Vision-Based Human Fall Detection and Severity Prediction Using Deep Learning Techniques

By

Kainat Sarfraz

NATIONAL UNIVERSITY OF MODERN LANGUAGES
ISLAMABAD
May 2025

Vision-Based Human Fall Detection and Severity Prediction Using Deep Learning Techniques

By

Kainat Sarfraz

MS-Computer Science, National University of Modern Languages, Islamabad, 2025

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

In **Computer Science**

To

FACULTY OF ENGINEERING & COMPUTING

NATIONAL UNIVERSITY OF MODERN LANGUAGES

ISLAMABAD

THESIS AND DEFENSE APPROVAL FORM

The undersigned certify that they have read the following thesis, examined the defense, are satisfied with overall exam performance, and recommend the thesis to the Faculty of Engineering and Computing for acceptance.

Thesis Title: Vision-Based Human Fall Detection and Severity Prediction Using Deep Learning Techniques

Submitted By: kainat Sarfraz **Registration #:**76 MS/CS/F22

Master of Science in Computer Sciences (MSCS)

Title of the Degree

Computer Science

Name of Discipline

Dr. Moeenuddin Tariq _____

Research Supervisor Signature of Research Supervisor

Anab Batool Kazmi _____

Research Co-Supervisor Signature of Research CO-Supervisor

Dr. Fazli Subhan _____

Head of Department (CS) Signature of HoD (CS)

Dr. Noman Malik _____

Name of Dean (FE&CS) Signature of Dean (FE&CS)

AUTHOR'S DECLARATION

I Kainat Sarfraz

daughter of Sarfraz Ahmed Abbasi

Registration no.76 MS/CS/F22

Discipline Computer Science

Candidate of Master of Science in Computer Science at the National University of Modern Languages do hereby declare that the thesis Vision-Based Human Fall Detection and Severity Prediction using Deep Learning Techniques submitted by me in partial fulfillment of MSCS degree, is my original work and has not been submitted or published earlier. I also solemnly declare that it shall not, in the future, be submitted by me for obtaining any other degree from this or any other university or institution. I also understand that if evidence of plagiarism is found in my thesis/dissertation at any stage, even after the award of a degree, the work may be canceled and the degree revoked.

_____

Signature of Candidate

Kainat Sarfraz

Name of Candidate

_____

Date

ABSTRACT

Vision Based Human Fall Detection and Severity Prediction using Deep Learning Models
In the healthcare industry, Falls have become a significant societal concern. In the last few years fall detection among senior citizens has gained a huge interest of researchers due to its increasing rate and serious consequences. Older adults who fall frequently suffer serious injuries and disabilities that may make it impossible for them to resume their prior level of function. These falls can have disastrous effects. Advancements in vision-based technologies, such as deep learning, have led to considerable improvements in action recognition, particularly for detecting falls. Traditional techniques sometimes depended on manually developed features, which limited the system's adaptability and performance. Therefore, we present a vision-based, fall detection, and severity prediction system using a long-term recurrent convolutional neural network. Additionally, this work introduces a novel approach to predicting severity after fall detection. Firstly the Fall dataset was trained for fall detection over the LRCN model. Once the fall detection model is trained and there is a fall event this will pass to the severity prediction model that was trained over the severity dataset will determine the severity of the occurred fall. The proposed technique, which leverages LRCN to evaluate generalization and sensitivity to overfitting, obtained Precision fall 92.0%, not-fall 97.0%, Recall fall 97.0%, not-fall 90.0%, F1-score fall 94.0%, not-fall 93.0%, and Accuracy fall 97.1%, not-fall 90.3% for fall detection. The suggested model outperforms other models, such as 3DCNN, 3D-CNN-LSTM, and fully connected Neural networks, for detecting falls. To the best of our knowledge severity prediction is the novel approach and we achieved In terms of precision (92.0% for severe and 83.0% for non-severe), recall (82.0% for severe and 92.0% for non-severe), F1-score (88.0% for severe and 88.0% for non-severe), class accuracy (82.0% for severe and 92.3% for non-severe), and confidence (97.97%) respectively.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	-	Artificial Intelligence
Cv	-	Computer Vision
NN	-	Neural Network
CNN	-	Convolutional Neural Network
LSTM	-	Long-short Term Memory
FCNN	-	Fully Connected Neural Network
DL	-	Deep Learning
LRCN	-	Long-Term Recurrent Convolutional neural network
FDD	-	Fall Detection Dataset
SD		Severity Dataset

LIST OF SYMBOLS

P	-	Precision
R	-	Recall
$F1$	-	F-1 score
A	-	Accuracy
σ	-	Confidence

ACKNOWLEDGMENT

First of all, I wish to express my gratitude and deep appreciation to Almighty Allah, who made this study possible and successful. This study would not be accomplished unless the honest espousal was extended from several sources for which I would like to express my sincere thankfulness and gratitude. Yet, there were significant contributors to my attained success and I cannot forget their input, especially my research supervisor, Asst. Prof. Dr. Moeenuddin Tariq and Ms. Anab Batool Kazmi, who never stopped guiding me along the way, were one of many steadfast supporters and important contributions to my accomplishment. I want to express my gratitude to my supervisor in particular for his help, direction, and insightful comments on the research.

Last but not least, I won't forget the management of the Department of Computer Sciences' generous assistance in helping me to complete this research. Many thanks to everyone whom I didn't include but who helped me stay motivated and made efforts I won't forget.

DEDICATION

This thesis is dedicated to my family for their endless love, support, and encouragement, without whom this journey would not have been possible. To my mentors, for their invaluable guidance and inspiration. And to all those who have believed in my vision, thank you for your faith and patience throughout this endeavor.

CHAPTER 1

INTRODUCTION

1.1 Overview

Falls among the elderly can cause serious injury or death if urgent aid is not given. According to the World Health Organization (WHO), a fall occurs when someone inadvertently falls to the ground, floor, or lower level. Falls can lead to disability and significant injuries, especially in older adults, who often do not regain their previous functional level after such incidents. Most injuries occur in the lower half of the body, upper limbs, head, and trunk; most are wounds or bruises, fractures, and displacements. The average rate of falls among adults older than 65 is 30% in the United States, 13.7% in Japan, 26.4% in China, and 53% in India. Falls are more common in elderly women than in males, according to research [1].

The fall of humans is an unconscious event that has different stages, i.e., sitting or standing positions, slipping or falling from a higher position to a lower vantage point, and relaxing. Several studies show that falling is one of the main causes of trauma for those with special needs, such as older people. 30% of the population over the age of 65 encounters one fall-related trauma incident every year; this reality is accepted by the World Health Organization (WHO) [2]. Furthermore, 47 percent of individuals who fall lose their independence. When an older person falls, the report shows that they need immediate help. This requirement caused a significant demand for falling detection devices. The fast advancement of surveillance video and technology for communication promotes such operations [3].

China has more than 158 million old citizens, and this sector is expected to grow in the coming years, as stated by the National Bureau of Statistics of China. Furthermore, Half of them are "empty nest seniors" who live alone. Falling is not an issue for teens, but for elders, it leads to a dangerous injury that includes a fracture in the hip, traumatic brain injury, as well as heart failure or heart attack. If medical assistance or treatment can be provided quickly, then one can be saved from serious injuries. The majority of older people live alone which makes it difficult for their relatives to know that they have fallen However, an alert system is designed in a way that it can transmit a signal automatically when a fall is detected so their relatives or hospital administration can respond immediately to save their lives and reduce the chance of severe injuries [4].

The severity of a fall is determined by a person's age, gender, and physical and medical health. There are various techniques for fall detection. Certain of these approaches rely on information gathered by sensors, such as acceleration and vibration sensors. Human movement, like sound and vibration, is linked to identifying falls. But these approaches have limitations over performance. For example, the floor vibration sensor is confined to the region that has such sensors, and surrounding disturbance can impair the accuracy of acoustic sensors. [3].

Another way to collect data for fall detection is by video sequencing. These methods use stereo-pair cameras, single cameras, multiple cameras, and bidirectional cameras. Information that is obtained by the camera is more informative and broader than that gathered by typical sensors. Vision-based fall detection not only saves one's life but also reduces the medical cost associated with fatal falls, especially in older people [4].

In recent times, a variety of methodologies have been introduced, utilizing disparate sensor technologies that exhibit a range of performance characteristics[5, 6, 7]. Wearable sensor-based systems and computer vision-based systems are the two main categories of human fall detection systems[8, 9, 10]. Wearable sensors offer high accuracy at a lower cost but are intrusive, whereas vision-based systems, leveraging deep learning and machine learning algorithms on time series data, provide high precision albeit being bulky and less unobtrusive. [3].

Although both categories have drawbacks, vision-based techniques are more attainable. Computer vision-based algorithms are simple, so it is easy to implement on security cameras and can easily differentiate between falls and no falls by examining the topic. Vision-based techniques are stronger than sensor-based systems. This technique not only differentiates falls but also displays the entire picture that helps to recognize the series of events that caused the fall. These series of events can then be utilized to build a fall prevention system and also to avoid mishaps

like damaged staircases, a slick floor, fire incidents, security breaches, and many more [11]. Artificial neural networks and supervised and unsupervised learning hierarchical probabilistic models are the different approaches to deep learning. Deep learning consists of several stratified structures that allow such structures to gather information at various levels of abstraction, which makes it distinct. They can understand massive amounts of complex data and provide valuable details. Artificial neural networks (ANN), convolutional neural networks (CNN), and long short-term memory (LSTM) are the most common and used techniques of deep learning that exceed state-of-the-art approaches in fields like image processing, natural language, and other sensor-based tasks. Deep learning plays a vital role in advancement in the field of computer vision, including object detection, activity recognition, semantic segmentation, and motion tracking [12].

A collection of unbroken frames or pictures that generates an even and continuous visual in place of someone's observation is called a video. Convolutional neural networks have a wide range of applications in video, including video categorization and analysis, human posture prediction, and human detection. The time aspect or temporal dimension in videos creates complexity. The motion of an object in a video is determined using 3D-CNN, which takes a collection of frames as input and then passes them through 3-dimensional filters to the convolution so that both time-based and spatial data are employed in the comparable convolution [13].

This research work focuses on vision-based fall detection and predicting the nature of severity using deep learning approaches. Cameras are placed around the area, and the approach observes every move of the individual, and in case of any unusual move, it indicates a sign of a fall. Moreover, after the fall detection, the model will predict the severity as major or minor. In the healthcare industry, deep learning has state-of-the-art technologies. Our model uses the video-based system to keep an eye on individual movement, specifically fall events that result in serious injury or even death. Based on the posture of the fallen person, our model analyzes video frames using deep learning techniques to recognize the falls and their severity.

1.2 Problem Background

The traditional fall detection systems mostly depend on sensor-based devices, such as accelerometers and gyroscopes, to detect sudden changes in motion that indicate a fall. Although

these systems perform well in certain situations, they have certain limitations that prevent their widespread effectiveness, specifically among people with intellectual disabilities or those unwilling to wear these devices.

1.3 Problem statement

Falls are the utmost public health dilemma. Individual age, gender, and physical and medical health can all have an impact on the kind and severity of injury. Although falling is a common phenomenon, sometimes it becomes injurious, which sometimes leads to serious injuries or might even cause death. People above the age of 65 are at great risk because if they fall and aid is not provided at the earliest time, they may encounter trauma or serious injury. This research aims to create an appropriate deep-learning model for human fall detection utilizing vision-based hybrid datasets. Research also aims to develop a deep learning-based model that can accurately predict the severity of falls after detection of falls based on vision. The proposed model aims to categorize the hybrid dataset into two major datasets, a fall dataset and a severity dataset, and after pre-processing, categorize falls as fall or not fall and severity, such as minor and major, based on video data. This involves selecting appropriate deep learning algorithms, feature engineering, and model training to achieve high prediction accuracy. The aim is to establish a high-quality dataset suitable for training and evaluating the video-based prediction model.

1.4 Research Questions

1. What type of deep learning model is suitable for a human fall detection system by leveraging vision-based hybrid datasets to ensure robust generalizability?
2. How can an advanced deep learning model based on vision be designed and developed to accurately predict the severity of falls after detecting a fall event?

1.5 Aim of the Research

This research work focuses on vision-based fall detection and predicting the nature of severity using Deep learning approaches to increase the safety and standard of living for senior citizens and others who have mobility issues.

1.6 Research Objectives

1. To determine the most suitable deep learning model for a vision-based human fall detection system, leveraging hybrid datasets to ensure robust generalizability.
2. To develop an advanced deep learning model based on vision that accurately predicts the severity of falls following fall detection.

1.7 Scope of Research Work

The scope of the research work encompasses several areas, including:

- Using cameras to gather data.
- Creating algorithms to process the data and extract relevant features.
- Using the attributes to train deep learning models.
- Developing systems that can react to falls instantly, including alerting caregivers or calling for help.
- Using tests and comparisons with other fall detection systems to demonstrate the efficacy and accuracy of the system.

1.8 Thesis Organization

The rest of the thesis is organized as follows:

- Chapter 2 outlines the research domain and offers a thorough overview of the available literature. It classifies literature based on multiple strategies and techniques, including models, algorithms, and methodologies. Chapter 2 outlines restrictions and problems for future study, resulting in a modified human fall detection and severity prediction model.
- Chapter 3 describes the research approach, which includes benchmarking methodologies and strategies for overcoming existing limitations. It describes the implementation tools and assessment methodologies for the proposed fall and severity prediction model. This chapter is important since it provides a thorough description of the model design, including the processes, models, and tools employed. It also covers data details, processing needs, and algorithms.
- Chapter 4 will present an evaluation of the proposed model and the parameters used to evaluate the model design. This chapter compares the proposed model to benchmark datasets to ensure the architecture's efficacy. The accuracy achieved utilizing the suggested technique when compared to other models is shown in the form of tables.
- Chapter 5 gives an overview of the contributions of this research study, with ideas for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

There are various techniques for detecting human falls, but essentially, there are two main categories of human fall detection systems: sensor-based and vision-based [14]. This section is composed of a detailed review of sensor-based and vision-based papers.

