Comparative Analysis of LSTM, GRU, and Bi-LSTM+CNN Models for Elderly Fall Detection

Author: Kajal Singh (A00268744), MSc. In Software Engineering
Supervisor: Dr. Yuansong Qiao

Abstract

With the demographic trend towards an aging global population, the prevalence of falls among seniors living independently has become a pressing concern. Rapid detection and response to such incidents are crucial to prevent serious long-term consequences. This research leverages the comprehensive MobiAct dataset, accessible publicly, which captures a range of everyday activities and falls through smartphone sensors. The dataset, built upon the foundational MobiFall dataset designed for fall detection, includes diverse movements and stationary activities that mimic the aftermath of falls.

The focus of this study is to harness and compare the strengths of advanced neural network architectures for accurate fall detection. It employs Long Short-Term Memory (LSTM) networks, Bi-Directional LSTM (BiLSTM), Convolutional Neural Networks (CNN), multi-layer CNNs, and Gated Recurrent Units (GRU) to analyse and learn from the temporally rich data provided by the MobiAct dataset. Each of these models brings unique capabilities to the task: CNNs excel at extracting spatial features from sensor data, multi-CNNs amplify this capability through depth, LSTMs are adept at recognizing sequential data over time, BiLSTMs extend this with dual-directional data processing, and GRUs offer a more efficient alternative with fewer parameters and a simplified gating mechanism.

Our comprehensive evaluation demonstrates that the synergy of these models yields high accuracy in differentiating between normal activities and falls. The BiLSTM model showcases exceptional performance with an accuracy of 99.21%, while the integration of GRU algorithms presents computational efficiency advantages. The multi-layer CNN approach also reveals significant promise in enhancing the detection framework's sensitivity and specificity.

Introduction

Falls are a prevalent and critical issue, especially among the elderly and those with mobility impairments. Such incidents can lead to severe injuries, long-term disabilities, and even fatalities, alongside substantial healthcare costs. To address this, the integration of machine learning with sensor technologies has emerged as a revolutionary approach, offering real-time monitoring and instantaneous fall detection capabilities.

Our research centers around evaluating the potential of advanced machine learning models for reliable fall detection. We employ Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRUs), and a hybrid model combining LSTM with Convolutional Neural Networks (CNNs) to analyze the MobiAct dataset—a benchmark in human activity recognition. The LSTM and GRU models, with their recursive nature, are proficient in identifying temporal patterns, while the hybrid LSTM-CNN model aims to exploit both temporal and spatial data features, offering a comprehensive understanding of movement dynamics.

By systematically assessing the performance of these models against robust metrics, we aim not only to enhance fall detection accuracy but also to pave the way for preemptive fall prevention strategies. The outcomes of this study are expected to inform the design of next-generation wearable devices, contribute to the autonomy and safety of individuals at risk of falls, and potentially lead to substantial reductions in emergency healthcare interventions.

In addition to the technical advancements in fall detection systems, our research also considers the practical implementation and scalability of these technologies. We explore how these machine learning models can be seamlessly integrated into existing healthcare infrastructure and wearable devices, ensuring user-friendliness and accessibility. Furthermore, we investigate the feasibility of deploying these systems in real-world settings, considering factors such as sensor placement, data variability, and user acceptance.

Test Configuration and Method

This study was rigorously designed to evaluate the efficiency of machine learning algorithms in fall detection, using the comprehensive MobiAct dataset. The test configuration involved a multi-stage process to ensure the reliability and validity of the results.
MobiAct is a publicly available dataset (available for download from https://bmi.hmu.gr/) [1] which includes data from a smartphone when participants are performing different types of activities and a range of falls. It is based on the previously released MobiFall dataset, which was initially created with fall detection in mind. It encompasses four different types of falls and nine different ADLs from a total of 57 subjects with more than 2500 trials, all captured with a smartphone. The diversity of activities and falls in the MobiAct dataset aligns with our research aim to create a fall detection system that is sensitive and accurate across a wide range of motions and scenarios. It also provides the opportunity to assess the generalizability of our proposed models to new, unseen data, a crucial aspect of any practical application.

Code	Activity	Trials	Duration	Description
FOL	Forward-lying	3	10s	Fall Forward from standing, use of hands to dampen fall
FKL	Front-knees-lying	3	10s	Fall forward from standing, first impact on knees
SDL	Sideward lying	3	10s	Fall sideward from standing, bending legs
BSC	Back-sitting-chair	3	10s	Fall backward while trying to sit on a chair

Table 1: Falls recorded in the MobiAct dataset

These activities were included from the start of the effort, since our ultimate objective has been to extend our work towards recognition of not only falls, but also complex everyday activities and, eventually, behaviors. Moreover, the fact that such activities are included is an advantage concerning human activity recognition (HAR) in general.

Code	Activity	Trials	Duration	Description
STD	Standing	1	5m	Standing with subtle movements
WAL	Walking	1	5m	Normal walking
JOG	Jogging	3	30s	Jogging
JUM	Jumping	3	30s	Continuous jumping
STU	Stairs up	6	10s	Stairs up (10 stairs)
STN	Stairs down	6	10s	Stairs down (10 stairs)
SCH	Sit on chair	6	6s	Sitting on a chair with subtle movements
CHU	Chair up	6	6s	Transition from sitting to standing
CSI	Car step in	6	6s	Step in a car
CSO	Car step out	6	6s	Step out of a car
LYI	Lying	12	-	Activity taken from the lying period after a fall

Table 2: Activities of Daily Living recorded in the MobiAct dataset

It encompasses four different types of falls and nine different ADLs from a total of 57 subjects with more than 2500 trials, all captured with a smartphone.
As a result, MobiAct dataset is suitable for investigating both fall detection and HAR. Table 1 and Table 2 summarize all captured activities (and activity codes), their present trial counts, durations, and a short description of each activity.

