Received 2 June 2023, accepted 20 June 2023, date of publication 26 June 2023, date of current version 29 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3289402

RESEARCH ARTICLE

Secure Your Steps: A Class-Based Ensemble Framework for Real-Time Fall Detection Using Deep Neural Networks

MD. MOHSIN KABIR[1], JUNGPIL SHIN[2], (Senior Member, IEEE), AND M. F. MRIDHA[3], (Senior Member, IEEE)

[1]Superior Polytechnic School, Universitat de Girona, 17071 Girona, Spain
[2]School of Computer Science and Engineering, The University of Aizu, Aizuwakamatsu 965-8580, Japan
[3]Department of Computer Science and Engineering, American International University-Bangladesh, Dhaka 1229, Bangladesh

Corresponding author: Jungpil Shin (jpshin@u-aizu.ac.jp)

This work was supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP23H03477.

ABSTRACT Falls represent a significant public health concern, particularly concerning vulnerable populations such as older adults. Accurate detection and classification of falls are critical for timely interventions that can prevent injuries and enhance the quality of life of these individuals. This work proposes a class ensemble approach based on convolutional neural networks and long short-term memory networks for three-class classifications of falling processes (non-fall, pre-fall, and fall) using accelerometer and gyroscope data. The research is conducted on the SisFall and UMAFall datasets, the publicly available dataset of annotated video recordings of falls and non-falls. This approach leverages convolutional neural networks for robust feature extraction from the accelerometer and gyroscope data. In addition, long short-term memory networks model the falling process's temporal dynamics. The proposed approach has demonstrated state-of-the-art performance in detecting falls, with accuracy rates of 96.45% and 96.12% and precision scores of 98.12% and 97.45% in identifying pre-fall and fall states, respectively.

INDEX TERMS Convolutional neural network, deep neural network, ensemble method, fall detection, long short-term memory networks.

I. INTRODUCTION

Falls are a significant health crisis for older adults as they can cause severe injuries, hospitalization, disability, or even death. As per the World Health Organization, falls are the second primary causality of unexpected or unwilling injury-related fatalities globally, with adults aged 65 years and above standing out as the most vulnerable segment of the population [1]. Implementing systems for fall detection can play an instrumental role in mitigating the occurrence of falls, thereby fostering the welfare and autonomy of older adults [2], [3].

Fall detection systems (FDS) have been developed to address this issue, with the two primary categories being context-aware (CAS) and non-context-aware systems (non-CAS) [4], [5], [6]. CAS systems employ sensors in the surroundings, such as cameras, pressure sensors, or microphones. Still, their installation is limited and unsuitable for sparsely populated areas or people frequently leaving the house. Wearable FDS, a non-CAS FDS, uses low-power sensors like accelerometers, gyroscopes, and magnetometers, making them ideal for hospitals and other scenarios where continuous monitoring is needed.

Although extensive research has been undertaken to devise wearable devices and context-aware algorithms for post-fall detection, researchers have now begun focusing on developing pre-fall detection systems that rely on fall risk evaluation and intervention [7]. Usually, Post-fall detection involves detecting a fall event after it has already occurred. This approach has some limitations, including the possibility of delayed intervention and the potential for false negatives. In other words, if the post-fall detection system fails to

The associate editor coordinating the review of this manuscript and approving it for publication was Yudong Zhang.

detect a fall event, the person may not receive timely medical attention or assistance, which could lead to further injuries or complications. Hence, Pre-impact fall prediction, which involves predicting falls before they occur, has emerged as a promising approach to overcome the constraints of post-fall detection and facilitate long-term fall risk estimation. Predicting falls before impact presents a formidable challenge due to the brief falling interval and the diverse range of fall types. Pre-impact fall prediction analyzes various risk factors, such as gait abnormalities, balance impairment, and environmental hazards, to assess a person's fall risk. This information can be used to develop personalized fall detection strategies and interventions to reduce the risk of falls.

In the wearable FDS, two distinct approaches are used to forecast pre-fall and post-fall states: threshold-based and machine learning-based algorithms [8]. Threshold-based strategies exhibit lower efficacy levels as they are susceptible to the types of falls involved. At the same time, machine learning-based techniques demand a significant number of samples per instance for accurate fall prediction [9]. Recent advancements in sensor technology, machine learning, and artificial intelligence have enabled the evolution of various advanced FDS [10], [11]. Among these, deep learning-based systems have shown promising results in detecting pre-fall, fall, and non-fall events [3], [6]. Convolutional neural networks (CNN) and long short-term memory (LSTM) models have been vastly used in fall detection systems to capture spatial and temporal features from sensor data [12], [13], [14].

However, the performance of Deep Learning (DL) models affects by diverse characteristics, such as sensor placement, data quality, and class imbalance [15]. Ensemble models have been suggested to address these challenges to enhance the robustness and accuracy of fall detection approaches [16]. Ensemble models combine multiple base models to leverage their strengths and minimize their weaknesses, resulting in a more accurate and stable prediction [17].

Consequently, the principal aim of this investigation is to devise a novel deep learning architecture founded on class ensembling, utilizing accelerometer and gyroscope data to accurately identify fall events in real-time, thereby providing a preemptive safety mechanism to minimize the occurrence of severe injuries before impact and issuing a remote notification to facilitate prompt medical intervention. The efficacy of the proposed architecture is assessed on the SisFall [18] and UMAFall [19] datasets and compared against state-of-the-art fall detection algorithms to evaluate its performance. By developing an effective fall detection system, we can proactively identify vulnerable populations with high fall risks, thereby implementing appropriate measures to diminish the likelihood of future falls. The major contributions are as follows:

- Developing a class ensemble-based deep learning architecture integrated with CNN and LSTM.
- Evaluating the presented architecture on the SisFall and UMAFall dataset using accelerometer and gyroscope data.
- Demonstrating the effectiveness of the proposed architecture over state-of-the-art fall detection systems.

The remainder of this article is structured as follows. Section II presents a comprehensive review of related research in the FDS field. Section III provides a detailed exposition of the proposed class-based ensemble FDS, encompassing the datasets, data preprocessing, and model architecture. The experimental outcomes are presented and examined in Section IV, including experimental setup and evaluation metrics. Additionally, Section V delineates the contributions and limitations of the proposed system while suggesting potential avenues for future research. Finally, Section VI ends this article.

II. RELATED WORK

Over the past few years, a burgeoning interest has been in adopting machine learning techniques, particularly deep learning, for fall detection. Various studies have investigated the feasibility of applying these techniques in real-world scenarios, including the use of wearable devices and associated algorithms to collect and analyze gait data. Numerous fall detection solutions have been proposed, broadly classified into three categories: wearable, ambient, and vision-based [28].

A. CATEGORIES OF FDS

Ambient fall detection refers to using sensors and devices installed in the environment to detect falls [29]. These devices can be placed in strategic locations such as walls, ceilings, and floors, and they can detect changes in movement patterns or the absence of movement to determine if a fall has occurred. There are several types of ambient fall detection systems, including those that use infrared sensors, pressure sensors, and cameras. Infrared sensors can detect temperature changes, indicating a person's presence or absence in a particular area. Pressure sensors can detect weight distribution changes, indicating a person's fall. Cameras can be used to track a person's movements and detect falls based on changes in their body position.

One of the advantages of ambient fall detection systems is that they do not require the user to wear any special equipment, which can be helpful for people who are resistant to wearing wearable devices. Additionally, ambient systems can be more effective at detecting falls in certain situations, such as when the person is unconscious or unable to press a panic button. However, ambient fall detection systems also have some limitations. For example, they may be less accurate than wearable systems in detecting falls, as they can be affected by other environmental movements, such as the movement of pets or objects. Additionally, they may not be able to detect falls in all situations, such as when a person falls out of view of the sensors or when the fall is not severe enough to trigger the sensors.

Furthermore, Vision-based fall detection methods utilize cameras and computer vision techniques to monitor and

TABLE 1. Comparative performance and limitations of existing fall detection methods.

Reference	Year	Method	Dataset	Performance	Limitation
Maray et al. [20]	2023	Transfer learning.	MSBAND	F1 Score 92%	The SmartFall system had limitations related to the limited availability of personalized training data and the occurrence of model drift when applied on different devices.
Sheikh et al. [21]	2023	OCSVM, ZART	Sis-Fall	Accuracy 96%.	The limitation of this study is a detailed analysis on sensitivity, specificity, precision, and recall metrics was not conducted, which could provide a more comprehensive evaluation of the proposed method's performance.
Wu et al. [22]	2022	GRU	MobiAct & R Fall Detection	Accuracy 99.56% and 90.69% F1 Scores 96.83% and 87.29%	The limitations of this study is a detailed analysis on sensitivity, specificity, precision, and recall metrics was not conducted, which could provide a more comprehensive evaluation of the proposed method's performance.
Li et al. [23]	2020	LSTM and Bi-LSTM	Collected data from (FMCW) radar (UWB) pulse radars	Bi-LSTM classifier achieved an accuracy of 93% NBC decision fusion method achieved an accuracy of 90%.	The proposed approach has a limitation in that it relies on radar sensors, which may not be suitable for environments where radar cannot be utilized.
Lin et al. [24]	2020	LSTM, GRU, TS-RNN)	570 video from different rooms and view angles	Framework accuracy 98.2% Outperformed accuracy 88.9% with a 9.3% improvement	A limitation of the proposed fall detection framework is its evaluation solely on single-view videos, which may not fully represent real-world scenarios that often involve multiple views and angles.
Li et al. [25]	2019	bi-LSTM	FMCW	Accuracy of approximately 96% worst-case accuracy up to 16.2%	The limitations of this paper encompass a small sample size, a limited range of activities, a restricted set of sensors used for data collection.
He et al. [26]	2019	FD-CNN	SisFall and MobiFall	Accuracy of this method is 98.61%	The paper lacks discussion on the real-world testing and evaluation of the fall detection technology, which could offer valuable insights into its practical feasibility and performance in real-life scenarios.
Zerrouki et al. [27]	2016	EWMA-based SVM classifier	UniMiB-SHAR, URFD, WISDM	Sensitivity of 96.8% Specificity of 96.9% F1 Score of 97.8%	In real-world fall detection scenarios, the practicality and feasibility of utilizing synchronized accelerometric and camera data, which the proposed method relies on, may vary and may not always be achievable.

detect falls [30]. These methods typically involve analyzing video feeds to detect changes in posture or motion patterns that indicate a fall event. One approach is to use 3D skeletal modeling to track the movement of a person's joints in real-time and detect anomalies in the motion pattern [31]. This method requires depth sensors such as Microsoft Kinect or Intel RealSense cameras to capture depth information, which is then processed using machine learning algorithms. Another vision-based approach is to use computer vision techniques such as object detection, tracking, and motion analysis to identify and track human figures in a video stream [32], [33]. Changes in the figure's position or posture can then be analyzed to detect falls.