2.1.1 Sensor-Based Human Fall detection

Jain et al.[15] employ deep learning and wearable sensors to detect pre-impact falls, achieving 97.52% accuracy using standard datasets SisFALL and KFall. They extract temporal features, manually calculate temporal labels, standardize, fuse, and normalize data before feeding it into the network. However, the study's limited testing on an individual robotic platform with sparse training data hampers its generalizability.

The critical issue of fall detection in older adults is addressed by Kabir et al.[16], with concerns regarding current systems, such as sensor placement and the impact of user activity being highlighted. Utilizing SisFall and UMAFall datasets, they employ a deep learning approach with a feature extraction module and three Long-Short-Term-Memory models. Achieving a 96.45% accuracy, the study meticulously evaluates performance metrics, though limitations include sensor placement variability, real-world fall variability, computational complexity, and

constraints of emulated falls datasets.

Al_Hassani et al. [17] enhance healthcare alert systems by predicting motion turbulence, achieving an impressive 98.615% accuracy using deep learning techniques like auto-encoding and 4SFNN-PSO. Utilizing an experimental dataset with fall feature characteristics from sensors, the study acknowledges limitations in human behavior complexity and the need for real-world validation.

The real-time fall detection challenges with wearable sensors are explored by Yhdego et al.[18], with a focus on gait variability and sensor placement, leading to the identification of the shank as the optimal location. Their study employs diverse datasets and investigates various segmentation techniques, achieving improved detection through convolution-based features. Despite early testing on a limited subject pool and reliance on video annotation, the research suggests that future work should expand subject testing and explore unsupervised learning methods.

2.1.2 Vision-Based Human Fall Detection

Alanazi T et al. [3] address the issue of human fall detection, which is an important concern in public health, specifically for elderly people who are at risk of serious injuries that lead to disability, whether temporary or permanent. This article focuses on an automatic system for detecting human falls using multi-stream CNN with fusion. Authors utilize Le2i fall detection datasets that are openly accessible. The methodology involves multi-stream 3-D CNN with fusion, where each stream corresponds to one of the stages, like standing, walking, falling, fallen, and resting. The methodology employs a multiple-stage image fusion methodology to analyze movement variation across 16 frames of video data. By utilizing three-fold cross authentication to validate the findings, the technique obtained high accurateness, sensitivity, specificity, and precision with 99.03%, 99.0%, 99.68%, and 99.0%, correspondingly. The presented model outperforms other SOTA models for fall incident detection, including GoogLeNet, Squeezenet, ResNet18, and DarkNet19.

The paper by Inturi A. et al. [11] presents a fresh visualization-based approach for fall detection via key points of the human skeleton. This paper proposes a new method to detect falls using videos from cameras. The method analyzes the position of joints like shoulders, hips, and knees between frames to see how a person moves. This author's paper uses UP-Fall detection datasets to validate their fall detection system. The proposed method involves using a pre-trained

network to acquire the key points, processing them through Convolutional neural network (CNN) layers, and employing long short-term memory (LSTM) architecture to preserve long-term dependencies. The proposed system achieved commendable results when compared to SOTA approaches. Different performance metrics, i-e, accuracy, precision, recall, specificity, and F1 score, were used to evaluate model performance. The result shows high accuracy, precision, recall, specificity, and an F1 score that indicates the effectiveness of the proposed approach.

The algorithm suggested by [19] uses OpenPose for the goal of estimating human poses. Top-down detection and bottom-up detection are two separate processes that go into the detection of body position. The person is first detected in the top-down detection procedure before the important points are found. The bottom-up openpose method, on the other hand, adopts a different strategy. The 80 target categories in the Coco Dataset are pre-trained using the MobileNet Network. The SSD MobileNet CNN is used for object detection and the deletion of non-human portions. Furthermore, a pre-training model that is then applied to the SSD network is created using a self-built human dataset. The settings are optimized using a grid search technique and the SVDD anomaly detection approach.

In the work proposed by [20], preprocessing and feature extraction/classification are the two primary steps in the article's suggested solution for fall detection in videos. Using rank pooling, the preprocessing stage converts video input into dynamic optical flow pictures, capturing motion connected with the activity and reducing the problem of recording unrelated pixel intensities. The VGG-16 CNN architecture is used to extract and classify features from dynamic images during the feature extraction and classification stage.

Alanazi T et al. [12] address several key aspects that are relevant to human fall detection. The paper presented a human fall detection system via Multi-streamed 3-D CNN through fusion. Furthermore, the model processes live images as of an observing surveillance camera by applying the fine-tuned human segmentation system and the image fusion approach. The other aspect discussed in this paper is about pre-processing of video sequences, model architecture, database description, datasets preparation, and results of experiments.

The Paper proposed a new approach for fall detection by utilizing a grouping of dense spatial-time-based graph convolution network (DST-GCN) and lightweight Open Pose. The authors address the importance of fall detection for the safety of aged people, and the aim is to improve the accuracy and efficiency of fall detection systems. The UP-Fall dataset is used in the research. The presented approach includes two main components: DST-GCN and lightweight Open Pose.

By creating a dense graph model of the human skeleton and executing graph convolution operations, DST-GCN captures the spatial and temporal interactions between body joints in a video clip. An open Pose is a lightweight software that predicts the 2D locations of body joints from video frames in real-time. This network's accuracy on the MCF dataset is 96.3%, while its accuracies on the two NTU RGB + D assessment standards are 85.6% and 93.5%, respectively Zhang X et al. [21] .

An innovative approach to automatic human fall detection in smart homes for elderly monitoring is presented by Jeffin Gracewell J et al. [22]. The paper emphasizes the importance of enhancing elderly care and reducing risks when they're alone. Using computer vision techniques, a hybrid model learns features automatically from a dataset comprising falls and normal activities. Keyframes are classified through a two-stream process and validated when in agreement; otherwise, additional data aids in classification. This unique integration of spatial and temporal features, employing a two-stage SVM classifier, proves robust even with limited training samples, outperforming existing methods and reducing errors. The study suggests potential improvements with larger datasets and multi-view cameras, promising enhanced efficacy for fall detection and better care in smart home settings.

Lian Wu et al. [23] provide a unique strategy for fall detection based on fusion approaches, which analyzes several fusion strategies on two datasets, Le2i and RFDS. The technology outperforms existing fusion approaches for fall detection, as evidenced by qualitative outcomes such as fall score, curve, and video frames. Data annotation effort, despite reduced workload, and the importance of training models for new scenarios are limitations.

Several deep learning approaches are compared with the YOLOK + 3DCNN strategy to identify human falls by Gomes M et al. [24]. On the AVA dataset, YOLOK, 3DCNN, and LSTM are utilized, achieving a recall of 0.9803 and an area under the curve (AUC) of 0.8454. However, issues in detection with the AVA dataset are encountered by the model.

The work presented by [25] uses a posture estimation network with raw RGB input. The ADLF Dataset (in-house) is utilized to preprocess the skeleton's coordinates before they are inserted in a window-style design. To learn the spatiotemporal dynamics present in the data, CNN and GRUS are used. The fully linked layer receives the GRUS output and classifies it. There is 96.7% accuracy.

The model presented by [26] uses the falling state and the fallen state as two different types of falling states. Preprocessing, fall event modeling, state categorization, and fall event detection

are some of the procedures that are involved. The 21,944 photos that make up the fall detection dataset are separated into the following groups: crawling, stooping, bending, standing, and sitting. With the le2i Fall Detection dataset, Openpose and Yolo are used for preprocessing, background subtraction, and body position extraction (191 videos). The dual-channel sliding window approach uses two classifiers and detects two different feature types. The human body keypoint location is extracted using Openpose to create the static feature. The rate of the core drop and the speed of the higher limbs are used to build the dynamic feature. A sliding window model is used to continuously extract new features from the movies while removing old ones.

Harrou F et al. [27] combine GLR detection with SVM classification in an integrated vision-based technique for human fall detection at home. URFD and FDD, two publicly available datasets, are used. However, employing pixel-based characteristics and body partition ratios, the approaches efficiently recognize and categorize falls. The GLR-SVM method beats previous classifiers. This paper's drawbacks include its reliance on RGB cameras, potential accuracy issues, and limited classifier comparisons across different scenarios.

Keskes O et al. [28] describe a fall detection system that uses transfer learning from action recognition and Kinect v2 camera skeletal data to identify falls, although it has drawbacks owing to inaccurately portrayed falls and a lack of occlusion scenarios in datasets. It outperforms previous approaches on the NTU-RGBCD, TST Fall detection v2, and Fall-free datasets, demonstrating resilience while highlighting dataset limits and urging for more diverse and realistic datasets for higher accuracy. The technique, which uses RGB-D and skeletal data with the ST-GCN algorithm and transfer learning, shows potential while emphasizing the need for datasets that mirror real-life circumstances for greater resilience.

The extraction of skeletal joints by Kinect's technology is employed in the system proposed by Mansoor, M. et al. [29]. The Software Development Kit for Kinect (SDK) is utilized to extract the skeletal pictures. Subsequently, the head and foot joints are used to generate a bounding box. The authors generated a dataset to classify falls into various categories. The k-Nearest Neighbors (KNN) model is created to classify events into two categories: those that involve falling and those that do not. The standard deviations determined for each activity in the dataset comprising 150 activities, along with their corresponding SD values, are used to categorize the data.

The strategy proposed by [30] uses the HCAE-FD approach. A multi-task system employs this intermediate characteristic for fall detection as the primary work and frame reconstruction as the auxiliary task. An encoder and a decoder are features of the neural network known as the HCAE.

The suggested technique, known as HCAE-FD, prevents feature loss by employing a shallow-layer network with three hourglass convolutional layers. To detect features like faces and hands and to comprehend general human activity, the suggested HRU in HCAE is employed to collect visual information at various scales. To compare the error value, the HCAE has an encoder and a decoder that compress the input into an intermediate feature, restore the approximate original frames, and compress the input once again. To reconstruct the original frames, the decoder functions as a weak supervisor and corrects the intermediate feature extracted by the encoder more effectively. Input of 10 consecutive frames as a stack into the HCAE encoder is required for both the training and testing phases of the method to make use of temporal information expected and actual outputs is computed.

In [31], McCall et al. introduce a novel transformer model approach for fall detection and prediction tasks, utilizing transfer learning to enhance performance. The model leverages 2D human skeletons extracted from video clips, pre-trained on a large dataset of human motion data to learn useful representations. Fine-tuning specific fall-related data further refines the model. Experimental results demonstrate superior performance compared to traditional machine learning and deep learning methods. Future research will explore incorporating additional video features and advanced transformer architectures to further improve fall detection and prediction accuracy, addressing critical public health concerns surrounding falls.

"YOLOv7-fall" has been recently introduced by Zhao et al.[32], an enhanced convolutional neural network model tailored for prompt fall detection, addressing challenges of accuracy, parameter count, and computational load. It incorporates novel attention modules and optimization techniques to improve feature extraction and reduce model complexity, resulting in improved detection accuracy and reduced computational requirements. Future directions include refining the model with dedicated datasets, enhancing detection accuracy while maintaining lightweight characteristics, and exploring practical deployment through embedded systems in industrial safety applications.

A novel approach for identifying human falls in surveillance films is presented by Vishnu C et al. [33], utilizing a fall motion mixture model (FMMM) and component analysis to extract essential features. Various surveillance video datasets, including narrow-angle (Le2i dataset), wide-angle (URFall dataset), and multiple cameras (Montreal dataset), were used to validate the proposed fall detection system. The approach demonstrates improved human fall detection compared to existing methods. However, due to comparable visual signals, the approach has

issues differentiating minor differences between some fall and non-fall occurrences.

Table 2.1: Summary of the vision-based fall detection human fall detection Literature Review

Ref.#	Year	Dataset	Techniques	Results	Limitations
[3]	2022	Le2i fall detection datasets	Multi-stream 3-D CNN fusion approach	An exceptional level of performance was attained, with an accuracy of 99.03%, sensitivity of 99.0%, specificity of 99.68%, and precision of 99.0%, achieved through a rigorous three-fold cross-validation procedure	Narrow focus on comparisons primarily using the Le2i dataset, which might limit the method's generalizability across diverse scenarios and datasets, potentially overlooking its performance in various real-world environments
[11]	2023	UP-Fall detection datasets	Convolutional Neural Network (CNN) layers Long Short-Term Memory (LSTM)	Outperforms state-of-the-art (SOTA) methods. High recall and F1 score.	Achieved lower sensitivity, specificity, and accuracy.
[12]	2023	Le2i dataset	3D Multi-Stream Convolutional Neural Network (CNN) with an Image Fusion Technique	Accuracy 99.44%, sensitivity 99.1%, specificity 99.12% precision 99.59%.	Detects falls only in scenes with a single person and lacks localization of the person in the video. Evaluation tests were limited to the Le2i fall detection dataset.

[21]	2023	UP-Fall	DST-GCN and lightweight Open Pose integration	accuracy on the MCF dataset is 96.3% NTU RGB + D dataset are 85.6% and 93.5%	Computational Complexity Decline in Accuracy Model Comparison.
[22]	2021	FDD dataset	Two-stage SVM Classifier	High accuracy, error rate, sensitivity and specificity	Lack of evaluation under different lightning conditions, i.e., night vision and real-time data collection.
[23]	2023	Le2i and RFDS Datasets	Dual-modal network integrating RGB and optical flow streams and Integrating a fusion strategy for final detection.	Achieves best results among all strategies	The data annotation workload model needs to train with relevant dataset when applied to new scenario.
[24]	2022	AVA dataset	YOLO object detection, Kalman filter, 3D CNN, LSTM	Recall 0.980, Area under curve (Auc) 0.85	Fall misclassification during focused body region scenes, Rapid or awkward sitting actions, Partial visibility, Occlusion scenarios, impacting accuracy.
[27]	2019	URFD and FDD datasets	GLR detection SVM classification	Outperforms prior classifiers in detecting falls	Using RGB cameras only, Possible accuracy concerns, Restricted classifier comparisons across many scenarios.