The dataset acquisition for the MobiAct study was meticulously executed to capture a diverse range of Activities of Daily Living (ADLs) and various types of falls. A critical aspect of the study's efficiency was the careful documentation of the processing time required for each type of ADL and fall. This documentation provides insights into the method's responsiveness to different activities and incidents. The processing times for different ADL types and fall events have been thoroughly analysed and presented in the following figures:

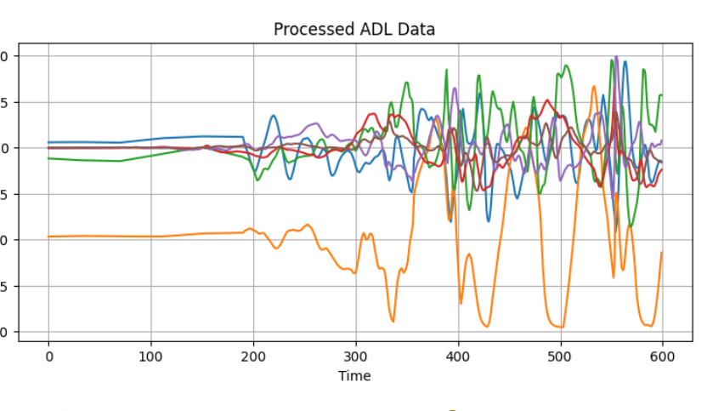


Fig 1: Processing Time for ADL Types

Figure 1 illustrates the time taken for ADL data reading and processing, with variations observed depending on the activity.
Figure 2 showcases the time consumed to process different fall events, highlighting the method's responsiveness to these critical incidents.

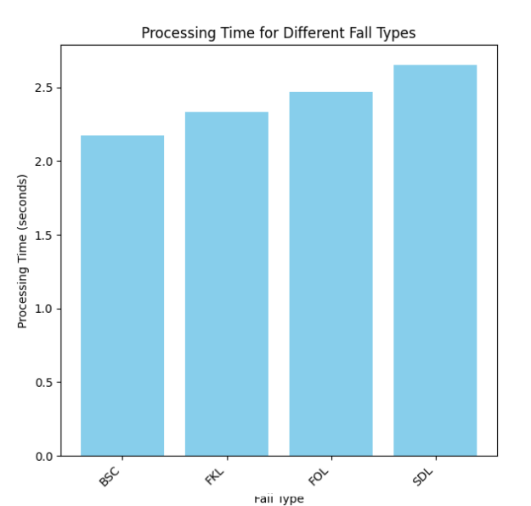


Fig 2: Processing Time for Fall Types

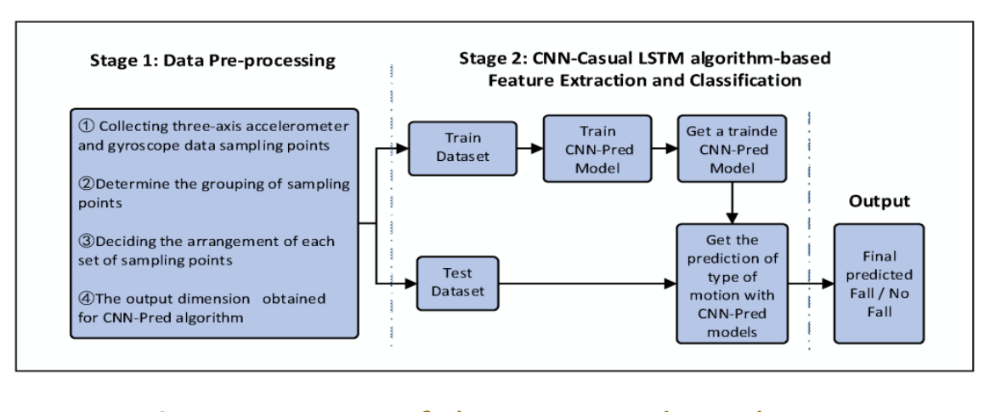

Stage 1: Data Pre-processing
① Collecting three-axis accelerometer and gyroscope data sampling points
②Determine the grouping of sampling points
③Deciding the arrangement of each set of sampling points
④The output dimension obtained for CNN-Pred algorithm
Stage 2: CNN-Casual LSTM algorithm-based Feature Extraction and Classification
Train Dataset
Train CNN-Pred Model
Get a trainde CNN-Pred Model
Test Dataset
Get the prediction of type of motion with CNN-Pred models
Output
Final predicted Fall / No Fall
Fig 3: Diagram of the Proposed Work

Figure 3 illustrates the workflow for fall detection using machine learning. It starts with preprocessing sensor data to filter noise and normalize signals. Key features are then extracted and used to train a hybrid CNN-LSTM model. Finally, the trained model classifies new data to detect falls.

Following data acquisition, the collected sensor data undergoes preprocessing to prepare it for model training. This preprocessing phase addresses challenges such as noise reduction, normalization, and feature extraction. Once the data is preprocessed, machine learning models are trained to distinguish between falls and activities of daily living (ADLs). Initially, the sensor data is visualized to understand its characteristics across different activities. Visualizations, such as plots of accelerometer and gyroscope readings during various activities, help identify patterns and variability in the data. For example, accelerometer data during walking may exhibit distinct patterns compared to jogging or fall activities.

