dropout layer with a 30% rate. The second dense layer contained 128 neurons, followed by another dropout layer with a 30% rate. The third dense layer had 64 neurons, followed by a dropout layer with a 30% rate. Finally, the fourth dense layer had a single neuron that represented the model's binary output.

To attain the best performance for the task, we selected the model's hyperparameters through a grid search to identify the optimal combination of the number of neurons in the hidden layers and the most effective dropout rate for the internal layers.

Experiments & Performance Evaluation

This section comprehensively discusses the simulation parameters and experimental findings. Specifically, this section outlines the procedures and parameters utilized in the study, as well as crucial observations and findings derived from the results.

By providing detailed information on the experimental setup and methodology, this section enables the replication of the study, which is crucial for verifying the accuracy and robustness of the results.

Feature Selection

The initial step preceding the training of decision trees is feature selection. We first computed the Pearson correlation coefficient among the various features to facilitate this process. Subsequently, we removed one of the features from each pair having a high absolute value of the correlation coefficient, as it was deemed that one of the features could effectively capture the variation in both. A threshold limit of 0.85 was established, and any two features with a correlation coefficient exceeding this value were deemed similar, with one of the features selected for removal. To identify the features for removal, we iterated over the upper triangle in the correlation coefficient matrix, selecting the feature on the horizontal axis for removal if the coefficient for the pair of features was greater than 0.85. Through this process, two features were identified that could be dropped from the feature set.

Furthermore, we employed mutual information score (MIS) to identify which features were significant for accurately predicting the target variable. We selected the two features with the lowest MIS values for removal. However, one of these features was the same as one of the features previously identified for removal through the Pearson correlation coefficient analysis. Therefore, one feature was ultimately removed based on the MIS score. In total, three features, namely gyro_max, gyro_skewness, and lin_max, were dropped from the feature set through these methods.

Decision Trees

We begin by standardizing our training and testing data using the StandardScaler from the sklearn.preprocessing library to ensure that each feature has a mean of zero and a standard deviation of one. Next, we define a parameter grid to search over using the GridSearchCV function from the sklearn.model_selection library. This grid includes hyperparameters such as the number of trees in the forest (n_estimators), the maximum depth of each tree (max_depth), and the minimum number of samples required to split an internal node (min_samples_split), among others.

We then create a RandomForestClassifier object and pass it to the GridSearchCV function along with the parameter grid and other hyperparameters such as the number of folds for cross-validation (cv=5) and the number of CPU cores to use (n_jobs=-1). We fit this grid search object to the training data and obtain the best hyperparameters by printing the best_params_ attribute of the fitted object. This approach allows us to optimize our random forest classifier for a given classification task and dataset, potentially improving its performance and generalization capabilities. The grid search results identified the optimal hyperparameters as follows:

Parameter	Value
bootstrap	False
class_weight	None
max_depth	10

max_features	sqrt
min_samples_leaf	4
min_samples_split	10
n_estimators	50

These hyperparameters provide the highest accuracy of the random forest classifier for the given dataset. The model accuracy with these hyperparameters was 97.19%. To further improve upon this, we did a depth search on the parameters grid to zero down on the best parameters in the identified range. Finally, the best model had an accuracy of 97.75%.

Neural Network

In this study, we developed a neural network model for binary classification using Keras. The model architecture consisted of one input layer, multiple hidden layers, and one output layer. We used the Rectified Linear Unit (ReLU) activation function in the hidden layers and the Sigmoid activation function in the output layer. To prevent overfitting, we used dropout regularization after each hidden layer.

We optimized the hyperparameters of the model using Grid Search Cross-Validation. The hyperparameters that we tuned were the number of neurons in the hidden layers and the dropout rate. We used a range of values for the number of neurons in the hidden layers (128, 64, and 32 for the first hidden layer, 256, 128, and 64 for the second hidden layer, and 512, 256, and 128 for the third hidden layer) and a range of values for the dropout rate (0.3, 0.5, and 0.7). We used the Adam optimizer with a learning rate of 0.001 and binary cross-entropy as the loss function.