[28]	2021	NTU-RGB+D, TST v2, and Fallfree datasets,	RGB-D and skeleton data with ST-GCN algorithm, transfer learning.	Outperformance of previous models.	Unrealistic falls and interference circumstances have an impact on system representation.
[31]	2024	2D human skeletons, larger human motion dataset	Transformer model, transfer learning, fine-tuning	Superior performance	Solely 2D pose features
[32]	2024	Not specified	YOLOv7-fall, SDI, GSConv, VoV-GSCSP, DBB	Improved mAP, reduced parameters, decreased computational requirements	False negatives
[33]	2021	Narrow-angle (Le2i dataset), wide-angle (URFall dataset), and multiple cameras (Montreal dataset).	Histogram of optical flow (HOF), Motion boundary histogram (MBH), Fall motion vector modeling (FMMM).	Human fall detection improved across many surveillance video datasets.	Difficulties in understanding variable-length patterns in human fall videos; minor differences between fall and non-fall occurrences.

2.2 Research gap and challenges in existing vision-based elderly fall detection

Research gaps and challenges in vision-based elderly fall detection and severity prediction include maintaining privacy while monitoring, improving algorithm robustness to varying lighting

and occlusions, and lowering computational requirements for real-time processing on resource-constrained devices. Furthermore, resolving ethical concerns about informed consent and data security is critical to the broad use of these devices.

2.2.1 existing vision-based elderly fall detection challenges

Gaps

Existing vision-based senior fall detection systems have limitations such as low computing efficiency, inability to generalize across varied surroundings, privacy issues over camera use, and a lack of defined benchmark datasets for performance measurement.

Significance

Regardless of these drawbacks, vision-based methods offer numerous advantages, such as unobtrusive monitoring that improves user compliance, the possibility of early intervention to reduce severe injuries, improved quality of life for the elderly, and contributions to the broader field of human action recognition, which has applications beyond fall detection.

2.3 Existing notable methods for fall detection

2.3.1 Sensor Fusion Techniques

This approach enhances the existing wearable, ambient, and vision-based fall detectors' performance, utilizing data collected from several sensors. Collecting several inputs decreases the number of erroneous alerts while improving the general performance of the system.

Drawback: The computational procedures may also become extremely intricate, which in return may lead to high costs and make it difficult to implement the system.

2.3.2 Depth Camera Systems

For fall detection, the depth cameras, coupled with infrared sensors, allow the ability to detect falls during short-light conditions. These systems quantify movement of the human body about the environment to aid in the detection of the event of a fall.

Drawbacks: Depth cameras are expensive, and in addition, the cameras may be limited by barriers or clutter in the monitored space.

2.3.3 Wearable Device Integration

Smart wearables that incorporate motion sensors, including an accelerometer and a gyroscope, can continuously assess physical activity and distinguish falls based on motion differences.

Drawbacks: One big issue is user rebellion: the elderly may sometimes forget to wear their devices or charge them as often as needed, which disrupts care.

2.3.4 Machine Learning Algorithms

New advancements have made it possible for machine learning algorithms to be developed with learning capacity, especially in response to particular user actions, thus improving the accuracy of the fall detection, as well as the severity of the fall, the longer the model is active.

Drawbacks: These algorithms have to be trained on big datasets, which are not easy to get, and even then, they may make predictions wrongly and give a false positive or a false negative.

2.4 Summary of Chapter 2

Fall detection technology plays a crucial role in promoting safety and well-being for elderly individuals and those with disabilities. This technology utilizes a multifaceted approach that leverages various techniques and algorithms. Feature extraction separates key characteristics from sensor data, while fall categorization classifies the type of fall that has occurred. Machine

learning algorithms play a significant role in fall detection, and camera-based systems offer an additional layer of analysis. Fusion techniques combine data from multiple sources to enhance accuracy. Research suggests the effectiveness of systems employing the architecture of convolutional neural networks (CNN), posture estimation networks, OpenPose, and support vector machines (SVM). These systems preprocess sensor data, extract relevant features, and classify the information in real time to identify falls. When a fall is detected, the system triggers emergency service notifications or alarms to prevent severe consequences. Following extensive research and development, comparisons of these various fall detection systems have been made readily available for implementation, empowering individuals and caregivers with invaluable safety solutions.

CHAPTER 3

METHODOLOGY

3.1 Overview

In the past few decades, computer vision and deep learning have gained the prominent attention of researchers for video and image processing to perform challenging tasks such as human fall detection. This section reviews the proposed approach for the detection of human falls and the prediction of severity. After that, we describe the data set utilized for the experiments and analyze the proposed approach. Next, we demonstrate how we preprocess video sequences using our technique. Finally, we provide a detailed description of the suggested classification model that finds occurrences of human falls and identifies the severity level in a particular video clip.

In wearable sensor-based techniques, accelerometers and gyroscopes detect falls. Vision-based methods involve cameras and computer vision algorithms to analyze posture and motions, and ambient sensor-based methods involve environmental sensors to detect falls without wearables. Multisensor data fusion increases the effectiveness of detection and simultaneously decreases the number of false alarms. Vision-based strategies are beneficial because the elderly do not need to wear gadgets, thus solving compliance issues. They can provide whole-body coverage in various settings while preserving privacy using sophisticated image processing.

3.2 Operational Framework

The operational architecture depicted in the 3.1 demonstrates a systematic way of handling a project centered on elderly fall detection and severity prediction. This framework has three main stages: the analysis phase, the design and development phase, and the discussion phase of the analysis of the results. All these phases are crucial to the development and evaluation of this system.

The analysis phase starts with a literature review of approaches to identifying falls in the elderly and predicting the severity of falls. This step involves an in-depth, detailed look at the existing innovations, computations, and frameworks to get it to the state of the art. After this examination, the shortcomings and gaps in the current techniques are noted, with an emphasis on problems with output accuracy, dependability, and general application. As a result of this thorough review, an extensive problem statement that outlines the particular difficulties the project seeks to solve is developed. To ensure that all ensuing phases stay focused on resolving the stated problem, the phase ends with the establishment of research objectives that will serve as project guidelines.

The primary system for fall detection and severity prediction is built throughout the design and development phases. The phase begins with a requirement analysis, where the hardware, software, data sources, and algorithms that are needed are identified and documented. Next, a dataset relevant to senior fall occurrences is either produced or acquired and then split into training and testing sets to help with the model's building. Next, two key models are created: the Fall Detection Model, which aims to identify falls using video or sensor data, and the Severity Prediction Model, which assesses the severity of recorded falls to provide insights about likely injuries and necessary medical care. After these models are developed, preliminary results are generated, which offer an evaluation of the models' functioning and point out areas that need more investigation.

The final stage, known as Result Analysis and Discussion, involves a thorough evaluation of the developed models. This includes a statistical and graphical examination of their performance, utilizing metrics such as accuracy, precision, recall, and F1 scores. To effectively demonstrate the models' performance, visual tools like confusion matrices and ROC curves are utilized. Subsequently, the results are compared with previous benchmarks or existing systems to deter-

mine whether the new approach yields significant improvements. This phase concludes with an in-depth analysis of the findings, a discussion of their implications for elderly care, and suggestions for potential future research or advancements in the domain of fall detection and severity prediction.

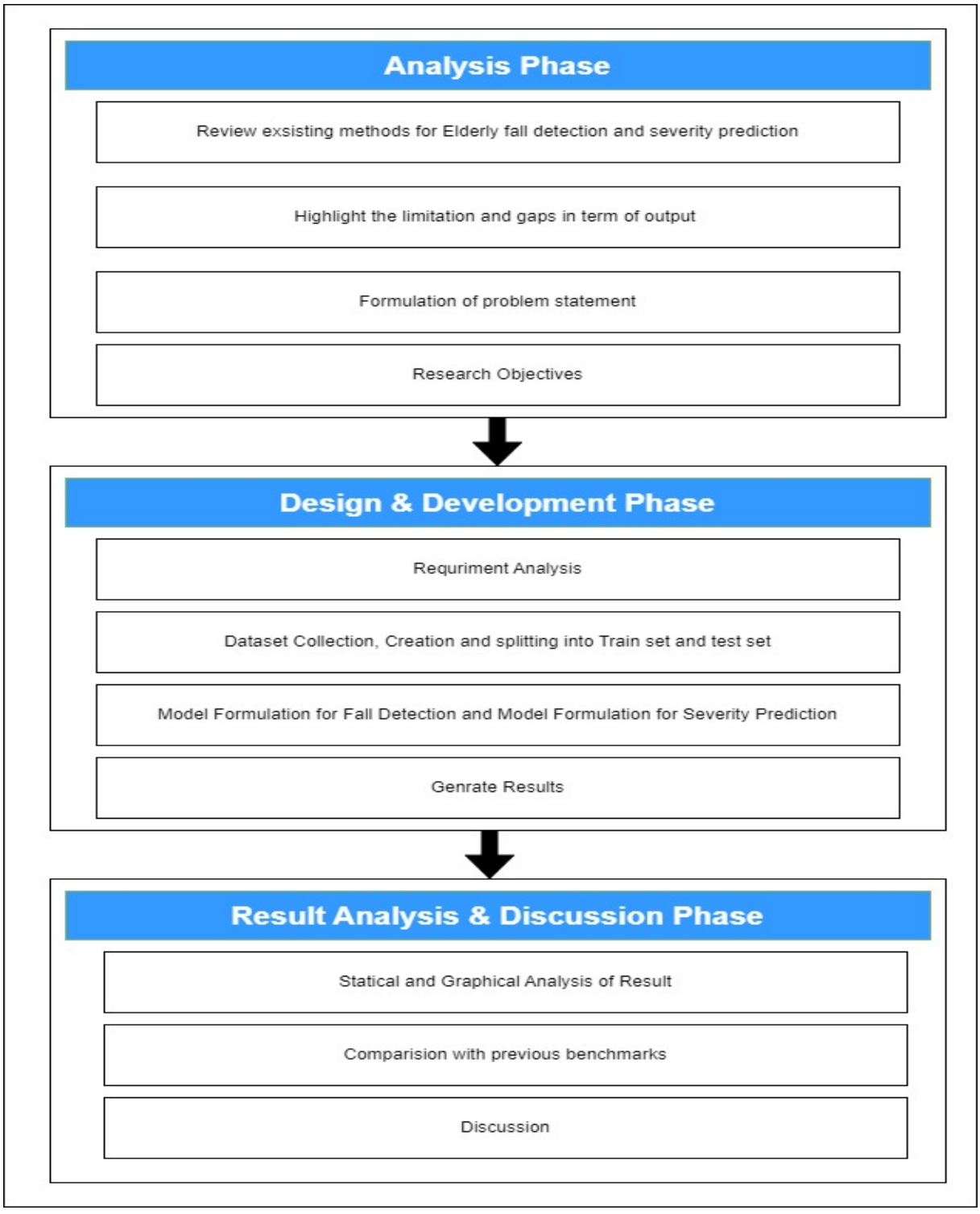

Figure 3.1: Framework of the Research Work

3.3 Requirement Analysis

The amount of resources needed for fall detection varies depending on various variables, such as the difficulty of the algorithms being used, the video data's resolution and frame rate, the processing capacity of the hardware being utilized, and the size of the dataset being analyzed. The following are some of the standard resource needs:

- A camera that can record video at a resolution and frame rate high enough to capture the amount of detail required for the intended application is referred to as having an appropriate resolution and frame rate.
- Systems that utilize image and video processing techniques to study human movements, identify whether a person has fallen, and make use of computer vision algorithms that can detect falls and categorize the severity of the fall. These algorithms normally operate by watching the movement of the subject in the video feed and searching for abrupt changes or disturbances that might be signs of a fall. For developing computer vision algorithms, we used OpenCV, and TensorFlow. OpenCV is used for preprocessing and real-time image processing, whereas TensorFlow is utilized to train and interpret deep learning models. This combination yields an effective, scalable, and high-performance system for vision-based fall detection and severity prediction. These libraries include pre-built tools and functions that are used to train deep learning models, implement the algorithms, and extract features from video feeds. OpenCV is more lightweight than PyTorch, while TensorFlow supports mobile and cloud deployments better.

3.3.1 Dataset

This research proposal showcased and tested algorithms for detecting falls in older people and predicting how serious they are. We used several well-known datasets to do this, including KFALL [34], UMFALL [35], CAUCA [36], URFALL [37], and MULTIPLE CAMERA [38], along with some online sources. These datasets have a lot of different types of information, such as data from sensors and video recordings. We used this data to train and check machine learning models.

KFall Dataset The KFall dataset, created in 2021 by Yu X, Jang J, and Xiong S, is a great

resource for developing fall detection for the elderly. It has 5,075 motion files from 32 young subjects performing 21 ADLs and 15 types of falls. Each fall is annotated with synchronized video references.[34]

UMAFall Dataset The UMAFall dataset has 12 subjects, 10 types of falls, and 14 types of ADLs. It has a wide range of mobility traces for fall detection research, but the number of video sequences is not specified.[35]

CAUCA Dataset Developed in 2022, the CAUCA dataset has 10 subjects simulating 5 types of falls and 5 types of ADLs, organized in structured directories for detailed analysis. Recorded at 23 fps with 1080×960 pixels.[36]

URFall Dataset The URFall dataset has 70 sequences from 5 subjects, 30 falls, and 40 ADLs. It has a balanced set of fall and non-fall scenarios to be used as a benchmark for fall detection algorithms.[37]

Multiple Camera Dataset The Multiple Camera dataset (2010) has a multi-camera setup for 24 scenarios and 22 falls. Good for testing algorithms that need coverage from multiple angles for fall detection.[38]

Internet websites We have utilized the videos, which are freely available over the internet. After

Table 3.1: Videos collection from datasets

Dataset	Total	Fall	Not Fall
KFALL	36	21	15
UMAFALL	11	3	8
CAUCA	100	50	50
URFALL	100	60	40
MULTIPLE CAMERA	178	131	47
Internet Websites	232	81	151

merging all of the above datasets and recorded videos together, we have created two new datasets, which are:

- Fall dataset The fall dataset is categorized into two main classes: fall and not fall.
- Severity dataset The severity data set is manually classified into two main classes, severe and non-severe.

Table 3.2: Fall Detection and Severity Datasets

Dataset	**Category**	**videos**	**Total**
Fall detection Dataset	Fall	346	657
	Not Fall	311	
Severity Dataset	Severe	285	610
	Non Severe	325	

3.3.2 System Requirements

System Requirements refer to the system capabilities necessary for the software to function properly. These requirements often include dependencies like a processor, operating system, storage space, etc. Several conditions were necessary to implement the suggested system in this research, which are stated below.