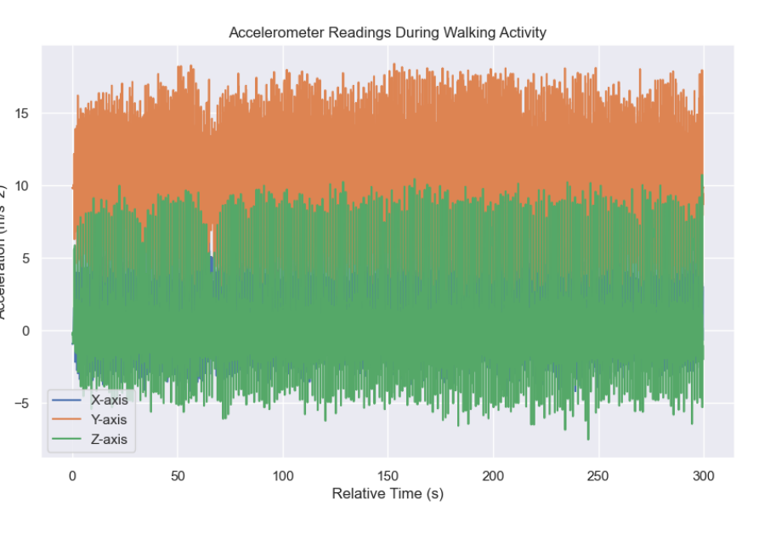


Fig 4: Accelerometer readings

Figure 4 (Accelerometer readings), **Figure 5** (Gyroscope readings), and **Figure 6** (Raw vs. filtered accelerometer data) illustrate that sensor data is initially noisy and requires preprocessing to extract meaningful patterns. For accelerometer data, a low-pass filter was applied to remove high-frequency noise. This process ensures that the inherent movement patterns indicative of a fall versus a daily activity, such as walking, are preserved and more pronounced for the learning algorithm.