Vision-based fall detection has the advantage of being non-invasive and does not require the use of any wearable devices. However, it can be affected by factors such as lighting conditions, occlusions, and the presence of multiple people in the frame. Additionally, privacy concerns may arise as cameras are typically required in private areas such as bedrooms and bathrooms.

In addition, Wearable fall detection systems are designed to be worn by the user and provide continuous monitoring of their movements to detect falls [34]. These systems typically have sensors measuring acceleration, orientation, and other gait-related parameters. Algorithms are then used to analyze this data and detect when a fall has occurred. Several studies have investigated the effectiveness of wearable fall detection systems, with some achieving high accuracy rates in detecting falls [35], [36], [37], [38]. However, the accuracy of these systems can be affected by factors such as the placement of the sensors on the body and the type of activity the user performs.

Recent studies have shown that wearable systems are more cost-effective and perform better than other fall detection systems, such as ambient and vision-based systems [28]. Wearables have also been found to offer better privacy and user acceptance compared to other types of systems [39], [40]. Therefore, wearable systems appear to be more convenient among the three types of fall detection systems due to their cost-effectiveness, better performance, and privacy concerns. Hence, this paper focuses on developing a wearable fall detection system.

B. RECENT DEVELOPMENTS

Zerrouki et al. [41] reviewed state-of-the-art fall detection technologies using machine learning methodologies, both classical and deep learning. The authors discuss the sensors, cameras, pre-treatments, attributes, and algorithms used in this field. The discussion highlights the limitations of fall detection, such as the lack of real data, the heterogeneity in falls, and the dependency on collected data. The authors suggest modeling like-fall activities as a separate class to reduce false detections. The paper also points out the imbalanced data and issues with performance evaluation criteria. The review aims to improve the quality of fall detection and reduce its impacts.

In a recent publication, Sheikh et al. [21] proposed a lightweight and inexpensive inertial sensing method for identifying falls among individuals using wheelchairs. The approach utilizes an unsupervised One-Class Support Vector Machine (SVM) and a hybrid scheme to achieve fall detection. The method employs a novel hybrid multi-sensor fusion strategy to correct sensor integral errors and is tested on a heterogeneous dataset, including unassisted transfers from wheelchairs. The proposed method achieves a fall detection accuracy of up to 96%, which surpasses the performance of other one-class learning approaches and threshold-based methods previously reported in the literature.

Maray et al. [20] conducted a study on a smartwatch-based fall detection system aimed at elderly individuals, which they evaluated on nine participants. The system was found to have two primary limitations: its incapability to gather a significant quantity of personalized training data and model drift when utilized on different devices. To overcome these challenges, the authors assembled three datasets from different devices. They applied transfer learning to address the problem of small dataset training and enable the model to generalize across heterogeneous devices. Their findings showed that transfer learning considerably enhanced the fall detection performance, as evidenced by superior F1_scores and AUC (Area Under the Curve) and lower false positive rates compared to the non-transfer learning strategy across different datasets accumulated utilizing various devices with various hardware specifications.

Csengul et al. [42] propose a smartwatch-based FDS that can differentiate between various activities, including falling, sitting, squatting, running, and walking. The system collects acceleration and gyroscope sensor data through a mobile app. It uses a deep learning algorithm based on the bi-directional long short-term memory (BiLSTM) neural network to classify the data in the cloud. The system achieves high accuracy rates of 97.35% for detecting falls with all activities considered and perfect accuracy for binary classification (falling vs. all other activities). The system uses Bica Cubic Hermite interpolation to boost the data available for training, and 38 statistical data features are used as input to the classifier, which is calculated using a rolling update approach.

Wu et al. [22] present a model based on Gated Recurrent Units (GRU) for automatic feature extraction in FDS. The study evaluates the proposed approach against six traditional machine learning-based classifiers and three other deep learning approaches using two publicly available datasets gathered from mobile sensors. The results demonstrate that the suggested model outperforms the other methods by achieving accuracies of 90.69%.

Lin et al. [24] address the issue of the high incidence of accidental falls among elderly individuals in Taiwan and propose an FDS that does not need the use of sensors. Instead, the proposed system employs computer vision and machine learning algorithms. The system uses OpenPose, a real-time multi-person 2D pose estimation method, to detect human activity by identifying changes in joint point locations. The system effectively filters ambient environmental noise to improve accuracy. The paper uses single-view images and experiments with time series recurrent neural networks, LSTM, and GRU architectures to continuously capture the differences in joint human points to reduce equipment costs. Empirical outcomes demonstrate that the proposed model outperforms the baseline, achieving a fall detection accuracy of 98.2% with a 9.3% improvement. This system delivers a cost-effective and accurate solution for fall detection without requiring users to wear sensors.

Li et al. [25] proposed a multi-layer bi-LSTM network framework that integrates wearable sensors and radar data for detecting daily activity patterns and high-risk events such as falls. The framework uses soft feature fusion, two robust hard-fusion methods, and a hybrid fusion approach to achieve approximately 96% accuracy in identifying continuous activities and fall events. The proposed approach is validated through a "leaving one participant out" method [23], [25]. It shows that the hybrid-fusion approach stabilizes the classification outcome among various participants, reducing accuracy variance and increasing worst-case accuracy. The study highlights the potential of multimodal sensor fusion for reliable and accurate fall detection.

Want et al. [43] proposed an FDS that utilizes multiple sensors and considers fall direction a multi-class problem to enhance the performance. The approach employs wearable sensors and a Multi-source CNN Ensemble (MCNNE) structure to extract features more efficiently. The data from various sensors are preprocessed and combined to create a comprehensive feature map. Compared to single CNN structures and different ensemble bi-model networks, MCNNE performs more promisingly. This approach offers a reliable and precise fall detection system that employs multiple sensors and a robust feature extraction technique.

TABLE 2. The table provides information on the two datasets used in the study, namely D1 (SisFall) and D2 (UMAFall), as well as the combined dataset D3 (D1+D2).

Dataset	Authors	Number of Subjects (Female/Male)	Scenario of the Experiment	Number of Samples (ADLs/Falls)	Number of Sensing Points	Type of Sensor	Sampling Rate (Hz)
D1 (SisFall)	Sucerquia et al.	38 (19/19)	Gym Hall	4505 (2707/1798)	1	1 external sensing mote with two accelerometers and a gyroscope	200
D2 (UMAFall)	Casilari et al.	17 (7/10)	Home environment	531 (322/209)	5	1 Smartphone and 4 external IMUs	100
D3 (D1+D2)	—	55 (26/29)	Both	5036 (3029/2007)	Both	Both	100

He et al. [26] introduced an FDS that addresses the limitations of Bluetooth-based wearable technology by integrating a sensor board with low-power ZigBee and MPU6050. This sensor board samples and caches three-axial acceleration and angular velocity data in sleep mode, which is transmitted to the server via ZigBee with low power consumption. The data is then normalized, cached in a sliding window, and mapped into an RGB bitmap, which is used to train a fall detection CNN to identify falls from the movements of everyday living. The method achieves an average accuracy of 98.61%, a sensitivity of 98.62%, and a specificity of 99.80%, making it appropriate for fall sensing in elderly residents with high accuracy and low power consumption.

Li et al. [44] proposed a novel deep learning model, TCN-GRU, for fall detection in elderly individuals using inertial sensors. The model outperformed other algorithms in nearly all four performance metrics examined for two open-source datasets, achieving prediction accuracy of 99.5% and 97.6% and F1_score of 98.9% and 97.6%, respectively. Despite a small data volume, the proposed model had higher detection accuracy and correctly detected all types of fall events from ten primary daily activity groups. The paper suggests that the proposed TCN-GRU model can be used for real-time, automatic fall detection using the IoT.

Zerrouki et al. [27] presented a fall detection strategy that uses accelerometric data and variations in human silhouette shape. The exponentially weighted moving average (EWMA) monitoring scheme is used to detect potential falls, and the support vector machine (SVM) algorithm is used to distinguish true falls from fall-like events based on features corresponding to detected falls. The approach detects and classifies falls accurately, with the EWMA-SVM approach proving superior to other commonly used classifiers.

Furthermore, Table 1 provides a summary of the above-mentioned fall detection methods along with their limitations. The remainder of the paper will focus on our proposed approach's performance and computational complexity and provide a theoretical basis for why it should outperform existing fall detection algorithms. The performance will be evaluated using standard metrics such as accuracy, precision, recall sensitivity, and specificity and compared to state-of-the-art approaches on the same datasets. Furthermore, we will provide a theoretical justification for why our approach should perform better and be more computationally efficient, based on using convolutional neural networks for feature extraction and long short-term memory networks for modeling temporal dynamics.

III. METHODS AND MATERIALS

In this section, we explain the data pre-processing techniques and the methodology employed in this research, which includes the proposed framework, baseline architectures, and algorithm.

A. DATASET DESCRIPTION

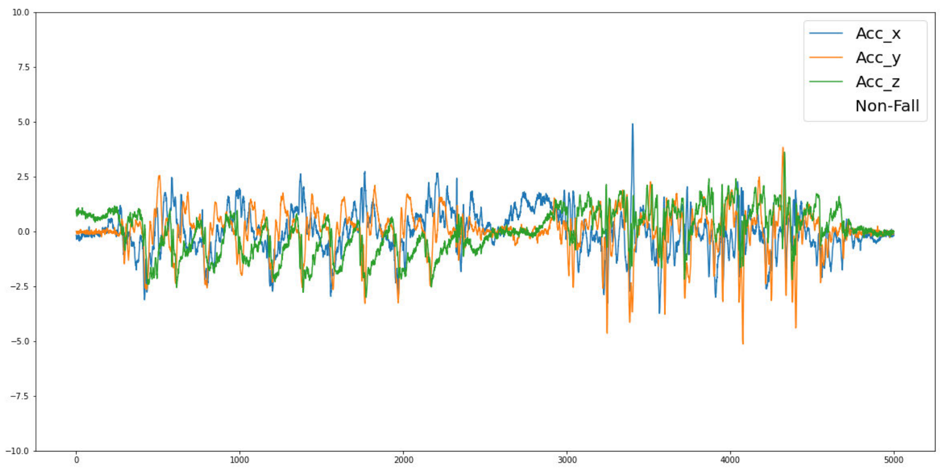
The SisFall and UMAFall datasets were used in this research to evaluate the proposed approach for fall detection. Both the SisFall and UMAFall datasets consist of annotated video recordings of emulated falls and non-fall events [45].