Laptop Hardware

For laptops, it is essential to have a reasonably powerful CPU and GPU, as fall detection and severity classification can be computationally intensive tasks. A modern multi-core processor (e.g., Intel Core i5 or i7) is recommended for handling the video processing and machine learning tasks.

PC Hardware

PCs typically offer more flexibility in hardware choices. Workstation-class GPU is considered for faster deep learning computations. PCs can also accommodate more RAM, which is essential for large-scale data processing and machine learning tasks.

Storage

Fast SSD storage is crucial for quick data retrieval, especially if the system is recording and storing video footage. Depending on data storage needs, the system needs several gigabytes of storage capacity.

Operating System

An operating system that supports the required software libraries for machine learning is chosen. Windows provides compatibility with popular libraries and frameworks like TensorFlow, OpenCV, and others, making it a viable choice for developing and running applications in the field of machine learning.

Python 2.7

Python 2.7 was launched in 2010 and was the most significant Python version. This Python version was used to implement the proposed model in this study.

Environment

Google Colab is a cloud-based collaboration environment that provides free access to GPUs/TPUs, pre-installed libraries, and seamless Google Drive integration, which makes it perfect for data science and machine learning projects.

3.4 Proposed Methodology

This proposed study aims to develop a vision-based approach for human fall detection and severity prediction using different deep learning techniques. After that, we selected the method based on the best results. Since the last decade, deep learning has gotten special attention due to its powerful applications in video and image processing and achieving the best results even for challenging tasks. On the other hand, wearable sensors are achieving good results but also have drawbacks, such as false alarms, inconsistent accuracy, user comfort, and short battery life. We opted to use a video-based solution. Based on the video and extra pre-processed severity data, the second component, the Severity Prediction Model, evaluates the fall's severity using a CNN and LSTM architecture akin to that of the first. After passing through many levels of processing, the outputs from both models yield the final predictions, which include whether or not a fall has occurred and how severe it was. With this structure, the system combines the advantages of CNNs for geographical data with LSTMs for temporal analysis to identify falls and assess their severity.

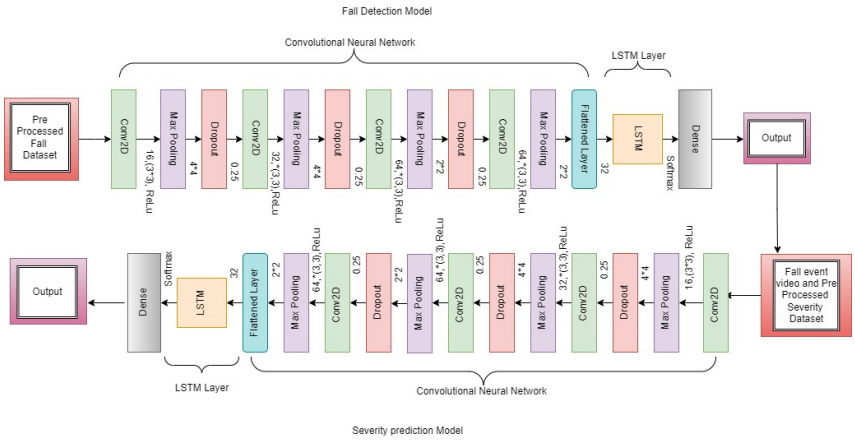

Figure 3.2: Proposed Model

A two-part neural network model for fall detection and severity prediction is shown in the 3.2. The Fall Detection Model, the first section, uses a sequence of Convolutional Neural Network (CNN) layers and then Long Short-Term Memory (LSTM) layers to handle pre-processed fall data. The model can now examine the video frames and determine whether a fall has taken place.

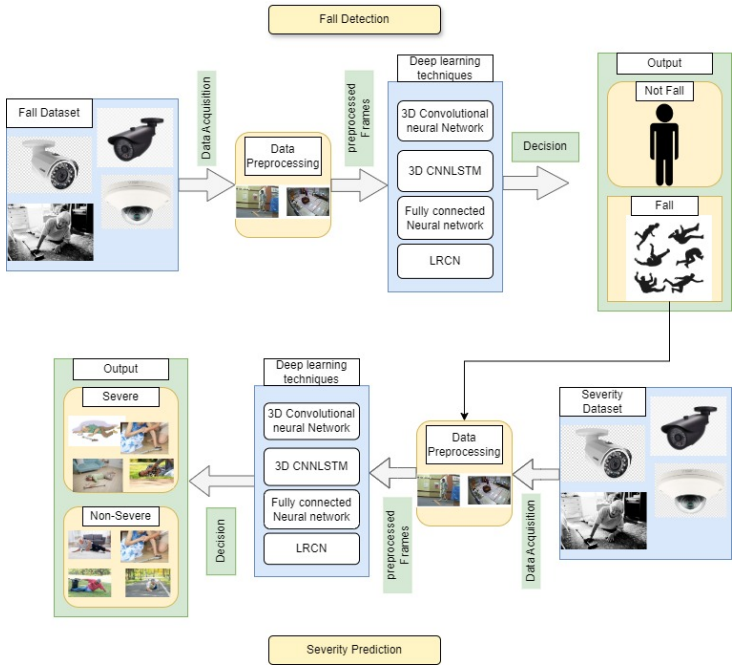

Figure 3.3: A simplified Block diagram of proposed human fall detection and severity prediction

Fig. 3.3 represents a sophisticated system for detecting falls and predicting severity using advanced deep-learning methodologies. The process initiates with the acquisition of video data from surveillance cameras. This raw video data undergoes preprocessing and is subsequently

analyzed through a series of deep learning models, including 3D convolutional neural networks (3D CNN), convolutional neural network long-short-term memory (CNN-LSTM), and long-term recurrent convolutional networks (LRCN). The system's primary objective is to detect the occurrence of falls accurately. Upon detecting a fall, the system further evaluates its severity, categorizing it into "severe" or "non-severe" classes. In the event of a severe injury, the next step is to generate a call to both the caregiver and medical professionals for assistance. Conversely, if the injury is deemed non-severe, the call is generated only to the caregiver. The system aims to enhance the accuracy of fall detection and severity assessment, offering significant potential for applications in healthcare monitoring and timely intervention.

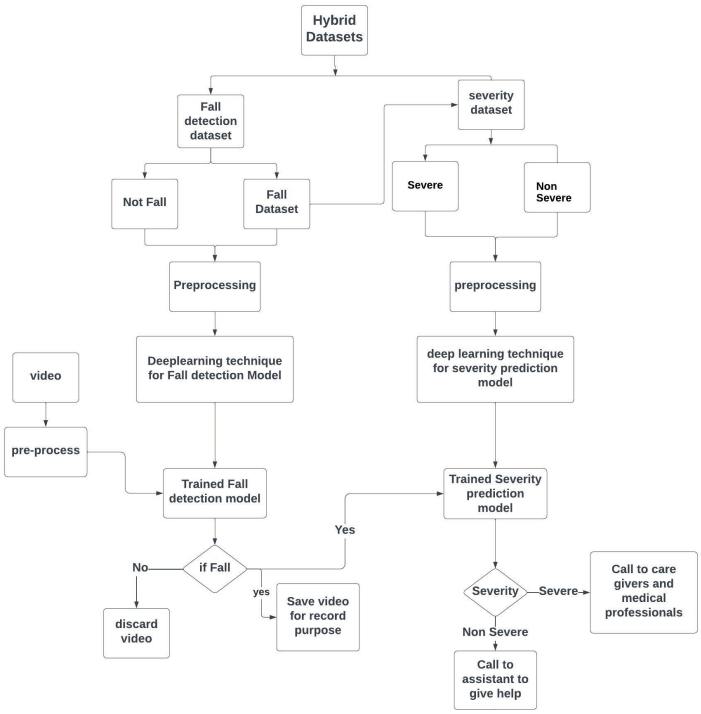

Figure 3.4: Basic Flow chart

3.5 Long-Term Recurrent Convolution Network For Fall Detection

Long-term recurrent convolutional network (LRCN) combines the convolutional neural network (CNN) and recurrent neural network (RNN) approaches. The model takes both spatial and temporal information in its raw form and passes it through a CNN to convert it into feature vectors before feeding it to an RNN for sequence analysis and prediction of the outcomes[39].

In the proposed methodology, a long-term recurrent convolution network model is used for the detection of falls. The methodology encompasses data preparation, data preprocessing, model architecture, model training, and evaluation.

3.5.1 Data Pre-processing

- Data annotation: The Data preprocessing starts with the manual annotation of the dataset. In dataset annotation, we categorized videos into the main dataset as the Fall dataset and Severity dataset. Furthermore, every category is sub-categorized as fall/not fall and severe/not severe. Every single video is seen and grouped accordingly to the category in which it lies.
- Data Augmentation: The data augmentation techniques applied during the frame extraction process improve both the robustness and generalization of the proposed vision-based fall detection and severity prediction. Several augmentation techniques are employed as follows:
  - Frame Skipping (Temporal Sampling): The frame extraction method chooses intervals of frames through temporal sampling. A skip-frames-window calculation determines which subset of frames to use rather than processing the complete video sequence during the extraction process. Temporal downsampling occurs through this method, which enables the model to concentrate on essential moments while eliminating redundant computations and data.
  - Resizing (Spatial Transformation) The framework scales all extracted frames until they reach IMAGE-HEIGHT and IMAGE-WIDTH specifications. Standardization procedures create consistent dimensions among multiple movies, thus resulting in fewer shape mismatches that enable efficient processing by the neural network.
  - Normalization (Pixel Value Scaling) To stabilize training and accelerate convergence, pixel values are normalized by dividing them by 255. This adjustment shifts the pixel intensity range from (0, 255) to (0, 1), decreasing the impact of changing lighting conditions and enhancing numerical stability during model training.

A video data preparation process is used in the suggested technique to build a deep learning model for fall detection. The dataset is divided into "Fall" and "Not Fall" classes. Each video's

frames are taken out, scaled to 64 × 64 pixels, and then normalized. The directory location is supplied, and the number of frames per video is set to 25 to guarantee constant training data. The reading of the video file, frame scaling, and normalization of the pixel values between 0 and 1 are all done by the frames_extraction function in the process of feature extraction. The preprocessed

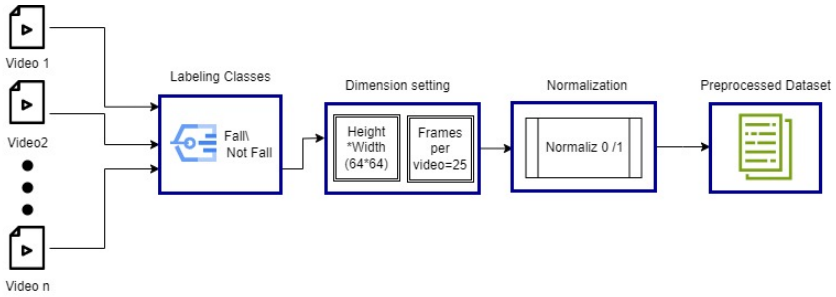

Figure 3.5: Pre-Processing

frames are combined into a feature array by the construct_dataset function, which also filters out films that don't meet the sequence length criteria and generates label arrays indexed to the relevant class indices.

3.5.2 Model Architecture

By combining the convolutional neural network and long short-term memory, the LRCN model captures spatial and temporal features. The model architecture includes the following.

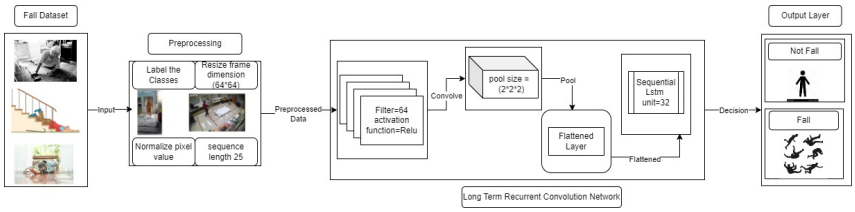

Figure 3.6: Proposed LRCN model's architecture for human fall detection

Time Distributed Convolution layers

- Conv 2D Layer

  In the initial phase of the model, a time-distributed layer wraps two convolutional layers. This makes it possible to apply the convolutional processes to every frame in the video stream separately.

- First Conv2D Layer

  This layer consists of 16 filters. With the kernel size (3*3). By using the ReLU activation function, non-linearity is introduced, which helps the model learn complex patterns. To preserve the spatial dimension and to make sure that the output size is the same as the input size, we set the padding as "same.

- Second Conv2D Layer

  This layer contains the same kernel size and ReLU as an activation function with 32 filters. With more filters, the model can extract more fine-grained spatial information from every frame.

- Third Conv2D Layer

  This layer contains the same kernel size and ReLU as an activation function with 64 filters. With more filters, the model can extract more fine-grained spatial information from every frame.

Max Pooling Layer

In this layer, every frame in the sequence passes through the max pooling layer separately. By picking the maximum value within a pool size window (2, 2), max pooling lowers the spatial dimensions (height and width) and helps down-sample feature maps while maintaining significant features and lowering computational complexity.

Flatten Layer

After going through the convolutional and pooling layers, each frame of the feature map is transformed into a single vector. The flattening procedure, which turns the 2D feature maps into 1D feature vectors, handles each frame independently because of the TimeDistributed wrapper.

LSTM Layer

LSTM layer Temporal dependencies are captured across the sequence of frames using a long-short-term layer. There are 32 units within the LSTM layer to learn the temporal dynamics and sequential patterns represented in the videos. When it comes to managing long-term dependencies and reducing the problem of vanishing gradients, LSTM networks are quite useful.