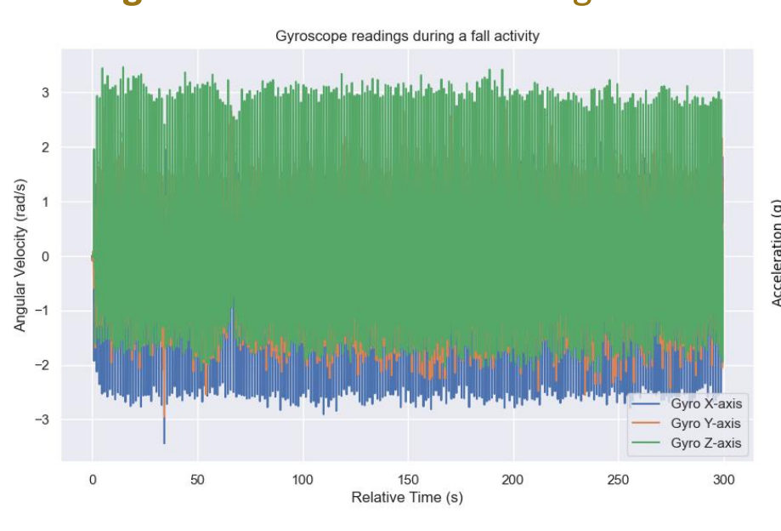


Fig 5: Gyroscope readings

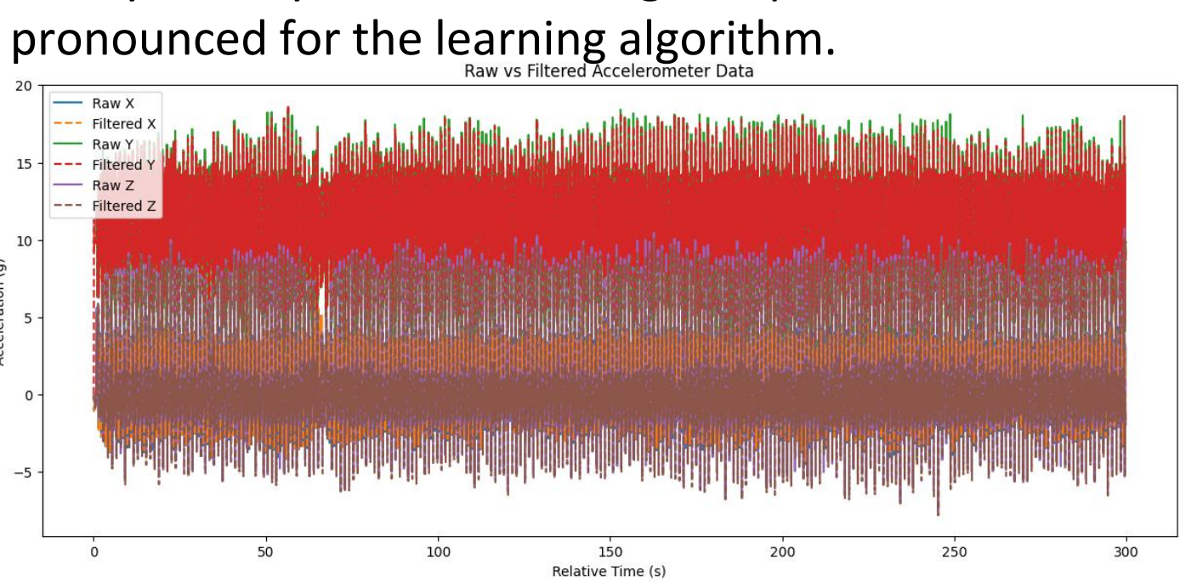
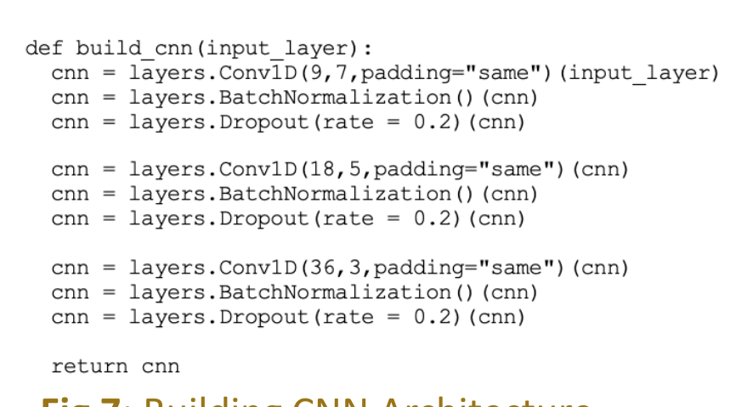
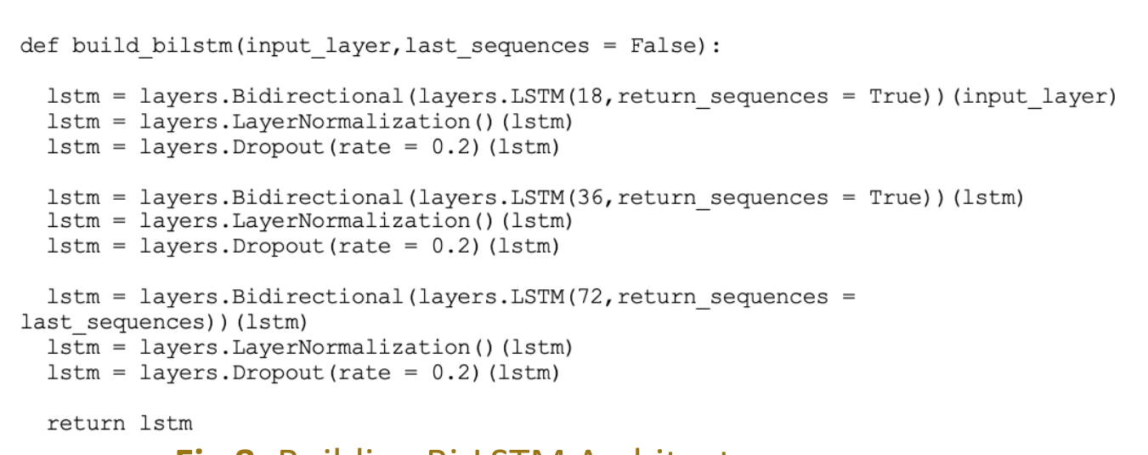


Fig 6: Comparison of raw vs. filtered accelerometer data



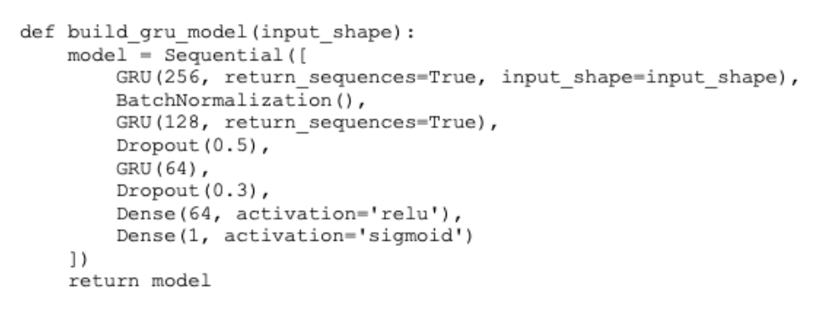
```
def build_cnn(input_layer):
  cnn = layers.Conv1D(9,7,padding="same")(input_layer)
  cnn = layers.BatchNormalization()(cnn)
  cnn = layers.Dropout(rate = 0.2)(cnn)

  cnn = layers.Conv1D(18,5,padding="same")(cnn)
  cnn = layers.BatchNormalization()(cnn)
  cnn = layers.Dropout(rate = 0.2)(cnn)

  cnn = layers.Conv1D(36,3,padding="same")(cnn)
  cnn = layers.BatchNormalization()(cnn)
  cnn = layers.Dropout(rate = 0.2)(cnn)

  return cnn
```

Fig 7: Building CNN Architecture



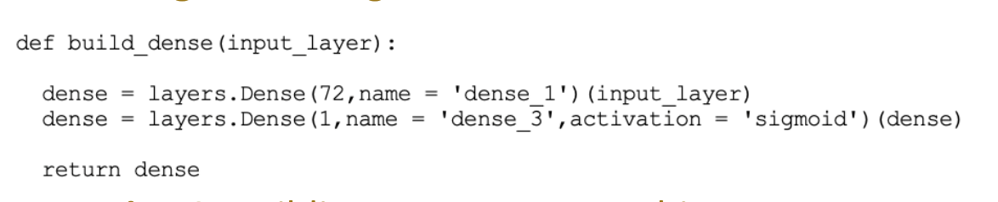
```
def build_bilstm(input_layer,last_sequences = False):

  lstm = layers.Bidirectional(layers.LSTM(18,return_sequences = True))(input_layer)
  lstm = layers.LayerNormalization()(lstm)
  lstm = layers.Dropout(rate = 0.2)(lstm)

  lstm = layers.Bidirectional(layers.LSTM(36,return_sequences = True))(lstm)
  lstm = layers.LayerNormalization()(lstm)
  lstm = layers.Dropout(rate = 0.2)(lstm)

  lstm = layers.Bidirectional(layers.LSTM(72,return_sequences =
last_sequences))(lstm)
  lstm = layers.LayerNormalization()(lstm)
  lstm = layers.Dropout(rate = 0.2)(lstm)

  return lstm
```