The SisFall dataset utilized in this study is publicly available. It comprises annotated video recordings of falls and non-falls collected from participants of various ages, including young and elderly individuals. The dataset features recordings captured within the Gym Hall environment. Data is collected using an external sensing mote fitted with two accelerometers (A1 & A2) and a gyroscope positioned at the participant's waist. The video data is synchronized with the sensor information, allowing for accurate labeling of the recorded events. The dataset contains 4505 fall and non-fall events, providing a comprehensive range of data to inform the development of the proposed ensemble-based deep neural network approach.

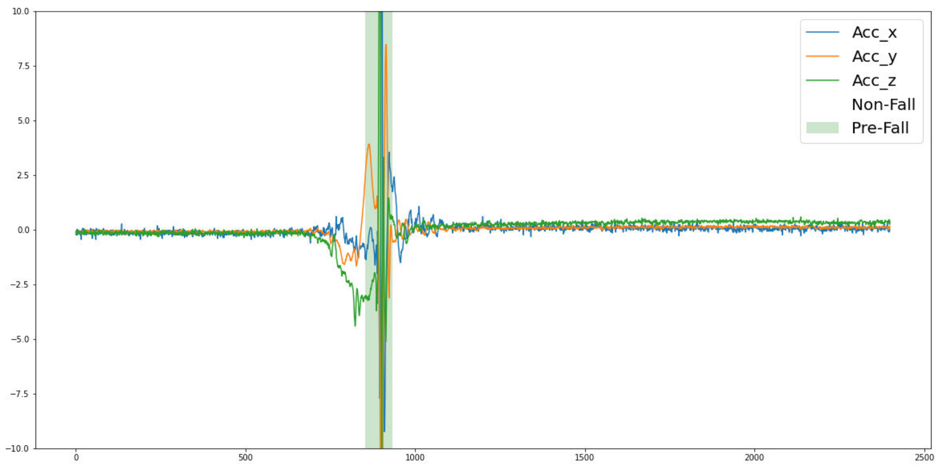
The UMAFall dataset utilized in this study is a publicly available resource featuring annotated video recordings of falls and non-falls collected from elderly participants. The dataset encompasses recordings captured within a Home environment. It includes accelerometer and gyroscope data collected via a smartphone and four external Inertial Measurement Units (IMUs) from five distinct sensing points, including the Ankle, Chest, Thigh (right trouser pocket), Waist, and Wrist. The dataset comprises 531 fall and non-fall events, providing a diverse range of data to facilitate the evaluation of the proposed ensemble-based deep neural network approach.

To evaluate the effectiveness of the proposed model, we conducted testing on both individual datasets separately, as well as on a mixed dataset comprising data from both the SisFall and UMAFall datasets. Table 2 presents a detailed explanation of the structure and characteristics of the utilized datasets.

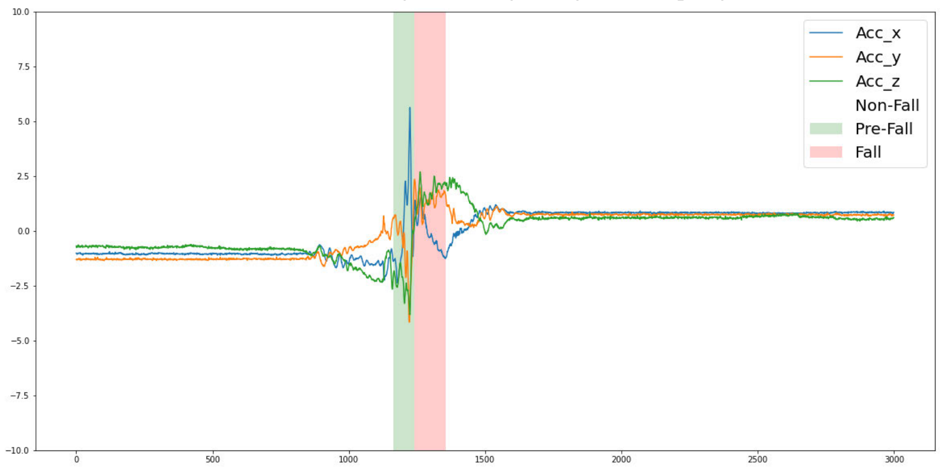
Furthermore, Figure 1 presents a schematic diagram of annotations that displays the 3-axis acceleration data of various samples.



(a) 3-axis accelerometer signals during walking upstairs at a slow pace.



(b) 3-axis accelerometer signals during sitting and collapsing down.



(c) 3-axis accelerometer signals during falling forward while jogging/tripping.

FIGURE 1. The figure depicts the annotation of accelerometer sensor data for various states: (a) represents a non-fall state where no fall occurred; (b) represents a combination of non-fall and pre-fall states where the fall is initiated but not yet impacted on the ground; and (c) represents a combination of non-fall, pre-fall, and fall states where the fall is initiated and impacts on the ground.

B. DATA PRE-PROCESSING

The SisFall and UMAFall dataset was used for this study, which contains accelerometer and gyroscope data along with labels for falls and activities of daily living (ADLs). The raw sensor data was preprocessed to extract relevant features and prepare it for use in the proposed model.

First, the data were resampled to a fixed sample rate of 100 Hz to ensure consistency across all samples. The data was then segmented into fixed-length windows of 3 seconds, with an overlap of 50% between adjacent windows.

Next, the data were normalized by subtracting the mean and dividing by the standard deviation for each sensor's data. This normalization step ensures that the data has zero mean and unit variance, which is essential for training deep learning models. The windows were converted into 3D tensors with dimensions (samples, timesteps, and features) to prepare the data for use in the proposed model.

Finally, the data was split into training, validation, and test sets using a 70:15:15 ratio respectively. This ratio has been widely used in previous studies on fall detection and classification using accelerometer and gyroscope data, making it a reasonable choice for our study [46], [47], [48]. In addition, as the sensor data contains a time component, it is crucial to handle the data split to preserve the temporal order of the sensor readings, ensuring the generalizability of the proposed approach [49]. Thus we applied a stratified splitting strategy that maintains the temporal order of the data samples while ensuring a balanced representation of the three falling process classes (non-fall, pre-fall, and fall) in each subset. To elaborate on the data split process, we first randomly shuffled the dataset while keeping the original order of consecutive samples from the same individual intact. This shuffling was performed to prevent any bias that might arise from the dataset's initial ordering. Subsequently, the stratified splitting was carried out to ensure that each subset maintains the temporal sequence of the sensor readings while containing a proportional representation of the falling process classes. The training set was utilized for training the parameters of the class ensemble framework, allowing the model to learn from a diverse range of falling process instances. The validation set was employed for hyperparameter tuning and model selection, enabling us to optimize the performance of the framework. Finally, the testing set, which was completely disjointed from the training and validation sets, was used to final evaluate the framework's performance. By adopting this stratified splitting strategy, we aim to mimic real-world scenarios where the model needs to process incoming sensor data in real time while considering the temporal dynamics of the falling process. This approach ensures that our evaluation reflects the framework's ability to generalize to unseen data and effectively detect falls.

The labels for each window were also one-hot encoded to enable multiclass classification of the falling processes (non-fall, pre-fall, and fall). The algorithm 1, presented below, illustrates the various techniques used to prepare the data for the proposed deep learning model.

The algorithm consists of a main function *PreprocessData*, and a helper function *NormalizeData*. The main function takes in raw sensor data X and returns normalized and segmented data for use in a deep learning model. The *NormalizeData* function is a helper function that takes in segmented data X and returns the normalized data. It computes each sensor channel's mean and standard deviation and then applies the normalization formula to each data point.

Algorithm 1 Data Preprocessing

```
Require: Raw sensor data from SisFall and UMAFall dataset
Ensure: Normalized and segmented data for use in the deep learning model
    function PreprocessData(X)
        $X_{resample} \leftarrow$ ResampleData($X$, 100 Hz)
        $X_{segments} \leftarrow$ SegmentData($X_{resample}$, 3 sec, 50% overlap)
        $X_{norm} \leftarrow$ NormalizeData($X_{segments}$)
        $X_{tensors} \leftarrow$ ConvertToTensors($X_{norm}$, 300 timesteps)
        $X_{train}, X_{val}, X_{test}, y_{train}, y_{val}, y_{test} \leftarrow$ TrainValTestSplit($X_{tensors}$, $y$, 70:15:15)
        $y_{train}, y_{val}, y_{test} \leftarrow$ OneHotEncodeLabels($y_{train}, y_{val}, y_{test}$)
        return $X_{train}, X_{val}, X_{test}, y_{train}, y_{val}, y_{test}$
    end function
    function NormalizeData(X)
        $mean \leftarrow$ ComputeMean($X$)
        $std \leftarrow$ ComputeStd($X$)
        $X_{norm} \leftarrow \frac{X - mean}{std}$
        return $X_{norm}$
    end function
```

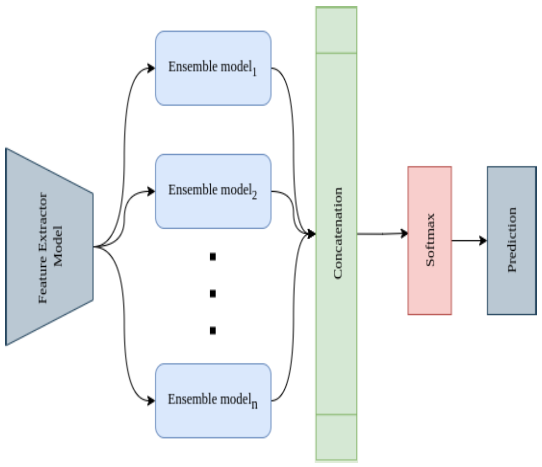


FIGURE 2. Class-based ensemble method of deep learning models.

C. BASELINE ARCHITECTURE

The proposed method employs a class-based ensemble approach to enhance the performance of the conventional deep learning model. Figure 2 illustrates the class-based ensemble method used in deep learning architecture.

In Figure 2, the class-based ensemble method is shown to begin with the extraction of feature maps through a head model. The ensemble method then employs separate Ensemble models to learn the features specific to each class: fall, pre-fall, and non-fall.

Figure 3 depicts the architecture proposed in this research, which was developed after evaluating various ensemble methods.

The following sections describe in detail the feature extractor model, ensemble model, and softmax function used in the proposed method.

1) FEATURE EXTRACTOR MODEL

The CNN architecture comprised multiple layers, including convolutional, pooling, and fully connected layers, which were utilized to extract relevant features from the input data.

The equations for the convolution layer can be represented as follows:

$$y_{i,j} = \sigma\left(\sum_{k,l} x_{i+k,j+l} w_{k,l} + b\right) \tag{1}$$

where x is the input data, y is the output feature map, w is the kernel (or filter) used for convolution, b is the bias term, and σ is an activation function, such as ReLU, which is commonly used in CNNs.