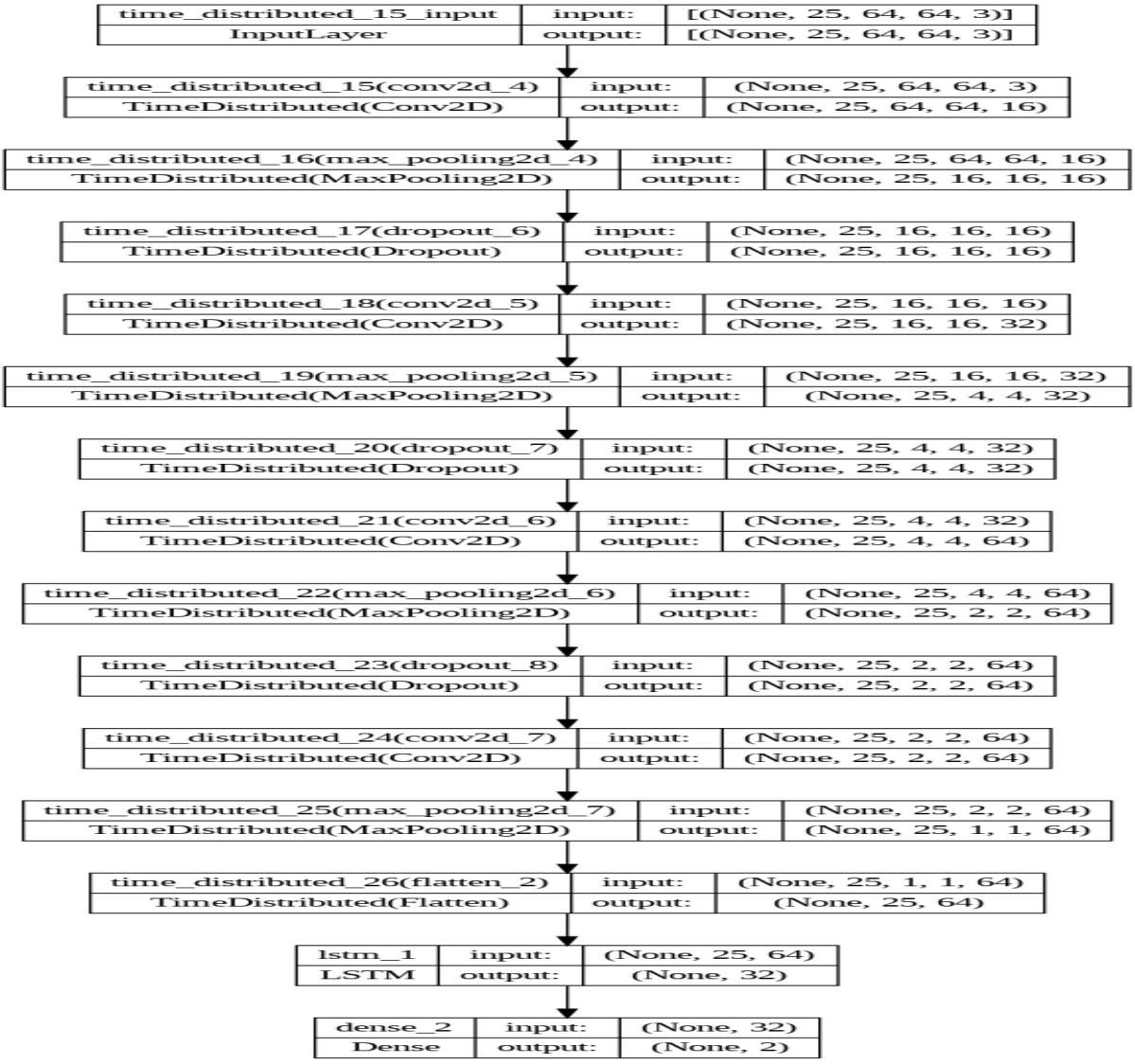

Figure 3.7: The architecture of the proposed LRCN models for human fall detection

Dense Layer

The dense output layer Softmax activation function is used in this final layer, which is densely connected. The number of units in this layer is equal to the number of classes in the dataset. This softmax activation function then gives probabilities indicating how likely each class is for a given input sequence.

Activation Functions

Activation functions are functions that are used to introduce nonlinearity into neural networks. They are used in the calculation of the weighted sum of inputs and biases to arrive at the output of that neuron..

ReLu

A common activation function in hidden layers of neural networks is the ReLU function, which is easy but efficient. If the input is affirmative, it gives the input; if not, it gives 0. Its value ranges from 0 to infinity [0 - inf]

$$ReLU(x) = max(0,x) \tag{3.1}$$

Figure 3.8: ReLu Activation Function

Softmax

In the output layer of neural networks, the softmax function is frequently utilized for classification tasks. It ensures that the average of each element in a vector of integers equals one by converting it into a probability distribution.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}. \tag{3.2}$$

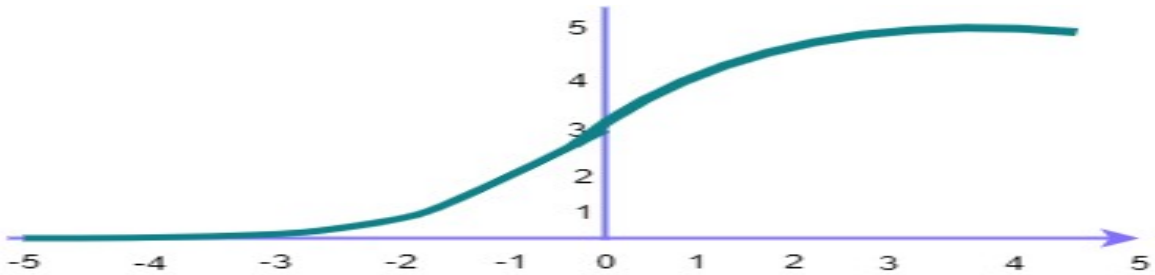

Figure 3.9: Softmax Activation Function

3.5.3 Training and Evaluation

The dataset is divided into training and testing subsets to facilitate effective model evaluation. To mitigate the risk of overfitting, the model employs early stopping, which tracks the validation loss and halts training if there is no improvement over a predetermined number of epochs, known as the patience level. Fitting the model to the training data and assessing its performance on a separate validation set comprise the training phase. In the current research, the ratio in which data are divided for train test split is set at 80:20, where 80% of data is reserved for training the model and the rest 20% for testing.

3.5.4 Prediction and Visualization

The proposed model predicts whether the action is falling or not in the test videos. For predicting the action, we used the prediction action function that extracts the frames from videos and normalizes them and, based on that, predicts action.

3.6 LRCN Algorithm For Fall Detection

This algorithm is for processing the video, extracting the fixed-length frames, and resizing them to standard size. The pseudocode starts by defining the constants to set its dimensions image_height as 64, image_width as 64, and to determine the number of frames taken from the video we used sequence length as 25.
The frames_extraction function accepts a video_path as an argument and processes the video file found at that path. The method creates an empty list named frames to hold processed frames. It then accesses the video file and determines the interval required to skip frames, extracting just SEQUENCE_LENGTH frames. For each frame, the function positions the video, reads the frame, resizes it to the requested height and width, normalizes its pixel values to [0, 1], and adds the processed frame to the frames_list. After processing the necessary amount of frames, the function returns a list of processed frames. This approach extracts a consistent, standardized collection of frames from any movie, which is critical for activities such as video analysis or

Algorithm 1 Frame Extraction from Video

```
DEFINE IMAGE_HEIGHT as 64
DEFINE IMAGE_WIDTH as 64
DEFINE SEQUENCE_LENGTH as 25
function FRAMES_EXTRACTION(video_path)
    INITIALIZE empty list frames
    OPEN video file from video_path
    GET total number of frames in the video
    CALCULATE interval to skip frames to achieve SEQUENCE_LENGTH
    for each frame from 0 to SEQUENCE_LENGTH do
        SET frame position in the video
        READ the frame
        RESIZE the frame to (IMAGE_HEIGHT, IMAGE_WIDTH)
        NORMALIZE the frame to range [0, 1]
        APPEND the processed frame to frames
    end for
    RETURN frames
end function
```

machine learning.

The algorithm creates a method called create_dataset() that processes video data for a machine

Algorithm 2 Dataset Creation and Label Encoding and Dataset Splitting

```
function CREATE_DATASET
    INITIALIZE empty lists features, labels, video_files
    for each class_name in class_names do
        PRINT "Processing" class_name
        for each video_file in train_dir/class_name do
            CALL frames_extraction(video_file) to extract frames
            if extracted frames count matches SEQUENCE_LENGTH then
                APPEND frames to features
                APPEND class_names.index(class_name) to labels
                APPEND video_file to video_files
            end if
        end for
    end for
    CONVERT features and labels to numpy arrays
    RETURN features, labels, video_files
end function
CALL create_dataset()
ONE-HOT encode labels for classification
SPLIT features and labels into training and testing sets with 80-20 ratio
```

learning project. It creates empty lists to store features, labels, and video file names. It processes each video file in the directory matching to each class name in a given list of classes. The frames_extraction(video_file) function is used to extract frames from each video, and the number of retrieved frames is compared to a defined sequence length. If so, it adds the frames, the class name index, and the video file location to the appropriate lists. After analyzing all videos, the lists are converted to numpy arrays and returned. The method then applies one-hot encoding to the labels for classification purposes and divides the dataset into training and testing sets in an 80-20 ratio.

At the start, we write a method called create _LRCN_model() that will initialize our LRCN (Long-term Recurrent Convolutional Network). In this function, we start by building a sequential

Algorithm 3 LRCN Model Creation and Training

```
procedure CREATE_LRCN_MODEL
    INITIALIZE Sequential model model
    ADD TimeDistributed Conv2D layers with ReLU activation to model
    ADD TimeDistributed MaxPooling2D layers to model
    ADD Dropout layers to model
    ADD LSTM layer to model
    ADD Dense output layer with softmax activation to model
    DISPLAY model summary
    RETURN model
end procedure
CALL create_LRCN_model()
DEFINE metrics for precision, recall, binary accuracy
COMPILE the model with Adam optimizer and categorical cross-entropy loss
DEFINE EarlyStopping callback
TRAIN the model with training data, using validation split and early stopping
```

model and then adding layers to it. These layers consist of TimeDistributed Conv2D layers with ReLU activation, TimeDistributed MaxPooling2D layers, and Dropout layers. Next, we add an LSTM layer, followed by a Dense output layer with softmax activation.
We use the create_LRCN_model() function to build the LRCN model and define precision, recall, and binary accuracy metrics. The model is based on the Adam optimizer with categorical cross-entropy loss. To prevent overfitting, an EarlyStopping callback is set up to monitor the training process. Finally, the model is trained using the training data with a validation split and early stopping to ensure successful learning.

The predict_action function uses an LRCN model to detect actions in a video. It starts by loading the movie from the supplied file directory and determining its size and total number of frames. The function then determines how frequently to sample frames based on the length of the sequence required for the model. It collects a set number of frames from the video, resizing and normalizing each one to ensure they meet the model's input criteria. These processed frames are compiled into a list for further investigation.

Once the frames are created, the function sends them into the LRCN model, which predicts

Algorithm 4 Predict Action from Video

function PREDICT_ACTION(video_file_path, SEQUENCE_LENGTH)
    Initialize video reader with `cv2.VideoCapture(video_file_path)`
    Get original video width and height
    Declare empty list *frames_list*
    Initialize *predicted_class_name* as an empty string
    Get total number of frames in the video
    Calculate skip_frames_window as `max(int(video_frames_count / SEQUENCE_LENGTH), 1)`
    for frame_counter = 0 to SEQUENCE_LENGTH - 1 **do**
        Set frame position to `frame_counter * skip_frames_window`
        Read frame from video
        if frame is not read properly **then**
            Break
        end if
        Resize frame to dimensions `(IMAGE_HEIGHT, IMAGE_WIDTH)`
        Normalize frame by dividing pixel values by 255
        Append normalized frame to *frames_list*
    end for
    Predict labels probabilities using `LRCN_MODEL.predict(np.expand_dims(frames_list, axis = 0))[0]`
    Find index of maximum probability
    Get class name from `CLASSES_LIST` using the index
    Print `predicted_class_name` and prediction confidence
    Release video reader
end function

the action in the movie. The model generates probabilities for several action classes, and the function selects the class with the best likelihood of determining the anticipated action. It then writes out the expected action, as well as the prediction's confidence level. Finally, the function frees the video file resource. The function's usage example shows how to apply it to a test video and display the results.

3.7 Long-Term Recurrent Convolution Network For Severity Prediction

The machine learning algorithm Lrcn predicts the severity upon receiving a video from a severity data set and the fall detection model if a fall occurs.

3.7.1 Data Pre-Processing

In this phase, we have created a function that can extract frames from the videos to create a dataset to train a model. The method retrieves frames from each movie in a directory that contains videos of various classifications. The frames are then added to the dataset after being resized and normalized. The paths of the video files, the frames, and the relevant class labels are stored to form the dataset.

- Frame Extraction: The frame extraction function extracts the frame from the video. The function takes the video frames as input and returns the list of resized and normalized frames.

- Resize and Normalizing: The extracted frames are resized into a specified height and width (64*64). After resizing the frame, each pixel value is divided by 255 to normalize it and make sure that the pixel value lies between 0 and 1.

- Dataset creation: We used the create dataset function for creating a dataset. It loops over the classes defined in the class list variable, extracting frames from each video file in the relevant class directory. If the number of extracted frames matches the sequence length, the frames, class index, and video file path are added to the dataset. The proposed model predicts whether the action is falling or not in the test videos. For predicting the action, we

```
def frames_extraction(video_path):

    # Declare a list to store video frames.
    frames_list = []

    # Read the Video File using the VideoCapture object.
    video_reader = cv2.VideoCapture(video_path)

    # Get the total number of frames in the video.
    video_frames_count = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))

    # Calculate the the interval after which frames will be added to the list.
    skip_frames_window = max(int(video_frames_count/SEQUENCE_LENGTH), 1)

    # Iterate through the Video Frames.
    for frame_counter in range(SEQUENCE_LENGTH):

        # Set the current frame position of the video.
        video_reader.set(cv2.CAP_PROP_POS_FRAMES, frame_counter * skip_frames_window)

        # Reading the frame from the video.
        success, frame = video_reader.read()

        # Check if Video frame is not successfully read then break the loop
        if not success:
            break

        # Resize the Frame to fixed height and width.
        resized_frame = cv2.resize(frame, (IMAGE_HEIGHT, IMAGE_WIDTH))

        # Normalize the resized frame by dividing it with 255 so that each pixel value then lies between 0 and 1
        normalized_frame = resized_frame / 255

        # Append the normalized frame into the frames list
        frames_list.append(normalized_frame)

    # Release the VideoCapture object.
    video_reader.release()

    # Return the frames list.
    return frames_list
```

Figure 3.10: Preprocessing and Frame Extraction

used a prediction action function that extracts the frames from videos, normalizes them, and, based on that, predicts action.