Fig 8: Building Bi-LSTM Architecture



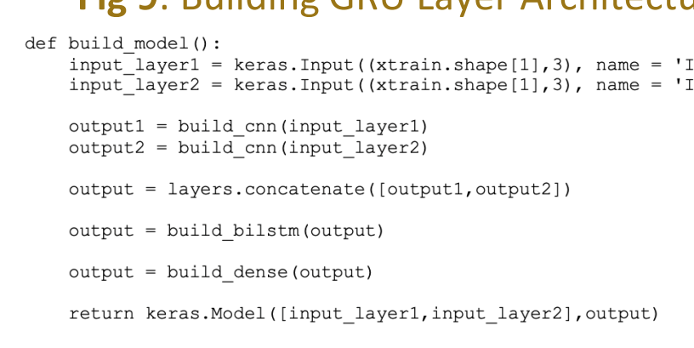
```
def build_gru_model(input_shape):
    model = Sequential([
        GRU(256, return_sequences=True, input_shape=input_shape),
        BatchNormalization(),
        GRU(128, return_sequences=True),
        Dropout(0.5),
        GRU(64),
        Dropout(0.3),
        Dense(64, activation='relu'),
        Dense(1, activation='sigmoid')
    ])
    return model
```

Fig 9: Building GRU Layer Architecture



```
def build_dense(input_layer):

  dense = layers.Dense(72,name = 'dense_1')(input_layer)
  dense = layers.Dense(1,name = 'dense_3',activation = 'sigmoid')(dense)

  return dense
```

Fig 10: Building Dense Layer Architecture



```
def build_model():
    input_layer1 = keras.Input((xtrain.shape[1],3), name = 'Input1')
    input_layer2 = keras.Input((xtrain.shape[1],3), name = 'Input2')

    output1 = build_cnn(input_layer1)
    output2 = build_cnn(input_layer2)

    output = layers.concatenate([output1,output2])

    output = build_bilstm(output)

    output = build_dense(output)

    return keras.Model([input_layer1,input_layer2],output)
```

Fig 11: Model Assembly

The fall detection model employs a multi-layered approach combining CNN for spatial feature extraction and Bi-LSTM for sequential temporal analysis, with the addition of Gated Recurrent Unit (GRU) layers offering a computationally efficient alternative for sequence processing. The model's architecture stacks CNN layers to distil input data into features, which are normalized and regularized to counter overfitting.

Bi-LSTM layers enhance sequence learning by integrating forward and backward temporal information, while GRUs simplify the learning process with fewer parameters. A final dense layer, with a sigmoid activation function, consolidates these insights to differentiate between falls and routine activities accurately.

The dataset has been partitioned into three distinct sets to facilitate the training, validation, and testing of the machine learning model. A pie chart representation showing in **Figure 12** how the MobiAct dataset was divided into training (60%), validation (20%), and testing (20%) sets.
In conclusion, test configurations and methodologies are rigorously designed to assess and enhance the predictive accuracy and efficiency of fall detection models.

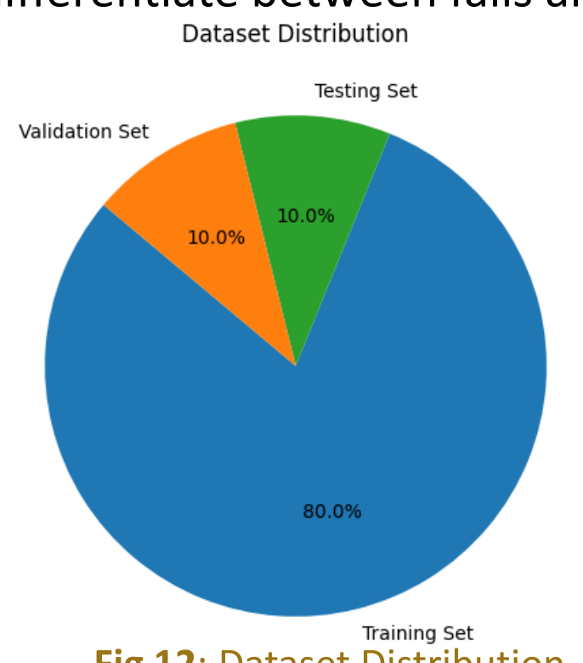


Fig 12: Dataset Distribution

Results

This section provides an in-depth analysis of the performances of three advanced machine learning models—Bi-LSTM+CNN, GRU, and LSTM—specifically tailored to the task of fall detection using the MobiAct dataset. Performance metrics including accuracy, precision, recall, F1-score, and test loss across various batch sizes are discussed to assess each model's effectiveness and efficiency. These metrics are instrumental in understanding each model's adeptness at distinguishing between fall and non-fall events, which is paramount for practical fall detection systems.
The effectiveness of the LSTM, CNN, GRU, and LSTM-CNN models was meticulously assessed based on a set of critical performance indicators: accuracy, precision, recall, and the F1-score. These metrics collectively offer a holistic view of each model's capabilities in accurately identifying fall events amidst various activities of daily living.