Then the pooling layer is used to reduce the feature maps' dimensionality, which helps reduce the computational cost and prevent overfitting. Here max pooling is used, which selects the maximum value from each pooling region. The equations for the max pooling operation can be represented as follows:

$$y_{i,j} = \max_{k,l} x_{i+k,j+l} \tag{2}$$

where x is the input data and y is the output of the pooling layer.

Finally, the fully connected layers combine the local features into global features that can be used for classification. The equations for the fully connected layer can be represented as follows:

$$y = \sigma\left(\sum_{i} w_i x_i + b\right) \tag{3}$$

where x is the input feature vector, w is the weight vector, b is the bias term, and σ is the activation function, such as softmax, which is commonly used for multiclass classification tasks.

After the fully connected layers, we obtained a feature vector that was fed into the ensemble of LSTM models for classification.

2) ENSEMBLE CONSTRUCTION

We used long short-term memory models to classify fall, pre-fall, and non-fall events based on the features extracted by the CNN. LSTMs are a type of recurrent neural network that can capture temporal dependencies in sequential data.

We designed three different LSTM models, one for each class, that take as input a sequence of feature vectors and output the corresponding class probability. Let x_t be the feature vector at time step t, and let h_t and c_t be the hidden state and cell state of the LSTM at time step t. The LSTM model can be defined as:

$$h_t, c_t = LSTM(x_t, h_{t-1}, c_{t-1}) \tag{4}$$

where LSTM is the LSTM function, which consists of an input gate, forget gate, output gate, and cell gate. The output of the LSTM at the last time step is fed into a fully connected layer with a softmax activation function to obtain the class probabilities.

We constructed an ensemble of the three LSTM models to improve the classification performance. Let M_1, M_2, M_3 be the three LSTM models for classifying fall, pre-fall, and non-fall events. The ensemble takes as input the feature vectors $x_1, x_2, \ldots, x_T$. It produces three sets of class probabilities $\mathbf{p}_1$, $\mathbf{p}2$, $\mathbf{p}3$, where $\mathbf{p}i = [pi1, pi2, pi3]$ is a vector of probabilities for classifying the i-th event as fall, pre-fall, and non-fall, respectively.

Each LSTM model M_i produces its own set of probabilities $\mathbf{p}_i$ by applying the softmax activation function to the output of the last fully connected layer. Let $\mathbf{o}_i$ be the output of the last fully connected layer of LSTM model M_i:

$$\mathbf{o}_i = \mathrm{FC}(h_T^i) \tag{5}$$

where h_T^i is the hidden state of LSTM model M_i at the last time step T, and FC is the fully connected layer.

Then, the probabilities $\mathbf{p}_i$ are obtained by applying the softmax function to $\mathbf{o}_i$:

$$p_{ij} = \frac{e^{o_{ij}}}{\sum_{k=1}^{3} e^{o_{ik}}}, \quad j = 1, 2, 3 \tag{6}$$

The final class probabilities $\mathbf{p}$ are obtained by taking the average of the three sets of probabilities:

$$\mathbf{p} = \frac{1}{3}(\mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3) \tag{7}$$

Therefore, the ensemble model classifies an input sequence $x_1, x_2, \ldots, x_T$ as fall, pre-fall, or non-fall depending on the class with the highest probability in $\mathbf{p}$.

Algorithm 2 presents the ensemble construction used in the proposed architecture.

This algorithm takes the feature vectors as input and returns the final class probabilities by applying each LSTM model to the input sequence, computing the softmax activation function for each output, and then averaging the three sets of probabilities.

3) SOFTMAX

The final step of the proposed method involves applying the softmax function to the output of the ensemble to obtain the corresponding class probabilities. The softmax function takes as input a vector $\mathbf{o} = [o_1, o_2, o_3]$ and returns a vector of probabilities $\mathbf{p} = [p_1, p_2, p_3]$ that sum up to 1. The softmax function is defined as:

$$p_i = \frac{e^{o_i}}{\sum_{j=1}^{3} e^{o_j}} \tag{8}$$

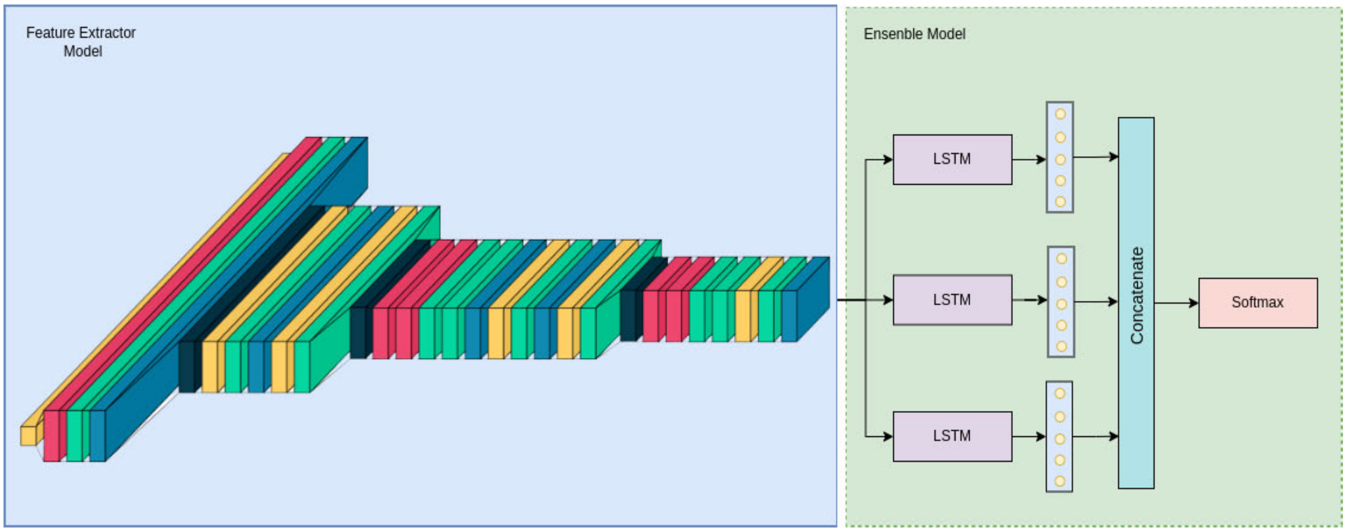


FIGURE 3. Proposed class-based ensemble method between CNN and LSTM.

```
Algorithm 2 Ensemble Construction
Input: Feature vectors $x_1, x_2, \ldots, x_T$
Output: Class probabilities $\mathbf{p} = [p_1, p_2, p_3]$
Apply each LSTM model $M_i$ to the input sequence
  $x_1, x_2, \ldots, x_T$ to obtain the output $\mathbf{o}_i = \mathrm{FC}(h_T^i)$, where
  $h_T^i$ is the hidden state of $M_i$ at the last time step $T$;
Apply the softmax activation function to each output $\mathbf{o}i$
  to obtain the class probabilities $\mathbf{p}i = [pi1, pi2, p_{i3}]$:
for $i = 1$ to 3 do
    for $j = 1$ to 3 do
        $p_{ij} = \frac{e^{o_{ij}}}{\sum_{k=1}^{3} e^{o_{ik}}}$;
Compute the average of the three sets of probabilities
  $\mathbf{p} = \frac{1}{3}(\mathbf{p}_1 + \mathbf{p}_2 + \mathbf{p}_3)$;
return p
```

where $i = 1, 2, 3$ and o_i is the i-th element of the output vector $\mathbf{o}$.

The final classification decision is made based on the class probabilities $\mathbf{p}$. The class with the highest probability is selected as the predicted class. Mathematically, the predicted class $\hat{y}$ is given by:

$$\hat{y} = \underset{i=1,2,3}{\operatorname{argmax}}\, p_i \tag{9}$$

where p_i is the probability of class i. If $\hat{y} = 1$, the input sequence is classified as a fall event. If $\hat{y} = 2$, the input sequence is classified as a pre-fall event. If $\hat{y} = 3$, the input sequence is classified as a non-fall event.

D. PROPOSED ALGORITHM

This section presents an algorithm 3 that demonstrates the proposed methods.

The algorithm presented is a class-based ensemble architecture for the fall detection system. It takes sensor data $\mathbf{x} \in \mathbb{R}^{T \times D}$ as input, where T is the number of time steps and D is the dimensionality of the sensor data. The algorithm's output is the predicted fall class $\hat{y} \in 0, 1, 2$, where 0 represents non-fall, 1 represents pre-fall, and 2 represents fall.

The algorithm consists of a head model and an ensemble model. The head model is a convolutional neural network that extracts features from the input sensor data, which are then reduced by feature map reduction. Let $\mathbf{z} \in \mathbb{R}^{T' \times D'}$ be the feature map obtained from the CNN, where $T' < T$ and $D' < D$. The reduced feature map is denoted by $\mathbf{z}' \in \mathbb{R}^{T'' \times D''}$.

The ensemble model consists of a Long Short Term Memory for each target class $c \in 0, 1, 2$. Let $\mathbf{h}_c \in \mathbb{R}^{T'' \times H}$ be the hidden state sequence obtained from the LSTM for target class c, where H is the hidden state dimensionality. The temporal dynamics of the hidden state sequence are recognized using a softmax function applied to the output of the LSTM, which is then passed through a fully connected layer to obtain the output sequence $\mathbf{y}_c \in \mathbb{R}^{T'' \times K}$, where $K = 3$ is the number of classes. The predicted class for target class c is denoted by $\hat{y}_c$, which is obtained by taking the argmax of the output sequence at time step T''.

The models are trained using cross-entropy loss on annotated data and fine-tuned on new data for better performance. Sensor data is input to the head model during testing, and the feature map is passed to the ensemble models to obtain the predicted fall class $\hat{y}$.