3.7.2 Model Architecture

The Long-term recurrent convolutional network model is proposed by combining CNN and recurrent neural networks, which is good for series data, such as video data and temporal data, and this model uses the keras api called sequential. This refers to the make-up of the services as comprised of the undermentioned parts.

Convolutional layer

The described model starts with the four convolutional layers followed by max pooling layers and the dropout layers. These are then wrapped up in the time-distributed layer, which helps the model to work on sequential input. This layer uses the following parameters:

- Filters: It includes the number of filters for each level of layers, namely 16, 32, 64, and 64.

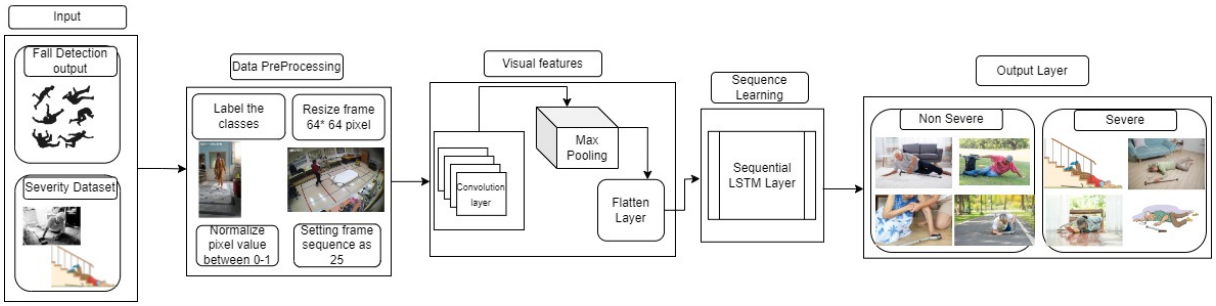

Figure 3.11: The Model diagram for the proposed LRCN model for Severity prediction

- Kerenal size: The size of the convolutional kernal is (3 * 3).
- Padding: The padding is the same, which means that the output feature map will have the same dimension as the original input.
- Activation function: we have used Relu as activation function because it include the nonlinearity into the model, it minimizes the problem of vanishing gradient during training of the model and it also helps the model to learn the very complex patterns.

Recurrent layer(Lstm)

Analyzing the architecture of this model, the author applied the single long short-term memory layer that contains 32 units. This layer treats the input it receives from the convolutional layer and comes up with the prediction or classifies it. LSTM consists of the following components:

- memory cells: It is the internal state of the LSTM layer that stores data from earlier time steps during a specific time step.
- Input gate: The input gate is used for transferring the flow of newly acquired information into the memory cell.
- Output gate: Decide on the values of the LSTM layer based on the memory cells and the current input.
- Forgot Gate: This gate is also meant to choose the information that is to be removed from a memory cell.

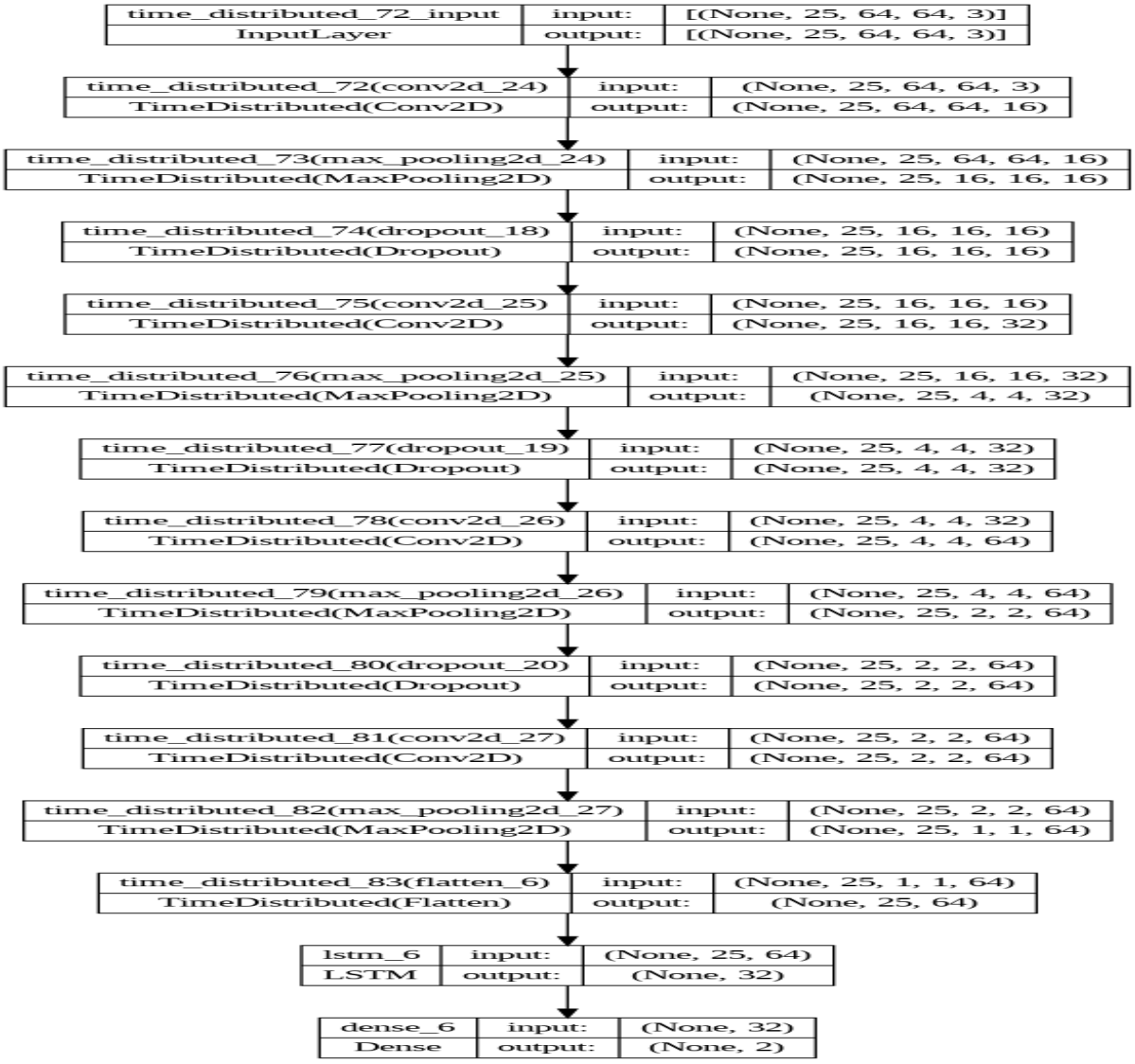

Figure 3.12: The architecture of the proposed LRCN model for Severity prediction

Dense layer

This is the final layer and is a fully connected layer with the nodes/units in the previous layers. This kind of layer accepts outputs of the previous layers of the network and generates the probability distribution of the classes. This layer applies the softmax activation function to get the final results.

3.7.3 Model Traning and Evaluation

The dataset is further divided into training and testing that provide the framework for the model's formation, which is done with the help of early stopping. The model is assessed on the

test set. We have divided training and test split to 80:20; however, 80% of data is reserved for training the model and the rest 20% for testing.

- Precision, recall, and binary accuracy metrics: These are important when it comes to analyzing the performance of classification models.
- Early Stopping Callback: EarlyStopping is a callback method that is used to monitor a given stat during training and pause the training session in an attempt to get a better stat.
- Model Compilation: This calls for the LRCN model and then defines the loss function, optimizer, and evaluation metric of precision, recall, and binary accuracy to train the model.
- Model Preparation: The training is done with the offer assistance of the fit strategy that utilizes the features train and labels train as the preparing set, number of epochs, batch size, validation split, and early stopping callback

3.7.4 Prediction and Visualization

We have defined a function named predict action that does single action recognition prediction on a video employing the LRCN model that is composed of CNN and RNN; the LRCN model is used to recognize actions in videos. Thus, using pre-processing of the frames, passing them to the model, and analyzing the predicted probabilities, we achieve an accurate recognition of the action in the videos.

3.7.5 Real-Time Notification and IoT-Based Emergency Alerts

If the system detects severe fall incidents, the system generates automatic real-time alerts, which notify caregivers, along with family members and emergency responders, via SMS and email and through push notifications for timely emergency support.
Through IoT-based emergency alert integration, the system becomes capable of communicating with smart home assistants to trigger alarms and open doors or contact nearby medical facilities. In case the system detects a non-severe incident, it only generates an alert for assistance to help them.

3.8 LRCN Algorithm For severity prediction

Algorithm 5 Frame Extraction from Severity Dataset

```
DEFINE IMAGE_HEIGHT as 64
DEFINE IMAGE_WIDTH as 64
DEFINE SEQUENCE_LENGTH as 25
function FRAMES_EXTRACTION(video_path)
    INITIALIZE empty list frames
    OPEN video file from video_path
    GET total number of frames in the video
    CALCULATE interval to skip frames to achieve SEQUENCE_LENGTH
    for each frame from 0 to SEQUENCE_LENGTH do
        SET frame position in the video
        READ the frame
        RESIZE the frame to (IMAGE_HEIGHT, IMAGE_WIDTH)
        NORMALIZE the frame to range [0, 1]
        APPEND the processed frame to frames
    end for
    RETURN frames
end function
```

The algorithm describes the process of extracting frames from a video file. First, it defines the frame size as 64 pixels in height and 64 pixels in width, and it specifies that each sequence will have 25 frames. It then provides a function called frames_extraction, which accepts the path to a video file as input. Within this method, an empty list is constructed to store the frames. The video is opened, and the total number of frames is calculated. To guarantee that only the essential frames are recorded, the algorithm determines how many frames to skip between each one to achieve the desired sequence length. It then loops over the required number of frames, adjusting the location in the video, reading the frame, resizing it to the specified dimensions, normalizing the pixel values to be between 0 and 1, and adding the processed frame to the list. Finally, the method produces a list of processed frames.

The create_dataset() method organizes and prepares video data for study. It starts by creating

Algorithm 6 Dataset Creation and Preparation

```
function CREATE_DATASET
    INITIALIZE empty lists features, labels, video_files
    for each class_name in class_names do
        PRINT "Processing" class_name
        for each video_file in train_dir/class_name do
            CALL frames_extraction(video_file) to extract frames
            if extracted frames count matches SEQUENCE_LENGTH then
                APPEND frames to features
                APPEND class_names.index(class_name) to labels
                APPEND video_file to video_files
            end if
        end for
    end for
    CONVERT features and labels to numpy arrays
    RETURN features, labels, video_files
end function
CALL create_dataset()
ONE-HOT encode labels for classification
SPLIT features and labels into training and testing sets with 80-20 ratio
```

empty lists to store features (extracted frames), labels (activity categories), and video file names. It processes all films in each action class, collecting frames from each one using a separate method named frames_extraction(). If the number of extracted frames equals the preset length, it adds the frames to the features list, assigns the matching class index to the labels list, and saves the video file name. After analyzing all the films, the method changes the features and labels to a machine-learning-friendly format (numpy arrays) before returning them. Finally, the labels are converted to a one-hot encoded format for classification, and the data set is divided into training and testing sets at an 80-20 ratio to guarantee successful model training and evaluation.

Algorithm 7 Create LRCN Model

```
function CREATE_LRCN_MODEL
    INITIALIZE Sequential model model
    ADD TimeDistributed Conv2D layers with ReLU activation to model
    ADD TimeDistributed MaxPooling2D layers to model
    ADD Dropout layers to model
    ADD LSTM layer to model
    ADD Dense output layer with softmax activation to model
    DISPLAY model summary
    RETURN model
end function
CALL create_LRCN_model()
```

This Algorithm explains how to build a Long-term Recurrent Convolutional Network (LRCN) model for action recognition in videos. It creates a sequential model and adds several layers, including TimeDistributed Conv2D layers for frame processing, TimeDistributed MaxPooling2D layers for dimensionality reduction, Dropout layers to prevent overfitting, an LSTM layer to capture temporal dependencies, and a Dense output layer with softmax activation for class probabilities. The method ends with showing the model summary and returning the generated model for predictions.

The algorithm specifies the procedures for properly training a model by specifying evaluation measures such as precision, recall, and binary accuracy. The model is compiled using the Adam optimizer with categorical cross-entropy loss, with an EarlyStopping callback to prevent overfitting. During training, a subset of the data is retained for validation, and the early stopping mechanism aids in determining the appropriate stopping moment. Finally, a function is developed

Algorithm 8 Model Training and Evaluation

```
DEFINE metrics for precision, recall, binary accuracy
COMPILE the model with Adam optimizer and categorical cross-entropy loss
DEFINE EarlyStopping callback
TRAIN the model with training data, using validation split and early stopping
function PLOT_METRIC(metric_name_1, metric_name_2, plot_name, color)
    PLOT training and validation metrics over epochs
end function
CALL plot_metric for loss, precision, recall, accuracy
```

to visualize the training and validation metrics over epochs, providing information about the model's performance and learning process.

This algorithm describes a machine learning model-based technique for predicting actions in videos. It opens the video file, determines its size, and prepares to read a set number of frames. It validates if each frame is legitimate, then resizes and normalizes it before adding it to a list. The analyzed frames are then sent into the model to generate predictions, which identify the activity with the highest likelihood and display it along with the confidence level before releasing the video clip.