Model	Accuracy	Precision	Recall	F1-score	Batch Size
LSTM	99.0%	99.0%	99.0%	99.0%	32
Bi-LSTM+CNN	99.1%	99.1%	99.1%	99.1%	128, 64, 32
GRU	98.96%	99.7%	98.0%	98.85%	128, 64, 32

Table 3: Comprehensive Performance Metrics

Table 3 delineates the performance metrics for each model, reflecting their predictive accuracy and reliability in fall detection.

Model Performance of Bi-LSTM+CNN model

The hybrid Bi-LSTM+CNN model leverages both bidirectional LSTM layers to capture temporal dependencies from both forward and backward directions and CNN layers to extract spatial features from sequential sensor data. This combination is particularly suited for time-series data that requires the recognition of complex patterns over time and space.

Batch Size	Accuracy	Precision	Recall	F1-Score	Test Loss
128	0.9896	0.99	0.99	0.99	0.0395
64	0.9911	0.99	0.99	0.99	0.0526
32	0.9911	0.99	0.99	0.99	0.0443

Table 4: Bi-LSTM+CNN Model Performance Across Different Batch Sizes

The Bi-LSTM+CNN model shows high consistency in performance across different batch sizes, indicating its robustness and the effectiveness of combining LSTM and CNN architectures for enhanced feature extraction.

Batch Size 128: **Figure 13** illustrates the training and validation accuracy/loss curves for the model trained with a batch size of 128. The performance indicates that while the larger batch size allows for faster computation, it may not capture the nuances within the dataset as effectively as smaller batch sizes.This is reflected in the marginally lower test loss when compared to a batch size of 64, suggesting a slight improvement in the model's ability to predict unseen data without overfitting.