IV. RESULTS ANALYSIS

In this section, we elaborate on the experimental analysis, which comprises the implementation environment and the evaluation metrics employed to measure the models' performance. Subsequently, we comprehensively analyze the experimental results to provide insights into the proposed

Algorithm 3 Class-Based Ensemble Architecture for FDS

```
1: Input: Sensor data $\mathbf{x} \in \mathbb{R}^{T \times D}$, where $T$ is the time steps and $D$ is the sensor data dimensionality.
2: Output: Predicted fall class $\hat{y} \in 0, 1, 2$ (non-fall, pre-fall, fall).
3: Head model:
4:     Convolutional Neural Network: $\mathbf{z} = \text{CNN}(\mathbf{x})$, where $\mathbf{z} \in \mathbb{R}^{T' \times D'}$ is the feature map, $T' < T$ and $D' < D$.
5:     Feature map reduction: $\mathbf{z}' = \text{reduce}(\mathbf{z})$, where $\mathbf{z}' \in \mathbb{R}^{T'' \times D''}$ is the reduced feature map.
6: Ensemble model:
7:     Long Short Term Memory for each target class $c \in 0, 1, 2$:
8:         LSTM model: $\mathbf{h}_c = \text{LSTM}_c(\mathbf{z}')$, where $\mathbf{h}_c \in \mathbb{R}^{T'' \times H}$ is the hidden state sequence, and $H$ is the hidden state dimensionality.
9:         Temporal dynamics recognition: $\mathbf{y}_c = \text{softmax}(\mathbf{h}_c \cdot \mathbf{W}_c + \mathbf{b}_c)$, where $\mathbf{y}c \in \mathbb{R}^{T'' \times K}$ is the output sequence, and $K = 3$ is the number of classes.
10:        Independent predictions: $\hat{y}c = \text{argmax}(\mathbf{y}c, T'')$, where $\hat{y}c$ is the predicted class for target class $c$.
11: Training:
12:     Train Head model and Ensemble models on annotated data using cross-entropy loss.
13:     Fine-tune models on new data for better performance.
14: Testing:
15:     Input sensor data $\mathbf{x}$ to Head model.
16:     Generate feature map $\mathbf{z}$ and pass it to Ensemble models.
17:     Obtain prediction for each target class $c$ from its Ensemble model.
18:     Output predicted fall class $\hat{y} = \text{argmax}(\mathbf{y}\cdot, T'')$, where $\mathbf{y}\cdot, T''$ is the concatenation of the output sequences for all target classes at time step $T''$.
```

method's effectiveness, the influence of distinct factors on its performance, and its practical implications.

A. EXPERIMENTAL SETUP

The Keras and TensorFlow deep learning frameworks were utilized to implement the neural network models. The experiments were conducted in a virtual environment equipped with an RTX3070 GPU, a Ryzen 7 processor, 16 GB of RAM, and a 1 TB solid-state drive. The operating system of the virtual environment was Ubuntu 20.04 LTS, and the software environment consisted of Python 3.8, Keras 2.6.0, and TensorFlow 2.6.0. To ensure the reproducibility of the experimental results, the virtual environment was created using the Anaconda package manager. To expedite the training of the neural network models, the CUDA toolkit and cuDNN library were installed and configured to function with the GPU.

B. EVALUATION METRICES

To evaluate the performance of the proposed method, we used three evaluation metrics: accuracy, sensitivity, and specificity, for each class.

1) ACCURACY

This metric measures the overall classification accuracy of the model and is defined as the proportion of correctly classified samples among all samples in the test set. Formally, let y_i be the true label of the i-th sample and $\hat{y}_i$ be the predicted label of the i-th sample. Then, the accuracy of the model is given by:

$$Acc = \frac{1}{N} \sum_{i=1}^{N} [y_i = \hat{y}_i] \tag{10}$$

where N is the total number of samples in the test set, and $[y_i = \hat{y}_i]$ is an indicator function that equals 1 if $y_i = \hat{y}_i$ and 0 otherwise.

2) SENSITIVITY

Sensitivity is also known as Recall or TPR (true positive rate). This metric measures the proportion of true positive samples (i.e., fall samples correctly classified as falls) among all positive samples (i.e., fall samples) in the test set. Formally, let P be the set of all positive samples in the test set, and TP be the set of true positive samples. Then, the sensitivity of the model is given by:

$$Sen = \frac{|TP|}{|P|} \tag{11}$$

where $|\cdot|$ denotes the cardinality of a set.

3) SPECIFICITY

This metric measures the proportion of true negative samples (i.e., non-fall samples correctly classified as non-falls) among all negative samples (i.e., non-fall samples) in the test set. Formally, let N be the set of all negative samples in the test set, and TN be the set of true negative samples. Then, the specificity of the model is given by:

$$Spec = \frac{|TN|}{|N|} \tag{12}$$

4) PRECISION

Precision measures the proportion of true positives among the total predicted positives. It is defined as the ratio of true positive predictions (TP) to the sum of true positive and false positive predictions (FP), which can be expressed as follows:

$$Prec = \frac{TP}{(TP + FP)} \tag{13}$$

These metrics provide a comprehensive evaluation of our proposed model's performance in terms of overall accuracy and the ability to detect falls and non-falls correctly.

TABLE 3. Accuracy, Specificity, and Sensitivity of the proposed method on the D1, D2, and D3 datasets.

Dataset	Accuracy			Specificity			Sensitivity		
	Non-Fall	Pre-Fall	Fall	Non-Fall	Pre-Fall	Fall	Non-Fall	Pre-Fall	Fall
D1	0.9656	0.9457	0.9734	0.9718	0.9628	0.9706	0.9610	0.9528	0.9822
D2	0.9736	0.9560	0.9843	0.9702	0.9514	0.9626	0.9602	0.9612	0.9832
D3	0.9546	0.9645	0.9612	0.9646	0.9412	0.9687	0.9588	0.9600	0.9812

TABLE 4. Precision, Recall, and F1-score of the proposed method on the D1, D2, and D3 datasets.

Dataset	Precision			Recall			F1-score		
	Non-Fall	Pre-Fall	Fall	Non-Fall	Pre-Fall	Fall	Non-Fall	Pre-Fall	Fall
D1	0.9451	0.9315	0.9524	0.9610	0.9528	0.9822	0.9528	0.9520	0.9670
D2	0.9464	0.9320	0.9658	0.9602	0.9612	0.9832	0.9533	0.9465	0.9744
D3	0.9245	0.9398	0.9425	0.9588	0.9600	0.9812	0.9413	0.9499	0.9614

5) F1-SCORE

It is the harmonic mean of precision and recall, giving equal importance to both metrics. The F1-score ranges between 0 and 1, with 1 indicating perfect precision (P) and recall (R). The F1-score is calculated using the following formula:

$$F1 - score = 2 \times \frac{P \times R}{(P + R)} \tag{14}$$

C. RESULTS

Table 3 displays the accuracy, sensitivity, and specificity of the proposed method under different dataset configurations, namely D1, D2, and D3. In dataset D1, the proposed method achieved an accuracy of 0.9656 for non-fall events, 0.9457 for pre-fall events, and 0.9734 for fall events. The specificity of the proposed method was 0.9718 for non-fall events, 0.9628 for pre-fall events, and 0.9706 for fall events. The sensitivity of the proposed method was 0.9610 for non-fall events, 0.9528 for pre-fall events, and 0.9822 for fall events. For dataset D2, the proposed method achieved an accuracy of 0.9736 for non-fall events, 0.9560 for pre-fall events, and 0.9843 for fall events. The specificity of the proposed method was 0.9702 for non-fall events, 0.9514 for pre-fall events, and 0.9626 for fall events. The sensitivity of the proposed method was 0.9602 for non-fall events, 0.9612 for pre-fall events, and 0.9832 for fall events. For dataset D3, the proposed method achieved an accuracy of 0.9546 for non-fall events, 0.9645 for pre-fall events, and 0.9612 for fall events. The specificity of the proposed method was 0.9646 for non-fall events, 0.9412 for pre-fall events, and 0.9687 for fall events. The sensitivity of the proposed method was 0.9588 for non-fall events, 0.9600 for pre-fall events, and 0.9812 for fall events.

Table 4 reports the precision, recall, and F1-score of the proposed method on three datasets (D1, D2, and D3). For the D1 dataset, the proposed method achieved a precision of 0.9451 for the Non-Fall class, 0.9315 for the Pre-Fall class, and 0.9524 for the Fall class. The recall values for Non-Fall, Pre-Fall, and Fall classes were 0.9610, 0.9528, and 0.9822, respectively. The F1-score values for Non-Fall, Pre-Fall, and Fall classes were 0.9528, 0.9520, and 0.9670, respectively. For the D2 dataset, the proposed method achieved a precision of 0.9464 for the Non-Fall class, 0.9320 for the Pre-Fall class, and 0.9658 for the Fall class. The recall values for Non-Fall, Pre-Fall, and Fall classes were 0.9602, 0.9612, and 0.9832, respectively. The F1-score values for Non-Fall, Pre-Fall, and Fall classes were 0.9533, 0.9465, and 0.9744, respectively. For the D3 dataset, the proposed method achieved a precision of 0.9245 for the Non-Fall class, 0.9398 for the Pre-Fall class, and 0.9425 for the Fall class. The recall values for Non-Fall, Pre-Fall, and Fall classes were 0.9588, 0.9600, and 0.9812, respectively. The F1-score values for Non-Fall, Pre-Fall, and Fall classes were 0.9413, 0.9499, and 0.9614, respectively.

We initially trained our proposed class-based ensemble approach with various architectural configurations to achieve these results. As a feature extractor model, we experimented with four combinations: convolutional layers with max pooling layers, convolutional layers with average pooling layers, separable convolutional layers with max pooling layers, and separable convolutional layers with average pooling layers. To enhance the performance of the feature extractor model, we further used ensemble methods including LSTM, GRU, and Bi-directional LSTM. The corresponding results obtained from all the aforementioned combinations are presented in Figure 4.

Based on the results shown in Figure 4, it can be observed that the combination of convolutional layers with max pooling layers and LSTM method produces the highest accuracy for all sets of data. Therefore, we propose this combination as the optimal architecture for our proposed fall detection model.

D. HYPER-PARAMETERS TUNING

Upon finalizing the model combination, the proposed approach incorporated various hyperparameters, including the number of filters, kernel size, dropout rate, learning rate, and LSTM cells. To determine their optimal values,