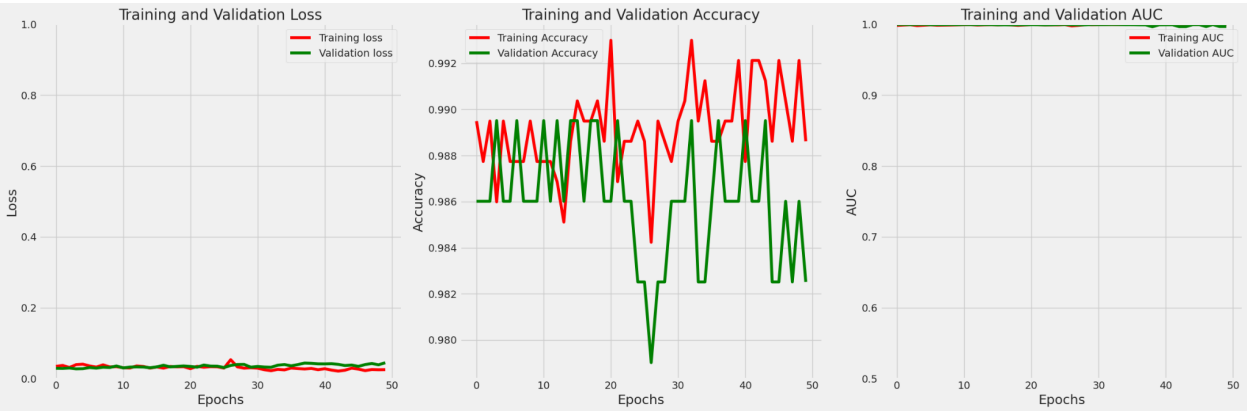
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               2560

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_1 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 64)                8256