3.9 Other Trained Models

3.9.1 3D Convolutional neural network for fall Detection and severity prediction

CNNs are a certain type of neural network that is most suitable for data that are in a grid format, such as image data. The convolutional layer is the simplest form of the layer in a CNN and sometimes goes by the name of the cell. It involves element-wise multiplication of the input with the set of filters (kernels) to produce feature maps that capture local spatial relations.[40] The model implementation begins by labeling the dataset into two classes: "fall" and "not fall." After that, the dimension of the height and width is adjusted to 64×64 pixels. We have used

Algorithm 9 Predict Action from Video

```
function PREDICT_ACTION(video_file_path, SEQUENCE_LENGTH)
    INITIALIZE the VideoCapture object to read from the video file
    GET the width and height of the video
    INITIALIZE a list to store video frames
    INITIALIZE a variable to store the predicted action
    GET the number of frames in the video
    CALCULATE the interval after which frames will be added to the list
    for each frame from 0 to SEQUENCE_LENGTH do
        SET the current frame position in the video
        READ the frame
        if frame not read properly then
            BREAK loop
        end if
        RESIZE the frame to fixed dimensions
        NORMALIZE the frame by dividing by 255
        APPEND the pre-processed frame to frames_list
    end for
    PASS the pre-processed frames to the model and get predicted probabilities
    GET the index of the class with the highest probability
    GET the class name using the retrieved index
    DISPLAY the predicted action along with the prediction confidence
    RELEASE the VideoCapture object
end function
CALL predict_action on the test video
```

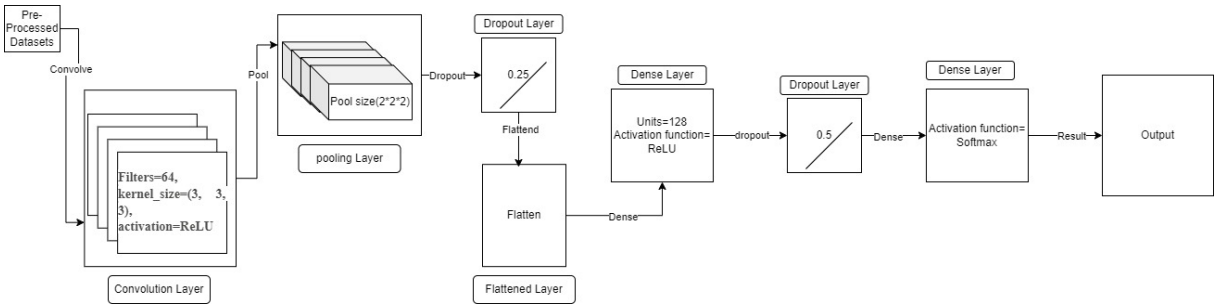

Figure 3.13: 3D-CNN model diagram for human fall detection and severity prediction

different frame sizes. The next preprocessing step is the normalization of the dataset after the resizing.

The authors of this work develop a 3D CNN model for the classification of video sequences. For the two parts of the video sequences, CNN extracts spatial and temporal features from video sequences, and LSTM captures sequential features in the data.

The CNN has the following layers: the first layer is the 3D convolutional layer, the second layer is the 3D max pooling layer, and the two dropout layers. The 3D convolution layer is defined with 64 filters, a kernel size of depth of 3, width of 3, and it takes in 3D inputs with a ReLU activation function. The max pooling layer in this architecture is defined as a 3D max pooling layer, which means that it lowers the size of the feature maps.

The CNN output is feed-forwarded through the dense layer that has 128 neurons with ReLU as the activation function. To decrease the problem of overfitting, another dropout layer is included as part of the convolutional neural network architecture. Lastly, the output is passed to a dense layer that contains the number of neurons as the number of classes in the data sets with a softmax function so that the result gives the probability of each class.

The model is trained with an Adam optimizer, and the loss function being used is the sparse categorical cross-entropy. The measure of effectiveness of the model is primarily in the form of accuracy.

3.9.2 3D CNN-LSTM for fall Detection and severity prediction

Preprocessing is the process in which, given a movie, frames are extracted from it, and the pixel values are normalized. To perform tasks such as reading the video files, picking out frames to resize them to 64x64 pixels, and normalizing pixel intensities to be between 0 and 1,1,1, there

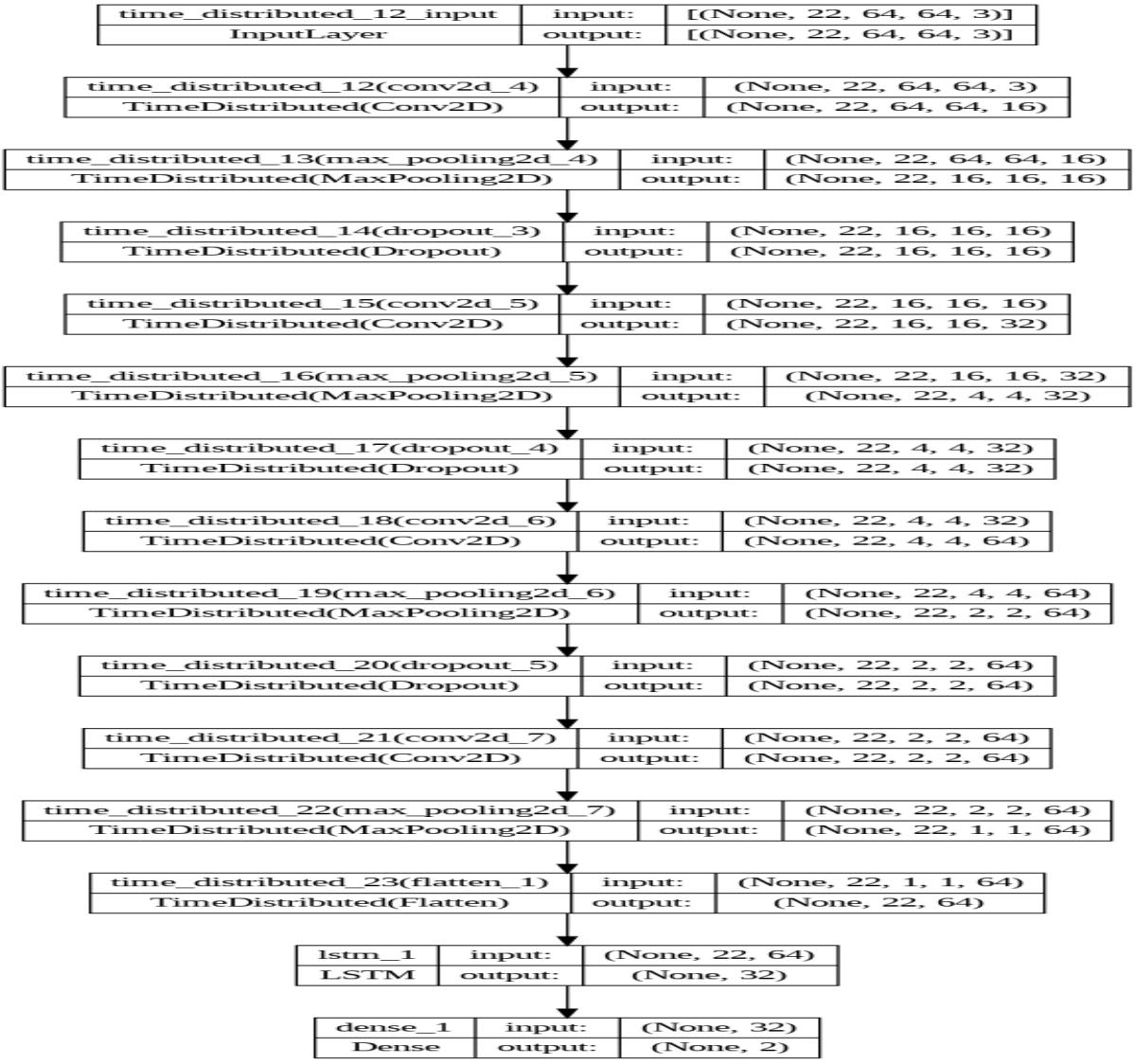

Figure 3.14: The architecture of the 3d CNN model for fall detection and Severity prediction

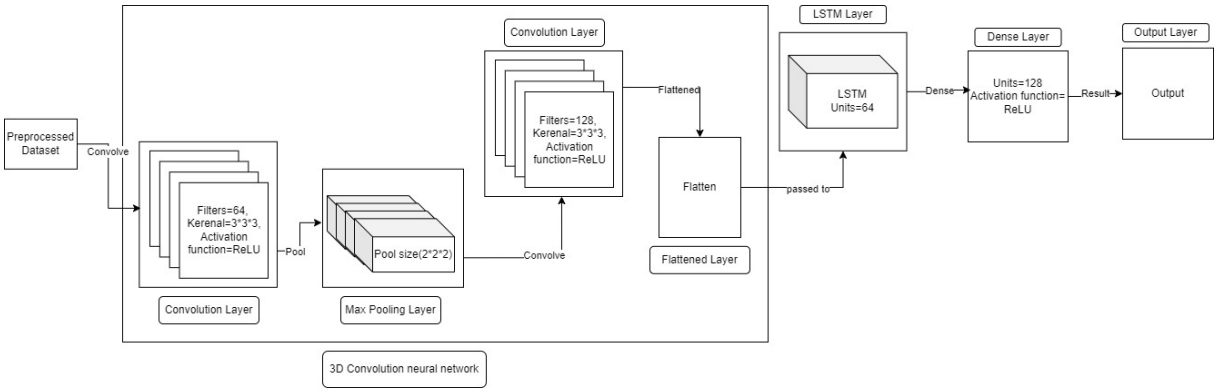

Figure 3.15: 3D-CNN model diagram for human fall detection and severity prediction

is a function known as frame extraction. We have a sequence of 25 frames for every video, which are then sampled periodically. The preprocessed frames are compiled into an array of features,

and any movie that does not conform to the inherent sequence length is eliminated through the construct dataset function. The label arrays are then obtained by indexed feature arrays by the corresponding class indices.

We utilized the 3D CNN-LSTM model in which the first two layers have a kernel size of 3x3x3, known as Conv3D, having 64 and 128 filters, respectively. These are followed by the 3D max pooling layer, which downsamples the spatial dimension of the feature maps. This layer has a pool size of 2 x 2 x 2. The TimeDistributed wrapper is then utilized to flatten the 3D CNN output since the output of the Dimension Difference layer is a 3D tensor where each 'frame' represents a time step in the input sequence where Flatten has been applied to each time step of the input sequence. In the next layer, a 64-unit long short-term memory layer (LSTM) captures the input sequence's long-term dependencies. Next, after the LSTM, two more dense layers with 128 inputs directly connected to the len of CLASSES_LIST are included. ReLU is the activation function applied in the first dense layer, while the softmax function is applied in the second dense layer since it is used for multiclass classification problems. The model is trained with the Adam optimization technique, a sparse categorical cross-entropy loss function, with accuracy as the evaluation measure.

3.10 Fully Connected Neural Network

As part of the research, a dataset had to be prepared for analysis, visualized, preprocessed, and created; a fully connected neural network model had to be built and trained; and action recognition on a video clip had to be done. To prepare the dataset, files from the original dataset were copied into folders for each class. Matplotlib was used to show the data after preprocessing it, which included removing frames from video files and standardizing the pixel values. By repeatedly going over each class and extracting frames from every video clip, the dataset was produced.

We have used a fully connected neural network in which a neuron of one layer is connected to every neuron of the subsequent layer. To specify the network structure, the first step in the code is getting the required classes from the Keras library, which includes sequential, flattened, and dense. The generation of the particular model generation procedure is then placed in the

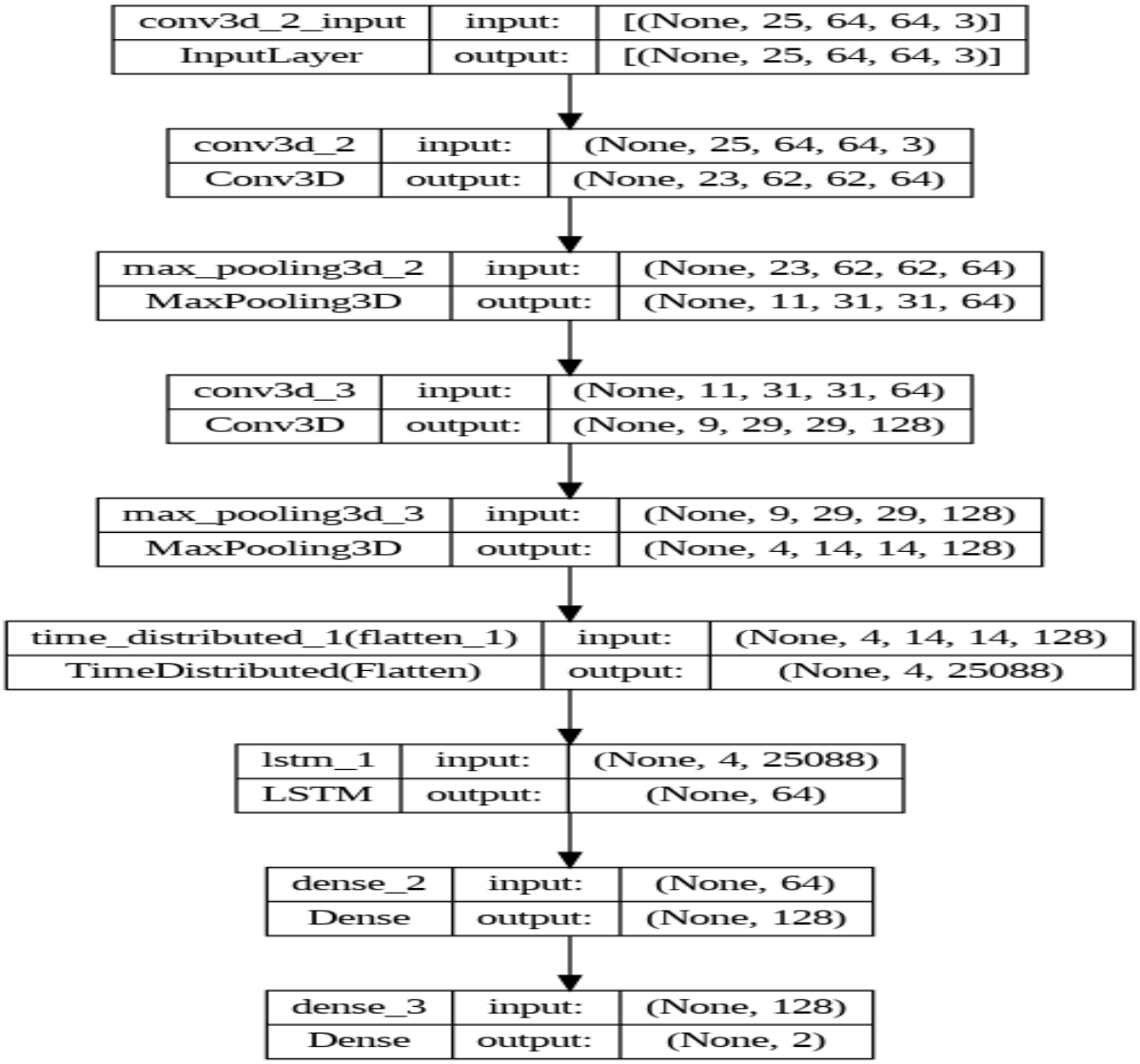

Figure 3.16: The architecture of the 3d CNN-LSTM model for fall detection and Severity prediction