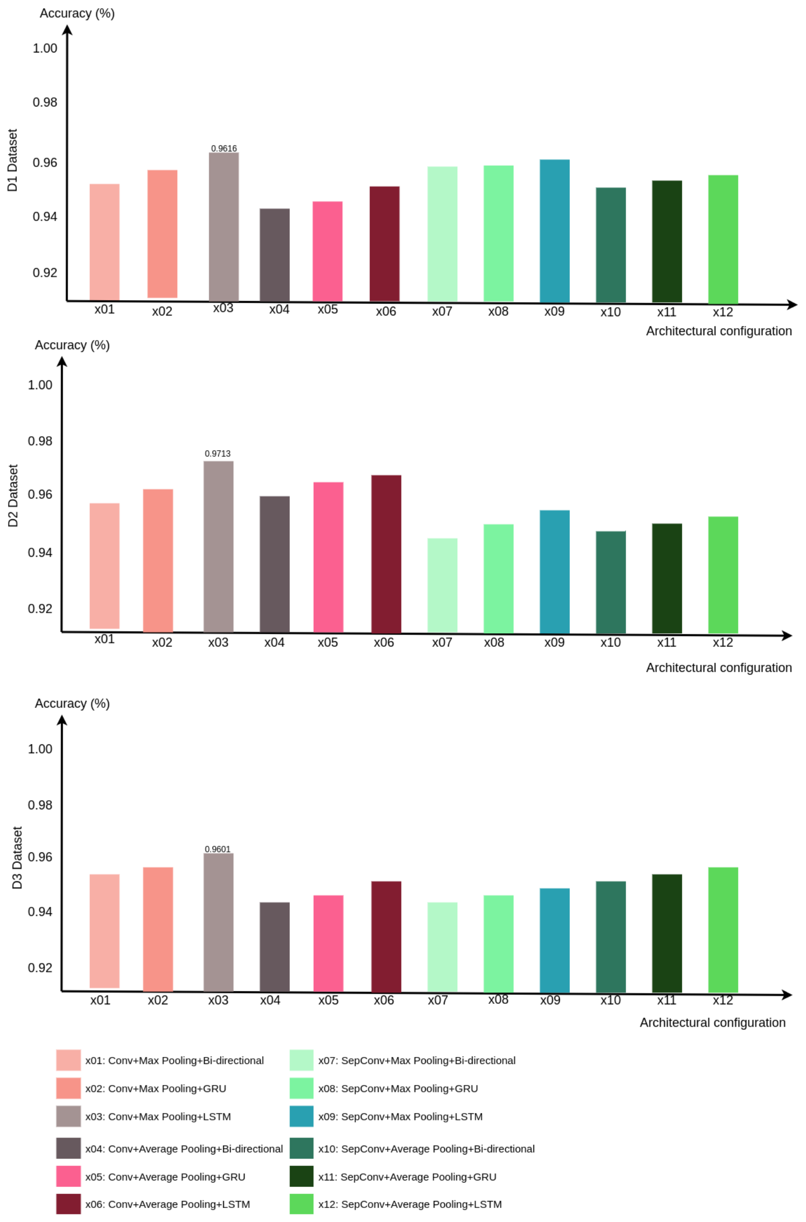


FIGURE 4. The figure illustrates the average accuracy of 3 classes (Pre-fall, Non-fall, Fall) obtained from different combinations of feature extractor and ensemble methods.

TABLE 5. Comparison of Accuracy, Specificity, and Sensitivity of the proposed method on the D1 dataset.

Ref.	Accuracy			Specificity			Sensitivity		
	Non-Fall	Pre-Fall	Fall	Non-Fall	Pre-Fall	Fall	Non-Fall	Pre-Fall	Fall
Musci et al. [50]	0.9485	0.8956	0.9591	0.9592	0.9503	**0.9861**	0.9378	0.8408	0.9321
Torti et al. [51]	0.9282	0.9090	0.9701	**0.9724**	0.9071	0.9785	0.8839	0.9108	0.9797
Yu et al. [52]	0.9492	0.9372	0.9733	0.9669	0.9439	0.9859	0.9315	0.9304	0.9607
Proposed	**0.9656**	**0.9457**	**0.9734**	0.9718	**0.9628**	0.9706	**0.9610**	**0.9528**	**0.9822**

a grid search technique was employed. Here, we provide an overview of each hyperparameter and the process used to compute them. The number of filters in the convolutional layers was carefully selected by exploring different values such as 16, 32, and 64. Through rigorous evaluation, we identified that utilizing 16 and 32 filters yielded favorable model performance. Additionally, we conducted experiments with multiple kernel sizes, including 3×3 and 5×5, to assess their

TABLE 6. Comparison of fall detection methods on the SisFall dataset regarding accuracy, sensitivity, and specificity.

Ref.	Method	Dataset	Accuracy	Sensitivity	Specificity
Yacchirema et al. [16]	Ensemble learning	SisFall	0.9872	0.9622	0.9460
Luna-Perejon et al. [38]	RNN	SisFall	-	0.882	0.964
Aguiar et al. [53]	Decision tree	SisFall	0.929	0.923	0.932
Pierleoni et al. [54]	Support vector machine	SisFall	0.9518	0.9037	1.00
Torti et al. [55]	RNN (LSTM)	SisFall	0.9551	0.927	0.941
Kabir et al.	Class-based Ensemble Method	SisFall	0.9616	0.9653	0.9684

TABLE 7. Hyperparameter values and their corresponding values used in the grid search for the proposed approach.

Parameter	Value 1	Value 2	Value 3	Value used
Number of filters	16	32	64	16, 32
Kernel size	3x3	5x5	-	3x3
Dropout rate	0.2	0.4	0.5	0.5
Learning rate	0.001	0.01	0.1	0.001
LSTM cells	64	128	256	64, 128

effectiveness in capturing spatial information. Based on our findings, the 3 × 3 kernel size demonstrated superior performance. Furthermore, we investigated dropout rates, such as 0.2, 0.4, and 0.5, to balance model performance and generalization. Notably, a dropout rate of 0.5 exhibited impressive results, leading us to select it as the optimal rate. As for the learning rate, which governs the step size for parameter updates during training, different values such as 0.001, 0.01, and 0.1 were explored. After careful evaluation, we determined that a learning rate of 0.001 facilitated the fastest convergence and exhibited the best performance. Regarding the LSTM layer, we investigated the number of LSTM cells or units to capture the temporal dynamics of the falling process. We evaluated various values, including 64, 128, and 256, intending to model long-term dependencies effectively. Our experiments indicated that 128 and 64 LSTM cells were more suitable for this task, improving model performance and capturing relevant temporal patterns. These selected hyperparameter values were obtained through the rigorous grid search process, ensuring their optimal configuration for the proposed approach. Table 7 provides information on the hyperparameters used in the grid search for the proposed approach.

E. BENCHMARK

Table 5 compares the proposed method with three state-of-the-art approaches regarding accuracy, sensitivity, and specificity. The results indicate that our proposed method outperforms all three methods in terms of accuracy, achieving scores of 0.9656, 0.9457, and 0.9734 for the Non-fall, Pre-fall, and Fall classes, respectively. While the method proposed by Torti et al. [51] exhibits better specificity for the Non-fall class, and the method proposed by Musci et al. [50] has better specificity for the Fall class, our method achieves the best specificity for the Pre-fall class. Moreover, our proposed method surpasses all three methods in terms of sensitivity, achieving scores of 0.9610, 0.9528, and 0.9822 for each respective class. In addition table 6 compares various fall detection methods in terms of their accuracy, sensitivity, and specificity on the SisFall dataset. The methods compared in the table include Decision tree, Support vector machine, Ensemble learning, RNN (LSTM), RNN, and Class-based Ensemble Method. Based on the table, the proposed method (Class-based Ensemble Method) achieves an accuracy of 0.9616, which is comparable to the best-performing methods in the literature. The sensitivity and specificity metrics of the proposed method are also relatively high (0.9653 and 0.9684, respectively), indicating that it can accurately detect falls and non-falls.

V. DISCUSSION

In this study, we proposed a deep learning architecture for fall detection that utilizes a class-based ensemble method. Our approach involved a CNN-based feature extraction module and an ensemble of three LSTM models, each trained to classify one of the three classes: fall, pre-fall, and non-fall. The final classification decision was made using the softmax function applied to the ensemble's output. Our experiments demonstrated that the proposed method achieved better performance than state-of-the-art approaches for elderly people. The results show that the class-based ensemble approach can effectively enhance classification accuracy by learning the features specific to each class.

The proposed architecture allows for the simultaneous extraction of spatial and temporal features from accelerometer and gyroscope data by combining the strengths of CNNs in capturing spatial features and LSTMs in modeling temporal dependencies. This leads to improved performance in fall detection compared to existing methods that use either CNNs or LSTMs alone. The proposed approach can be used in real-time fall detection systems. The methodology combines convolutional neural networks for robust feature extraction and long short-term memory networks for modeling temporal dynamics, making it suitable for real-time fall detection applications. The study's findings suggest that the proposed approach achieves commendable results in identifying pre-fall and fall states, which makes it a promising method for real-time fall detection systems.

During the tuning phase of our fall detection method, we observed that the convolutional layers outperformed the separable convolutional layers. We attribute this to the fact that falls usually involve intricate body movements and environmental interactions, necessitating a high degree of spatial awareness and pattern recognition. Convolutional layers can better capture complex spatial patterns, while separable convolutional layers may not capture the same level of detail and context [56]. This is due to their ability to learn local feature representations and hierarchically combine them to learn more abstract and complex patterns. In the context of fall detection using accelerometer and gyroscope data, convolutional layers effectively capture relevant features that distinguish between different classes of motion [57]. Our experimental results indicate that max-pooling layers outperform average pooling layers in fall detection. This can be attributed to the fact that max-pooling layers retain the most significant features of the input while discarding irrelevant and noisy information. In fall detection, capturing critical features of fall-related movements, such as sudden changes in acceleration and orientation, is crucial. Max-pooling layers excel in capturing these critical features and preserving their spatial relationships, which may not be as efficiently achieved with average pooling layers. Thus, using max-pooling layers in our method enhances the accuracy and reliability of fall detection.

Our experimental results also show that LSTM outperforms GRU and Bi-directional LSTM as an ensemble method for each class in fall detection. This can be attributed to the fact that LSTM has more trainable parameters than GRU, enabling it to capture more complex temporal dependencies in the data. Moreover, LSTM's gating mechanism allows it to retain or discard information from previous time steps selectively. It can be particularly useful in detecting falls involving sudden acceleration and orientation changes. In contrast, Bi-directional LSTM may not be as effective in fall detection, as it requires access to future time steps to make predictions, which may not always be available in real-time applications. Additionally, the complexity of the bi-directional architecture may lead to overfitting and reduced generalization performance. Therefore, our experiments indicate that using LSTM as the ensemble method is a more suitable choice for fall detection.

The use of CNN-based feature extraction also helped to overcome some of the challenges associated with fall detection, such as variations in lighting, occlusion, and differences in clothing and body shapes. Furthermore, the proposed architecture is computationally efficient and essential for real-time fall detection applications due to its class-based mechanism. In traditional approaches to fall detection using deep neural networks, a single model is trained to classify all three classes of falling processes (non-fall, pre-fall, and fall) based on accelerometer and gyroscope data. This model must be trained on a large amount of data and optimized for high accuracy in all three classes, which can be computationally intensive. In the class-based ensemble framework, on the other hand, separate models are trained to classify each class of falling process individually. This allows each model to specialize in detecting a particular type of fall rather than being optimized for all three categories. Ensemble learning combines the outputs of the individual models to obtain a final prediction. Hence this approach has been shown to improve overall accuracy while reducing the computational cost compared to traditional techniques.

In addition, the full sensor-sets used in D1 and D2 are highly applicable in real-life scenarios. The external sensing mote with two accelerometers and a gyroscope used in D1 is a compact and wearable device that can easily be attached to clothing or worn on the body without causing inconvenience or discomfort to the user. Similarly, the smartphone and four external IMUs used in D2 are commonly available devices that can be easily carried in a pocket or attached to clothing. These sensor sets are practical and cost-effective compared to other more sophisticated and expensive sensor systems. This makes them more accessible to a wider population and allows easier deployment in real-world fall detection scenarios. However, these sensor configurations may require additional devices or equipment worn by the user, which may not be comfortable for the user.