 dropout_2 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 43,777
Trainable params: 43,777
Non-trainable params: 0
_________________________________________________________________
```

Fig-6 Model Architecture

We trained the best model on the full training set and evaluated its performance on the test set using classification metrics such as precision, recall, and F1-score. The best model had a validation accuracy of 0.98 and consisted of three hidden layers with 256, 128, and 64 neurons, respectively. The dropout rate was 0.5. We achieved a test accuracy of 0.98, a precision of 0.97, a recall of 0.98, and an F1-score of 0.97. These results demonstrate the effectiveness of our neural network model for binary classification tasks.

The results of this study demonstrated the effectiveness of the developed neural network model for predicting a binary outcome based on a set of input features. The optimized hyperparameters and the model architecture can be used as a guideline for future studies involving similar prediction tasks. The set of hyperparameters obtained is available in the code file accompanying this paper.

Fig-7 Model Training (Loss, Accuracy, AUC Curve)



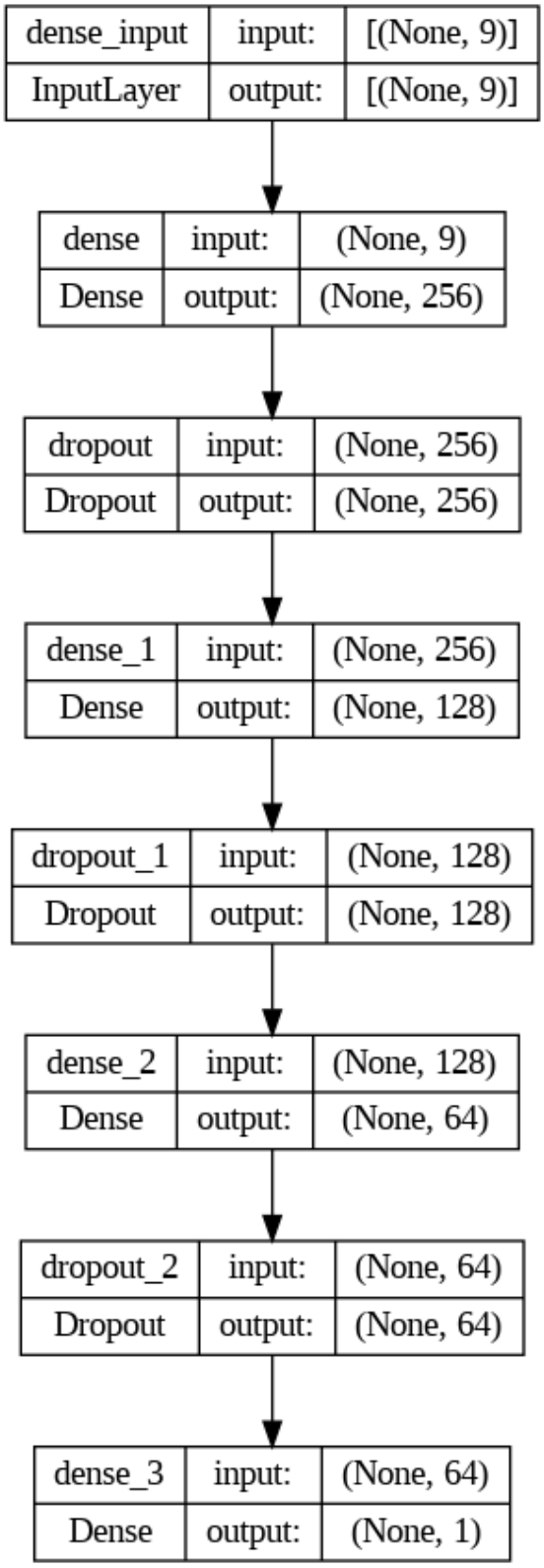


Fig-8 Model Layers

Performance Evaluation

To sum up how both models performed, we present the following confusion matrix for both the models.

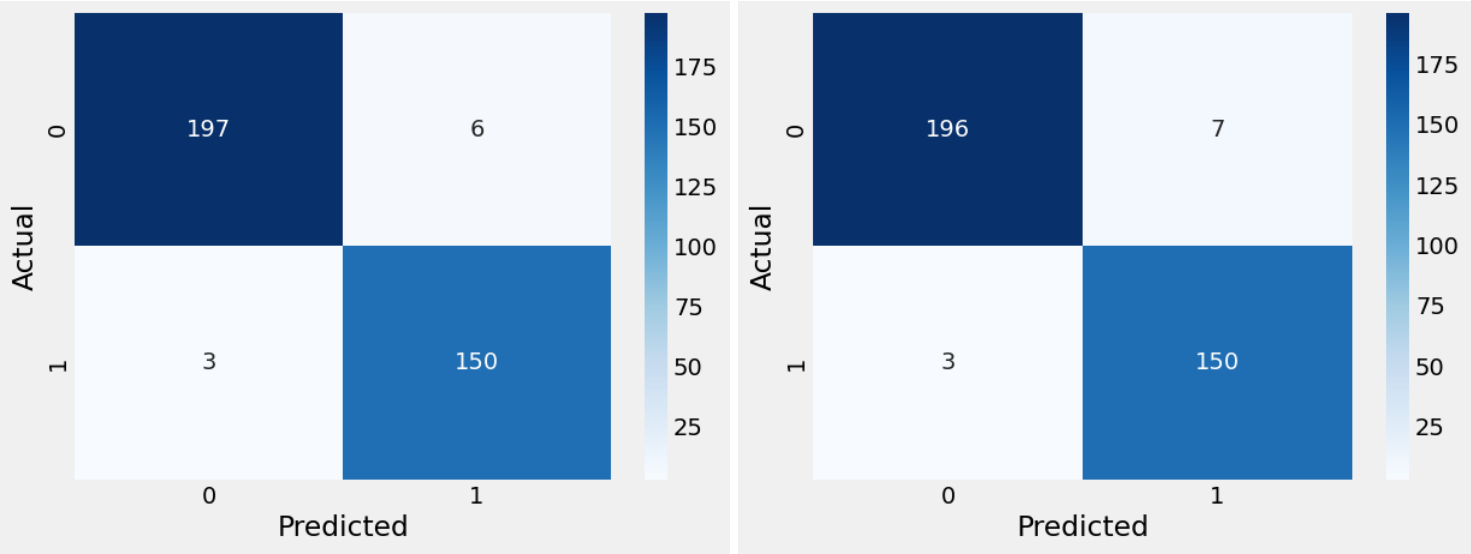


Fig- 9 Confusion Matrix (Neural Network & Decision Tree)

The horizontal axis depicts the actual labels where 1 indicates a positive instance i.e. and activity of fall and 0 indicates ADL. The vertical axis indicates the label predicted by the model. The left figure corresponds to that of neural network and the right one corresponds to decision trees model. This summarises the overall performance of both the approaches.

Results

The results obtained from the conducted experiments provide empirical evidence that the neural network model outperforms decision trees, albeit with a negligible difference. While the decision trees exhibited an accuracy of 97.19% (the first model), the neural network model achieved an accuracy of 98%, which implies that the former outperformed the latter by a marginal amount. Given that the performances of both models are similar, it is pertinent to compare other associated features of the models to determine the better option.

The decision trees model is characterized by a lower complexity level than the neural network model, resulting in a smaller footprint and lower computational requirements. In addition, decision trees are interpretable, unlike neural networks, which operate as a black box, rendering them less transparent and amenable to interpretation. However, the neural network model exhibits remarkable flexibility in handling raw data without the need for feature engineering. This feature affords it the ability to achieve higher accuracy rates than decision trees, making it an attractive option for certain applications.

Furthermore, the neural network model generalizes better to data from various sources than decision trees. On the other hand, decision trees may improve their performance through drift detection and generalization to data from different sources. Given that the primary objective of this research is to design a computationally efficient and effective model, the results suggest that decision trees represent a better choice based on our key performance indicators.
In summary, while the neural network model outperforms decision trees by a marginal amount, other factors, such as complexity, interpretability, and generalization, must be considered when selecting the most suitable model. Ultimately, the selection should be based on the key performance indicators relevant to the task at hand.