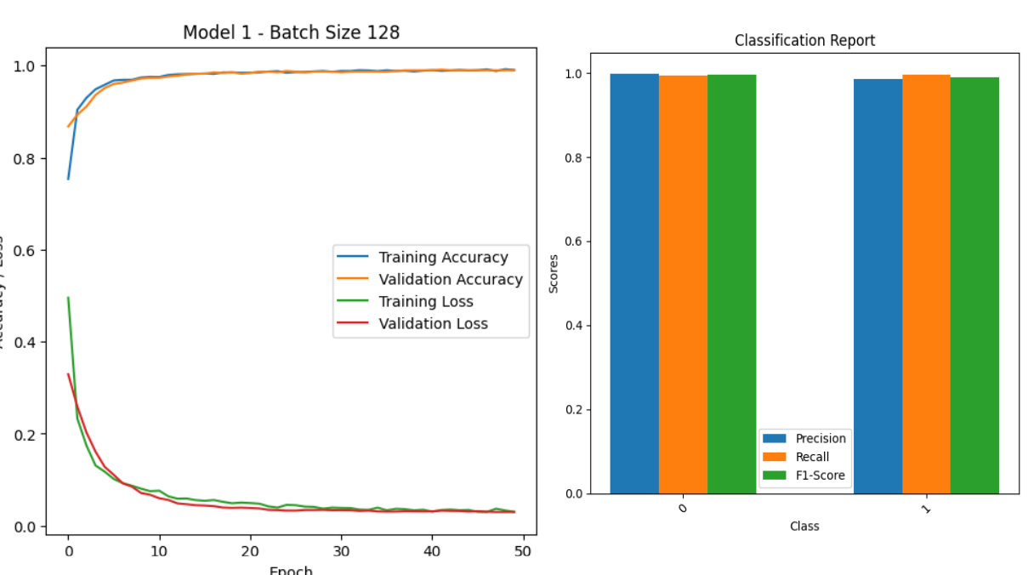


Fig 13: : Training and Validation Accuracy/Loss with Batch Size 128

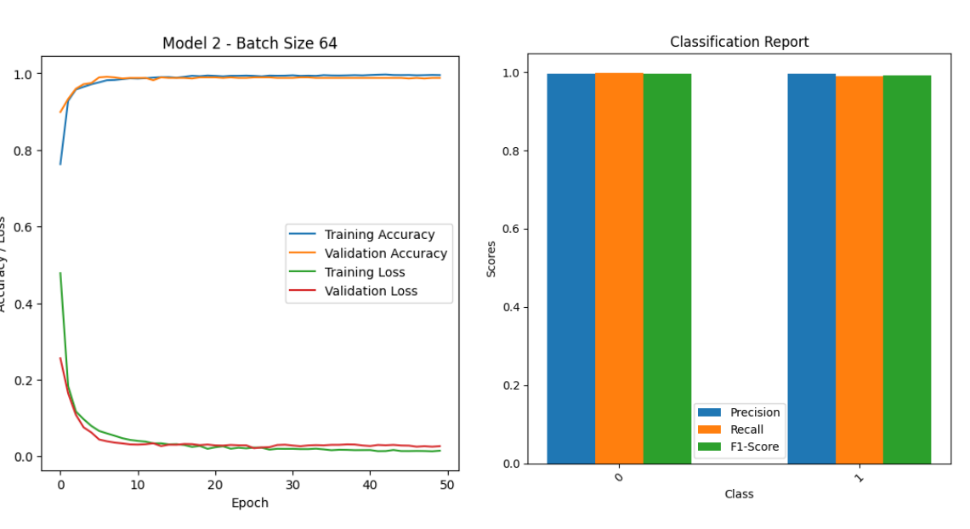


Fig 14: : Training and Validation Accuracy/Loss with Batch Size 64

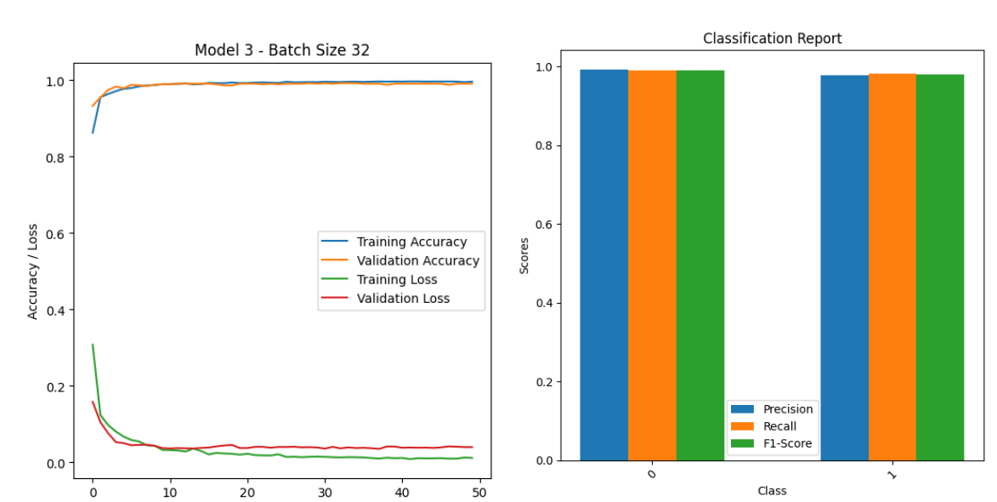


Fig 15: : Training and Validation Accuracy/Loss with Batch Size 32

Batch Size 64: Training with a batch size of 64, as shown in **Figure 14**, provided a balance between computational efficiency and model learning capability. The model achieved a notable improvement in learning the nuances of the dataset compared to the larger batch size.
The intermediate batch size of 64 strikes a balance between computational efficiency and the model's capacity to learn finer details from the training data. Here, the model has more updates per epoch than with a batch size of 128, which may contribute to a more refined fitting to the dataset. However, this comes with a slightly increased test loss, indicating a potential for overfitting or variance in the model's predictions.

Batch Size 32: As the smallest batch size used in this experiment, 32 offered a more nuanced update to the weights with each step, as can be seen in **Figure 15.**
The smallest batch size, 32, allows more frequent weight updates, which can fine-tune the model's learning but may slow down training. Despite achieving high metrics similar to larger batches, its slightly higher test loss suggests potential overfitting to training data. On the other hand, a batch size of 128 offers a balance, showing lower test loss while maintaining high performance, indicating effective learning without over-specializing on the training set.

Model Performance of GRU model

The GRU model simplifies the recurrent neural network architecture but maintains robustness, especially in learning long-term dependencies with fewer parameters than traditional LSTMs. This efficiency is crucial for fall detection, where timely and accurate responses are essential.

This model demonstrates excellent precision and recall, particularly at smaller batch sizes, reflecting its capacity to effectively classify fall events with minimal false positives and missed true fall events.

Batch Size	Accuracy	Precision	Recall	F1-Score	Test Loss
128	0.9629	0.97	0.95	0.96	0.0995
64	0.9733	0.97	0.97	0.97	0.0813
32	0.9896	0.99	0.98	0.99	0.0298

Table 5: GRU Model Performance Across Different Batch Sizes

Figure 16 displays the training and validation accuracy and loss for three different batch sizes over 50 epochs for a model.