Despite the promising results achieved in this study, there is still room for further improvements in the proposed method. For example, while we used a class-based ensemble approach, other ensemble methods could be explored to improve the classification performance further. The proposed architecture can also be optimized for different input modalities, such as depth or infrared cameras, to enhance detection accuracy in low light or occluded environments. Another direction for future research is to investigate the interpretability of the proposed architecture. While deep learning models have shown impressive results in many applications, their black-box nature limits their interpretability. Therefore, exploring methods to extract meaningful information from the features learned by the proposed architecture can help to build trust and understanding of the model's decision-making process.

One limitation of this study is the dataset used both UMAFall and SisFall are datasets of emulated falls. Emulated falls have some limitations compared to real-world falls [45]. In real-world falls, there is a higher degree of variability in terms of how the falls occur, the physical environment where the fall occurs, and the demographic characteristics of the fallers. On the other hand, emulated falls are created under controlled conditions, and the variability in the falling process is limited. As a result, the datasets of emulated falls may not capture all the variations and subtleties of real-world falls, making it challenging to generalize the results to the real-world fall detection scenario. Another limitation is that the UMAFall dataset only included subjects up to 55, and the two subjects over 50 did not experience any falls. Similarly, in SisFall, the 15 healthy elderly subjects did not experience any falls. In addition to the differences between emulated and real-world falls, it is also possible that there are variations in fall characteristics and dynamics between younger individuals and older individuals.

The computational complexity of this approach is relatively high due to the use of multiple layers in both the CNN and LSTM networks. The CNN network comprises several convolutional layers that perform feature extraction from the input data. The LSTM network, on the other hand, utilizes recurrent connections that enable the network to retain information over time, thereby capturing the temporal dynamics of the falling process. The proposed approach's complexity is increased using class ensembles, which involves training multiple models on the same data but with different initial conditions. The output of each model is then aggregated to obtain a final prediction. This technique requires training and storing multiple models, leading to higher computational and memory requirements. Despite its high computational complexity, the proposed approach achieves state-of-the-art performance in fall detection, indicating that the benefits outweigh the computational costs. However, optimizing the approach's computational efficiency without compromising performance is an area for future research.

In addition, in our future work, we aim to propose a comprehensive fall prevention system that will address key aspects such as sensor calibration, robustness, battery life, sensor placement, and evaluation in real-world settings specifically for older adults. Also, we plan to conduct more statistical tests to compare the detection performance of the investigated models.

VI. CONCLUSION

The proposed ensemble CNN-LSTM-based fall detection system for older adults has shown promising results in detecting pre-fall, fall, and non-fall events. Combining the strengths of convolutional neural network and long short-term memory models, the proposed system achieved 97.34% accuracy in detecting falls, 94.57% accuracy in detecting pre-fall events, and 96.56% accuracy in detecting non-fall events for well-known SiSFall. The system's non-invasive, cost-effective, and easily deployable nature makes it a valuable tool for fall detection, a critical issue for the health and safety of older adults. Future work could focus on optimizing the model's hyperparameters, exploring the use of additional sensor data, and conducting large-scale studies to evaluate the system's effectiveness in real-world settings. Overall, the proposed system has great potential to become an essential tool for fall detection in the future, making it a valuable contribution to healthcare technology.

REFERENCES

[1] *WHO Global Report on Falls Prevention in Older Age*, World Health Organization, Geneva, Switzerland, 2008.

[2] M. Montero-Odasso, N. van der Velde, F. C. Martin, M. Petrovic, M. P. Tan, J. Ryg, S. Aguilar-Navarro, N. B. Alexander, C. Becker, and H. Blain, "World guidelines for falls prevention and management for older adults: A global initiative," *Age Ageing*, vol. 51, no. 9, 2022, Art. no. afac205.

[3] L. Ren and Y. Peng, "Research of fall detection and fall prevention technologies: A systematic review," *IEEE Access*, vol. 7, pp. 77702–77722, 2019.

[4] A. S. Syed, D. Sierra-Sosa, A. Kumar, and A. Elmaghraby, "A deep convolutional neural network-XGB for direction and severity aware fall detection and activity recognition," *Sensors*, vol. 22, no. 7, p. 2547, Mar. 2022.

[5] A. S. Syed, D. Sierra-Sosa, A. Kumar, and A. Elmaghraby, "A hierarchical approach to activity recognition and fall detection using wavelets and adaptive pooling," *Sensors*, vol. 21, no. 19, p. 6653, Oct. 2021.

[6] Md. M. Islam, O. Tayan, M. R. Islam, M. S. Islam, S. Nooruddin, M. N. Kabir, and M. R. Islam, "Deep learning based systems developed for fall detection: A review," *IEEE Access*, vol. 8, pp. 166117–166137, 2020.

[7] S. Anjum, N. Khan, R. Khalid, M. Khan, D. Lee, and C. Park, "Fall prevention from ladders utilizing a deep learning-based height assessment method," *IEEE Access*, vol. 10, pp. 36725–36742, 2022.

[8] T. Tongskulroongruang, P. Wiphunawat, W. Jutharee, W. Kaewmahanin, T. Rassameecharoenchai, T. Jennawasin, and B. Kaewkamnerdpong, "Comparative study on fall detection using machine learning approaches," in *Proc. 19th Int. Conf. Electr. Eng./Electron., Comput., Telecommun. Inf. Technol. (ECTI-CON)*, May 2022, pp. 1–4.

[9] D. K. Agrawal, W. Usaha, S. Pojprapai, and P. Wattanapan, "Fall risk prediction using wireless sensor insoles with machine learning," *IEEE Access*, vol. 11, pp. 23119–23126, 2023.

[10] S. Usmani, A. Saboor, M. Haris, M. A. Khan, and H. Park, "Latest research trends in fall detection and prevention using machine learning: A systematic review," *Sensors*, vol. 21, no. 15, p. 5134, Jul. 2021.

[11] S. Rastogi and J. Singh, "A systematic review on machine learning for fall detection system," *Comput. Intell.*, vol. 37, no. 2, pp. 951–974, May 2021.

[12] G. Santos, P. Endo, K. Monteiro, E. Rocha, I. Silva, and T. Lynn, "Accelerometer-based human fall detection using convolutional neural networks," *Sensors*, vol. 19, no. 7, p. 1644, Apr. 2019.

[13] T. Han, W. Kang, and G. Choi, "IR-UWB sensor based fall detection method using CNN algorithm," *Sensors*, vol. 20, no. 20, p. 5948, Oct. 2020.

[14] L. Ma, M. Liu, N. Wang, L. Wang, Y. Yang, and H. Wang, "Room-level fall detection based on ultra-wideband (UWB) monostatic radar and convolutional long short-term memory (LSTM)," *Sensors*, vol. 20, no. 4, p. 1105, Feb. 2020.

[15] S. A. Ajagbe and M. O. Adigun, "Deep learning techniques for detection and prediction of pandemic diseases: A systematic literature review," *Multimedia Tools Appl.*, vol. 82, pp. 1–35, May 2023.

[16] D. Yacchirema, J. S. de Puga, C. Palau, and M. Esteve, "Fall detection system for elderly people using IoT and ensemble machine learning algorithm," *Pers. Ubiquitous Comput.*, vol. 23, nos. 5–6, pp. 801–817, Nov. 2019.

[17] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers Comput. Sci.*, vol. 14, no. 2, pp. 241–258, Apr. 2020.

[18] A. Sucerquia, J. López, and J. Vargas-Bonilla, "SisFall: A fall and movement dataset," *Sensors*, vol. 17, no. 12, p. 198, Jan. 2017.

[19] E. Casilari, J. A. Santoyo-Ramón, and J. M. Cano-García, "Analysis of a smartphone-based architecture with multiple mobility sensors for fall detection," *PLoS ONE*, vol. 11, no. 12, Dec. 2016, Art. no. e0168069.

[20] N. Maray, A. H. Ngu, J. Ni, M. Debnath, and L. Wang, "Transfer learning on small datasets for improved fall detection," *Sensors*, vol. 23, no. 3, p. 1105, Jan. 2023.

[21] S. Y. Sheikh and M. T. Jilani, "A ubiquitous wheelchair fall detection system using low-cost embedded inertial sensors and unsupervised one-class SVM," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 1, pp. 147–162, Jan. 2023.

[22] X. Wu, Y. Zheng, C.-H. Chu, L. Cheng, and J. Kim, "Applying deep learning technology for automatic fall detection using mobile sensors," *Biomed. Signal Process. Control*, vol. 72, Feb. 2022, Art. no. 103355.

[23] H. Li, A. Mehul, J. Le Kernec, S. Z. Gurbuz, and F. Fioranelli, "Sequential human gait classification with distributed radar sensor fusion," *IEEE Sensors J.*, vol. 21, no. 6, pp. 7590–7603, Mar. 2021.

[24] C.-B. Lin, Z. Dong, W.-K. Kuan, and Y.-F. Huang, "A framework for fall detection based on OpenPose skeleton and LSTM/GRU models," *Appl. Sci.*, vol. 11, no. 1, p. 329, Dec. 2020.

[25] H. Li, A. Shrestha, H. Heidari, J. L. Kernec, and F. Fioranelli, "Bi-LSTM network for multimodal continuous human activity recognition and fall detection," *IEEE Sensors J.*, vol. 20, no. 3, pp. 1191–1201, Feb. 2020.

[26] J. He, Z. Zhang, X. Wang, and S. Yang, "A low power fall sensing technology based on FD-CNN," *IEEE Sensors J.*, vol. 19, no. 13, pp. 5110–5118, Jul. 2019.

[27] N. Zerrouki, F. Harrou, Y. Sun, and A. Houacine, "Accelerometer and camera-based strategy for improved human fall detection," *J. Med. Syst.*, vol. 40, no. 12, pp. 1–16, Dec. 2016.

[28] M. Mubashir, L. Shao, and L. Seed, "A survey on fall detection: Principles and approaches," *Neurocomputing*, vol. 100, pp. 144–152, Jan. 2013.

[29] V. Spasova and I. Iliev, "A survey on automatic fall detection in the context of ambient assisted living systems," *Int. J. Adv. Comput. Res.*, vol. 4, no. 1, p. 94, 2014.

[30] J. Gutiérrez, V. Rodríguez, and S. Martin, "Comprehensive review of vision-based fall detection systems," *Sensors*, vol. 21, no. 3, p. 947, Feb. 2021.

[31] M. Amsaprabhaa, "Multimodal spatiotemporal skeletal kinematic gait feature fusion for vision-based fall detection," *Expert Syst. Appl.*, vol. 212, Feb. 2023, Art. no. 118681.