Conclusion & Future Work

This study explores two distinct approaches for fall detection using an accelerometer and gyroscope. The authors investigated the unique methodologies required for decision trees and neural networks and provide an analysis of the advantages and disadvantages of each approach, while also recommending which approach should be utilized based on specific circumstances.
The primary objective of this study is to address a significant gap in current literature by developing lightweight models that are both computationally efficient and highly effective. As a result, this study proposes novel solutions that meet these criteria.

As a potential area for future work, one could experiment with the model performance using data from various devices, each with varying recording frequencies and errors. This will help assess the model's ability to generalize across different device types and recording settings, thus highlighting its effectiveness in practical scenarios. Moreover, since real-world accelerometers

may be skewed, determining how well the model performs in these cases will be critical to assessing its practical utility.

Furthermore, this study opens up opportunities for data analytics by collecting acceleration data and identifying critical relationships that could benefit individuals who are at risk of falling. Such an approach could lead to improved fall detection systems and ultimately improve the quality of life for at-risk populations.

In conclusion, this study presents a comprehensive analysis of fall detection using accelerometers and gyroscopes, with a focus on developing lightweight models. The proposed solutions are highly effective and computationally efficient, making them ideal for practical applications. Future work should aim to further refine and evaluate these models using data from diverse sources and exploring potential data analytics approaches to improve fall detection systems.

References

[1] Y. Harari, N. Shawen, C. K. Mummidisetty, M. V. Albert, K. P. Kording, and A. Jayaraman, “A smartphone-based online system for fall detection with alert notifications and contextual information of real-life falls,” *J. Neuroeng. Rehabil.*, vol. 18, no. 1, p. 124, 2021.

[2] H. Sadreazami, M. Bolic, and S. Rajan, “Contactless fall detection using time-frequency analysis and convolutional neural networks,” *IEEE Trans. Industr. Inform.*, vol. 17, no. 10, pp. 6842–6851, 2021.

[3] I. Wayan Wiprayoga Wisesa and G. Mahardika, “Fall detection algorithm based on accelerometer and gyroscope sensor data using Recurrent Neural Networks,” *IOP Conf. Ser. Earth Environ. Sci.*, vol. 258, p. 012035, 2019.

[4] A. Ramanathan and J. McDermott, “Fall detection with accelerometer data using Residual Networks adapted to multi-variate time series classification,” in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021.

[5] D. Mrozek, A. Koczur, and B. Małysiak-Mrozek, “Fall detection in older adults with mobile IoT devices and machine learning in the cloud and on the edge,” *Inf. Sci. (Ny)*, vol. 537, pp. 132–147, 2020.

[6] M. V. Albert, K. Kording, M. Herrmann, and A. Jayaraman, “Fall classification by machine learning using mobile phones,” *PLoS One*, vol. 7, no. 5, p. e36556, 2012.

[7] S. Mellone, C. Tacconi, L. Schwickert, J. Klenk, C. Becker, and L. Chiari, “Smartphone-based solutions for fall detection and prevention: the FARSEEING approach,” *Z. Gerontol. Geriatr.*, vol. 45, no. 8, pp. 722–727, 2012.

[8] V. Mehta, A. Dhall, S. Pal, and S. S. Khan, “Motion and region aware adversarial learning for fall detection with thermal imaging,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021.

[9] G. Vavoulas, M. Pediaditis, E. G. Spanakis, and M. Tsiknakis, “Smartphone Human Fall Dataset.” Kaggle, 2022.