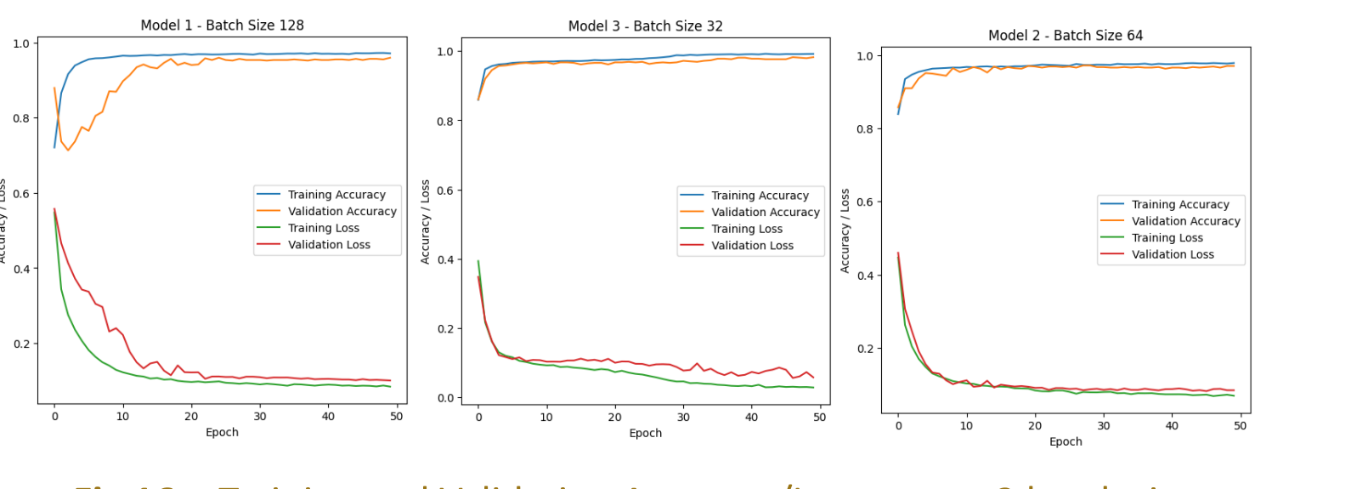


Fig 16: : Training and Validation Accuracy/Loss across 3 batch sizes

Batch Size 128 (Model 1): Shows quick learning and stability as epochs progress, providing a good balance between learning efficiency and computational speed.
Batch Size 32 (Model 3): Reveals more refined learning, with better validation loss over time, which may indicate superior generalization but at the cost of longer training.
Batch Size 64 (Model 2): Offers a middle ground with effective learning and moderate training time, representing a practical choice for both performance and speed.
In summary, each batch size has its merits, with Batch Size 64 presenting an optimal balance for both learning and computational resource management.

Model Performance of LSTM model

The LSTM (Long Short-Term Memory) network tested with a batch size of 32 demonstrates a notable proficiency in fall detection, achieving a high recall rate of 98.91%. This suggests that the model is highly capable of identifying true fall events, minimizing the risk of missed incidents which could have serious safety implications.

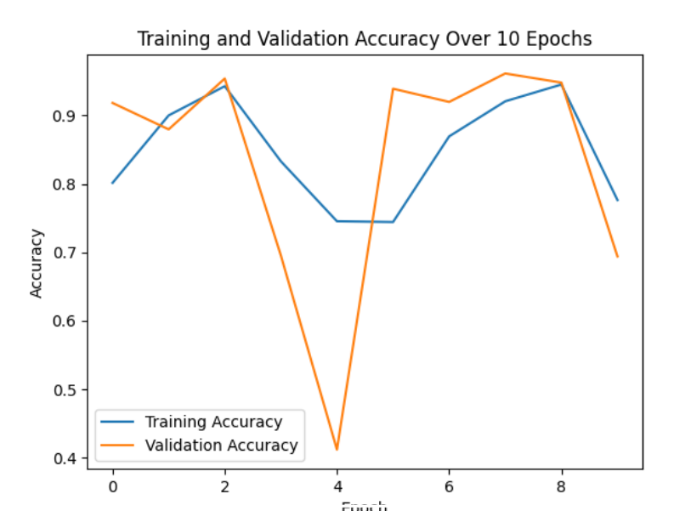


Fig 17: : Training and Validation Accuracy/Loss

Batch Size	Accuracy	Precision	Recall	F1-Score	Test Loss
32	0.9614	0.8829	0.9891	0.9330	0.1167

Table 5: LSTM Model Performance Metrics

The training dynamics of our LSTM model, depicted over 10 epochs, reveal a robust initial learning phase with typical fluctuations in accuracy.
Notably, a significant dip and subsequent recovery around epoch 6 highlight the model's adaptive learning capabilities. The convergence of training and validation accuracies towards the later epochs suggests improving generalization, a critical factor for the practical deployment of the model.

The models we evaluated — LSTM, Bi-LSTM+CNN, and GRU — each demonstrated impressive performance in the task of fall detection. Across various batch sizes, each model maintained high levels of accuracy, precision, recall, and F1-score, as shown in the accompanying tables and figures. Notably, the Bi-LSTM+CNN model achieved remarkable consistency regardless of batch size, underscoring the combined power of LSTM's temporal processing and CNN's spatial feature extraction.

Conclusion and Future Work

The research undertaken meticulously evaluated the effectiveness of LSTM, GRU, and Bi-LSTM+CNN models in the context of elderly fall detection using the comprehensive MobiAct dataset. The findings confirm that all models exhibit remarkable predictive accuracy, with the Bi-LSTM+CNN model demonstrating slight superiority in performance metrics. The GRU model, on the other hand, offers significant computational advantages due to its simplified architecture. The LSTM model's high recall rate underscores its potential in scenarios where the cost of missing a fall is critically high. Overall, each model has its strengths and the choice between them would be dictated by the specific requirements of the deployment environment, considering the trade-offs between accuracy, computational efficiency, and response time.

Future work aims to enhance accuracy and reduce false positives through data augmentation, hyperparameter optimization, real-time deployment, wearable device integration, exploring advanced deep learning techniques, and interdisciplinary collaboration with healthcare professionals. These efforts could yield a more reliable fall detection system for eldercare and independent living.

By addressing these areas, the research could lead to the development of an even more effective and reliable fall detection system that could be readily implemented in eldercare facilities and for personal use by seniors living independently.

References

1. K. Vrotsou et al., "The MobiFall and MobiAct Datasets," Human Movement Science, 2015.
2. Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville. MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/
3. L. Z. Rubenstein, "Falls in older people: epidemiology, risk factors and strategies for prevention," Age and Ageing, vol. 35, suppl. 2, pp. ii37-ii41, 2006. DOI: 10.1093/ageing/afl084
4. M. M. Hassan et al., "A smartphone-enabled fall detection framework for elderly people in connected home healthcare," IEEE Network, vol. 33, no. 6, 2019. DOI: 10.1109/MNET.008.1800421