[32] A. Núñez-Marcos, G. Azkune, and I. Arganda-Carreras, "Vision-based fall detection with convolutional neural networks," *Wireless Commun. Mobile Comput.*, vol. 2017, pp. 1–16, Dec. 2017.

[33] J.-L. Chua, Y. C. Chang, and W. K. Lim, "A simple vision-based fall detection technique for indoor video surveillance," *Signal, Image Video Process.*, vol. 9, no. 3, pp. 623–633, Mar. 2015.

[34] A. Ramachandran and A. Karuppiah, "A survey on recent advances in wearable fall detection systems," *BioMed Res. Int.*, vol. 2020, pp. 1–17, Jan. 2020.

[35] E. Casilari, R. Lora-Rivera, and F. García-Lagos, "A wearable fall detection system using deep learning," in *Proc. Int. Conf. Ind., Eng. Other Appl. Appl. Intell. Syst.* Graz, Austria: Springer, Jul. 2019, pp. 445–456.

[36] F. Wu, H. Zhao, Y. Zhao, and H. Zhong, "Development of a wearable-sensor-based fall detection system," *Int. J. Telemedicine Appl.*, vol. 2015, pp. 1–11, Jan. 2015.

[37] W. Saadeh, M. A. B. Altaf, and M. S. B. Altaf, "A high accuracy and low latency patient-specific wearable fall detection system," in *Proc. IEEE EMBS Int. Conf. Biomed. Health Informat. (BHI)*, Feb. 2017, pp. 441–444.

[38] F. Luna-Perejón, M. J. Domínguez-Morales, and A. Civit-Balcells, "Wearable fall detector using recurrent neural networks," *Sensors*, vol. 19, no. 22, p. 4885, Nov. 2019.

[39] A. Spagnolli, E. Guardigli, V. Orso, A. Varotto, and L. Gamberini, "Measuring user acceptance of wearable symbiotic devices: validation study across application scenarios," in *Proc. Int. Workshop Symbiotic Interact.* Helsinki, Finland: Springer, Oct. 2014, pp. 87–98.

[40] A. Puri, B. Kim, O. Nguyen, P. Stolee, J. Tung, and J. Lee, "User acceptance of wrist-worn activity trackers among community-dwelling older adults: Mixed method study," *JMIR mHealth uHealth*, vol. 5, no. 11, p. e173, Nov. 2017.

[41] N. Zerrouki, F. Harrou, Y. Sun, A. Z. A. Djafer, and H. Amrane, "A survey on recent advances in fall detection systems using machine learning formalisms," in *Proc. 7th Int. Conf. Frontiers Signal Process. (ICFSP)*, Sep. 2022, pp. 35–39.

[42] G. Şengül, M. Karakaya, S. Misra, O. O. Abayomi-Alli, and R. Damaševičius, "Deep learning based fall detection using smartwatches for healthcare applications," *Biomed. Signal Process. Control*, vol. 71, Jan. 2022, Art. no. 103242.

[43] L. Wang, M. Peng, and Q. Zhou, "Pre-impact fall detection based on multi-source CNN ensemble," *IEEE Sensors J.*, vol. 20, no. 10, pp. 5442–5451, May 2020.

[44] Y. Li, Z. Zuo, and J. Pan, "Sensor-based fall detection using a combination model of a temporal convolutional network and a gated recurrent unit," *Future Gener. Comput. Syst.*, vol. 139, pp. 53–63, Feb. 2023.

[45] E. Casilari and C. A. Silva, "An analytical comparison of datasets of real-world and simulated falls intended for the evaluation of wearable fall alerting systems," *Measurement*, vol. 202, Oct. 2022, Art. no. 111843.

[46] B. T. Nukala, N. Shibuya, A. Rodriguez, J. Tsay, J. Lopez, T. Nguyen, S. Zupancic, and D. Y.-C. Lie, "An efficient and robust fall detection system using wireless gait analysis sensor with artificial neural network (ANN) and support vector machine (SVM) algorithms," *Open J. Appl. Biosensor*, vol. 3, no. 4, pp. 29–39, 2014.

[47] N. De Raeve, A. Shahid, M. de Schepper, E. De Poorter, I. Moerman, J. Verhaevert, P. Van Torre, and H. Rogier, "Bluetooth-low-energy-based fall detection and warning system for elderly people in nursing homes," *J. Sensors*, vol. 2022, pp. 1–14, Jan. 2022.

[48] S. K. Yadav, A. Luthra, K. Tiwari, H. M. Pandey, and S. A. Akbar, "ARFDNet: An efficient activity recognition & fall detection system using latent feature pooling," *Knowl.-Based Syst.*, vol. 239, Mar. 2022, Art. no. 107948.

[49] S. Saeb, L. Lonini, A. Jayaraman, D. C. Mohr, and K. P. Kording, "The need to approximate the use-case in clinical machine learning," *GigaScience*, vol. 6, no. 5, May 2017, Art. no. gix019.

[50] M. Musci, D. De Martini, N. Blago, T. Facchinetti, and M. Piastra, "Online fall detection using recurrent neural networks," 2018, *arXiv:1804.04976*.

[51] E. Torti, A. Fontanella, M. Musci, N. Blago, D. Pau, F. Leporati, and M. Piastra, "Embedding recurrent neural networks in wearable systems for real-time fall detection," *Microprocessors Microsyst.*, vol. 71, Nov. 2019, Art. no. 102895.

[52] X. Yu, H. Qiu, and S. Xiong, "A novel hybrid deep neural network to predict pre-impact fall for older people based on wearable inertial sensors," *Frontiers Bioeng. Biotechnol.*, vol. 8, p. 63, Feb. 2020.

[53] B. Aguiar, T. Rocha, J. Silva, and I. Sousa, "Accelerometer-based fall detection for smartphones," in *Proc. IEEE Int. Symp. Med. Meas. Appl. (MeMeA)*, Jun. 2014, pp. 1–6.

[54] P. Pierleoni, A. Belli, L. Palma, M. Pellegrini, L. Pernini, and S. Valenti, "A high reliability wearable device for elderly fall detection," *IEEE Sensors J.*, vol. 15, no. 8, pp. 4544–4553, Aug. 2015.

[55] E. Torti, A. Fontanella, M. Musci, N. Blago, D. Pau, F. Leporati, and M. Piastra, "Embedded real-time fall detection with deep learning on wearable devices," in *Proc. 21st Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2018, pp. 405–412.

[56] L. Jiao, M. Liang, H. Chen, S. Yang, H. Liu, and X. Cao, "Deep fully convolutional network-based spatial distribution prediction for hyperspectral image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 10, pp. 5585–5599, Oct. 2017.

[57] N. Lu, Y. Wu, L. Feng, and J. Song, "Deep learning for fall detection: three-dimensional CNN combined with LSTM on video kinematic data," *IEEE J. Biomed. Health Informat.*, vol. 23, no. 1, pp. 314–323, Jan. 2019.

MD. MOHSIN KABIR received the Bachelor of Science degree in CSE from the Bangladesh University of Business and Technology (BUBT), Bangladesh, in 2021. He is currently pursuing the joint master's degree in intelligent field robotics systems (IFRoS) with the University of Girona, Spain, and Eötvös Loránd University, Hungary, funded by Erasmus Mundus Scholarship (2022–2024). In addition to his studies, he holds a position as a Lecturer with the Department of Computer Science and Engineering, BUBT (study-leave), where he was a Research Assistant and a Researcher with the Advanced Machine Learning Laboratory. Also, he has had the privilege of collaborating with several prominent research laboratories around the globe, including the Computer Vision & Pattern Recognition Laboratory, University of Asia Pacific, Bangladesh, and the Database System Laboratory, The University of Aizu, Japan. With an extensive research background, he has authored over ten articles in high-impact journals, such as IEEE ACCESS, *Sensors*, *Computer Systems Science and Engineering*, *Biology*, and *Mathematics*. In addition, he has contributed to the scientific community by publishing over ten conference papers and actively participating in well-established conferences, including IEEE HONET, ICCIT, ICIEV, icIVPR, DASA, BIM, and ICSCT. Moreover, some of his research work has been published as a chapter in a few esteemed books related to machine learning and AI. His research interests include artificial intelligence, machine learning, deep learning, computer vision, the IoT, and robotics.

JUNGPIL SHIN (Senior Member, IEEE) received the B.Sc. degree in computer science and statistics and the M.Sc. degree in computer science from Pusan National University, South Korea, in 1990 and 1994, respectively, and the Ph.D. degree in computer science and communication engineering from Kyushu University, Japan, in 1999, under a scholarship from the Japanese Government (MEXT). He was an Associate Professor, a Senior Associate Professor, and a Full Professor with the School of Computer Science and Engineering, The University of Aizu, Japan, in 1999, 2004, and 2019, respectively. He has coauthored more than 300 published papers for widely cited journals and conferences. His research interests include pattern recognition, image processing, computer vision, machine learning, human–computer interaction, non-touch interfaces, human gesture recognition, automatic control, parkinson's disease diagnosis, ADHD diagnosis, user authentication, machine intelligence, handwriting analysis, recognition, and synthesis. He is a member of ACM, IEICE, IPSJ, KISS, and KIPS. He served as the program chair and as a program committee member for numerous international conferences. He serves as an Editor for IEEE journals, *Sensors* (MDPI) and *Electronics*, and *Tech Science Press*. He serves as a reviewer for several major IEEE and SCI journals.

M. F. MRIDHA (Senior Member, IEEE) received the Ph.D. degree in AI/ML from Jahangirnagar University, in 2017. He is currently an Associate Professor with the Department of Computer Science, American International University-Bangladesh (AIUB). Before that, he was an Associate Professor and the Chairperson with the Department of CSE, Bangladesh University of Business and Technology. He was a Faculty Member with the CSE Department, University of Asia Pacific, and the Graduate Head, from 2012 to 2019. His research experience within both academia and industry, results in over 120 journals and conference publications. His research work contributed to the reputed journals of *Scientific Reports*, *Nature*, *Knowledge-Based Systems*, *Artificial Intelligence Review*, IEEE Access, *Sensors*, *Cancers*, and *Applied Sciences*. For more than ten years, he has been with the master's and undergraduate students as a supervisor of their thesis work. His research interests include artificial intelligence (AI), machine learning, deep learning, natural language processing (NLP), and big data analysis. He has served as a program committee member in several international conferences/workshops. He served as an Associate Editor for several journals including *PLOS One* journal. He has served as a Reviewer for reputed journals and IEEE conferences, such as HONET, ICIEV, ICCIT, IJCCI, ICAEE, ICCAIE, ICSIPA, SCORED, ISIEA, APACE, ICOS, ISCAIE, BEIAC, ISWTA, IC3e, ISWTA, CoAST, icIVPR, ICSCT, 3ICT, and DATA21.