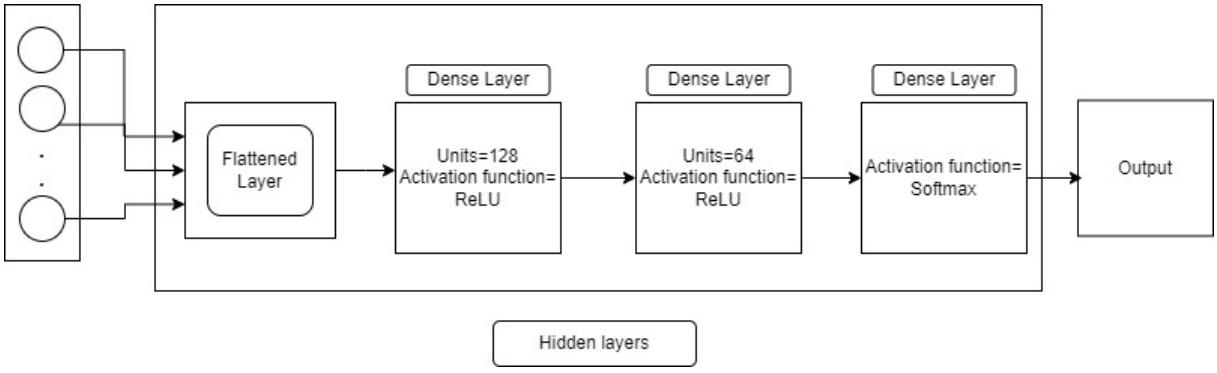

Figure 3.17: Fully connected neural network model diagram for human fall detection and severity prediction

Create Fully Connected Model function so that the code is well modulated and reusable. First, a sequential model is formed, and it gives the possibility of layering additions. The input shape

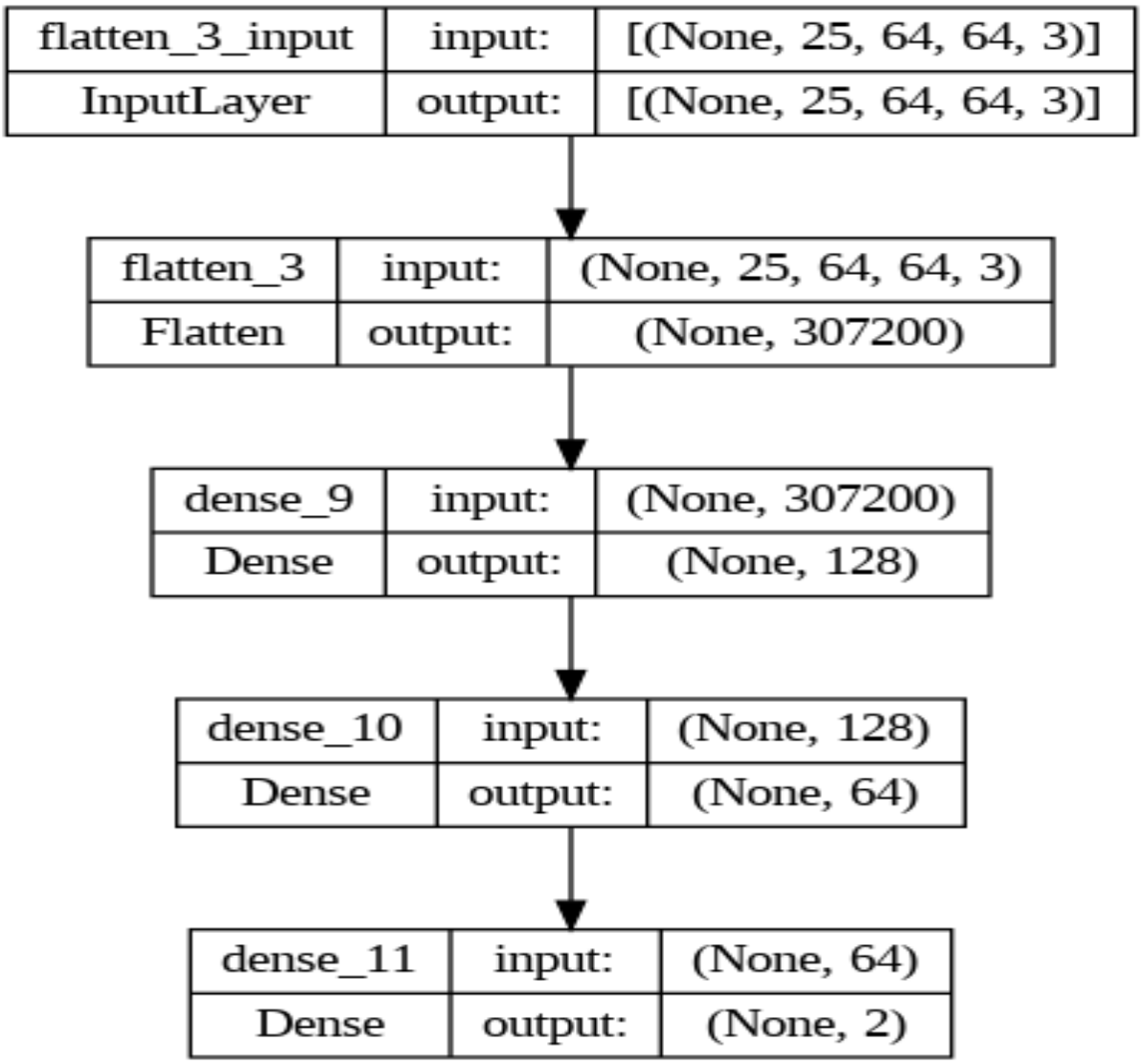

Figure 3.18: The architecture of the fully connected neural network model for fall detection and Severity prediction

parameter is used to specify the dimensions of the input, and the flattening layer is used to flatten the input, which is multidimensional. After that, two more layers are recorded, which are the complete connected layers; one of them contains 128 units and added non-linearity using the ReLU activation function, while the second layer contains 64 units and uses the ReLU activation function. The last dense layer takes advantage of the Softmax activation function to generate the probabilities concerning the classes and the quantity of classes in the classification task.

3.11 Summary

This chapter provides details of how techniques and steps were followed in the development of the fall detection and severity prediction model. Firstly, the details of the dataset and system requirements are described. Further, all the algorithms explained are used in both fall detection and severity prediction. Additionally, activation functions and algorithms are described for the best model.

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Overview

The proposed vision-based fall detection and severity prediction models and methodologies that were introduced in chapter 3 will be implemented, and the outcomes will be presented in this chapter. This stage looks at how well the system works in real-world scenarios, understands its benefits and drawbacks, and verifies its outputs against predefined benchmarks. This chapter is structured as follows: Section 4.2 provides specifics on the assessment parameters and outcomes, while Section 4.3 describes the analysis and discussion. Section 4.4 illustrates the noteworthy accomplishment and a comparison with other suggested gesture models. Section 4.5 provides a summary of this chapter.

4.2 Performance Metric

Performance measures in deep learning are used to assess a model's ability to generate predictions or classify data. These metrics allow developers to compare the accuracy of the model's outputs to the actual outcomes, determining if the model is useful or needs to be improved. The suggested method for detecting and predicting elderly falls is assessed using several metrics outlined below:

4.2.1 Precision

Precision is described as the ratio of accurately predicted positive cases to total expected positive instances. It evaluates the accuracy of optimistic predictions. Precision varies between 0 and 1, with higher numbers signifying better performance. A perfect accuracy of 1.0 indicates that the model accurately recognizes all positive instances without any false positives.

$$P = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{4.1}$$

4.2.2 Recall

Recall, also known as sensitivity or true positive rate, is the proportion of accurately anticipated positive cases to the total number of positive instances. It assesses how well the model detects positive cases. Recall is measured from 0 to 1, with higher numbers indicating greater performance. A perfect recall of 1.0 indicates that the model accurately recognizes all positive events.

$$R = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4.2}$$

4.2.3 F1-Score

The F1-Score represents the harmonic mean of accuracy and recall. It gives a balanced assessment of a model's performance, taking into account both accuracy and recall. The F1-Score ranges between 0 and 1, with higher values signifying greater performance. A perfect F1-score of 1.0 indicates that the model has both flawless accuracy and recall.

$$F_1\ score = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.3}$$

4.2.4 Accuracy

Accuracy is the percentage of accurately predicted occurrences inside a given class. It is determined individually for each class.

$$A = \frac{\text{Correctly Predicted Instances for a Class}}{\text{Total Instances for that Class}} \tag{4.4}$$

Class accuracy runs from 0 to 1, with higher numbers signifying superior achievement in that particular class.

4.2.5 Confidence

Confidence measures how convinced the model is about its predictions. It is often stated as a probability or a score ranging from 0 to 1. Higher confidence levels suggest that the model is more confident in its predictions.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}. \tag{4.5}$$

In deep learning, confidence is generally generated by applying a softmax function to the model's output. The softmax function turns the output logits into probabilities, with the highest probability indicating the predicted class.

4.3 Experimental Configurations

Experimental configurations in research relate to the exact settings, characteristics, and conditions under which an experiment is carried out. These combinations can have a major influence on the experimental results and conclusions. For Fall detection, we have set different parameters to achieve better results. On the below parameters, we have achieved the best results through further hyperparameter tuning. Their results will be discussed in the next section.

Table 4.1: Expermintal Configurations for Fall detection

Settings	**Value**	**Description**
Frame	23	The Frame size 23 is used in this research. This value is essential for recording temporal dynamics in video data. By giving the model a series of frames, it may learn patterns over time.

Batch Size	4	In experimentation, a batch size of 4 is employed. This implies that the model processes 4 samples of data at once during training or inference before changing parameters or generating predictions.
Epoch	100	The model was fine-tuned using 100 epochs, which involved passing the full dataset 100 times through it.
test size	0.1	This indicates that 10% of the whole dataset is held aside to evaluate the trained model's performance on previously unseen data.
validation split	0.2	This indicates that 20% of the entire dataset is utilized for validation throughout the training process.
Device	Google Colab	The proposed approach was trained on core i7 7th Generation, providing access to GPUs, TPUs, and other processing resources.

The table 4.2provides an overview of the experimental setups utilized in the research on fall severity prediction. Understanding the experimental design, as well as how the model was trained and assessed, requires understanding these combinations.

Table 4.2: Expermintal Configurations for Severity Prediction

Settings	**Value**	**Description**

Frame	25	The Frame size 25 is used in this research. This value is essential for recording temporal dynamics in video data. By giving the model a series of frames, it may learn patterns over time.
Batch Size	4	In experimentation, a batch size of 4 is employed. This implies that the model processes 4 samples of data at once during training or inference before changing parameters or generating predictions.
Epoch	100	The model was fine-tuned using 100 epochs, which involved passing the full dataset 100 times through it.
test size	0.10	This indicates that 10% of the whole dataset is held aside to evaluate the trained model's performance on previously unseen data.
validation split	0.10	This indicates that 10% of the entire dataset is utilized for validation throughout the training process.
Device	Google Colab	The proposed approach was trained on core i7 7th Generation, providing access to GPUs, TPUs, and other processing resources.

4.4 Results and Discussion

4.4.1 Fall Detection Results

We evaluated the technique we proposed for detecting human falls using Long-Term Recurrent Convolutional Networks to ensure its generalizability and reliability. As described in Chapter 3, each input video sequence was preprocessed by setting the frame size to 23,64*64 for the height and width dimensions and normalizing the dataset. This preprocessed dataset was used as the input for each evaluated model. Table 4.3 demonstrates that the proposed model was

Table 4.3: Hyperparameter Tuning for LRCN Human Fall Detection Model

Frame Size	**Test/ Validation**	**Batch/ Epoch**	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Confidence**
25	0.10/ 0.2	4/ 100	F=0.94, NF=0.94	F=0.94, NF=0.94	F=0.94, NF=0.94	F=0.94, NF=0.94	0.94
22	0.2 /0.3	16/ 100	F=0.89, NF=0.92	F=0.93, NF=0.88	F=0.91, NF=0.90	F=0.92, NF=0.87	0.99
22	0.2/ 0.3	4/ 100	F=0.87, NF=0.92	F=0.93, NF=0.86	F=0.90, NF=0.89	F=0.92, NF=0.85	0.94
23	0.1/ 0.2	16/ 100	F=0.87, NF=0.93	F=0.94, NF=0.84	F=0.90, NF=0.88	F=0.94, NF=0.83	0.99
23	0.1/ 0.2	32/ 100	F=0.94, NF=0.88	F=0.89, NF=0.84	F=0.89, NF=0.89	F=0.88, NF=0.93	0.99

trained using different hyperparameters, and those parameters were chosen that yielded the best results.

The performance matrices Precision, Recall, F1-score, Class Accuracy, and Confidence were used to assess the proposed approach. The model was implemented in the Google Colab environment, and many libraries were used, including TensorFlow, Keras, and OpenCV. The dataset was trained using various models, including 3D CNN, 3D CNN-LSTM, fully connected Neural Networks, and Long-Term Recurrent Convolutional Networks. The proposed LRCN model

outperformed the state-of-the-art models in terms of precision (92% for fall and 97% for not-fall), recall (97% for fall and 90% for not-fall), F1-score (94% for fall and 93% for not-fall), class accuracy (97% for fall and 90.3% for not-fall), and confidence (99.9%). This demonstrates the potential of the proposed LRCN model to enhance elderly fall detection.

Table 4.4: Human Fall Detection different trained models results having same parameters

Model	**Precision**	**Recall**	**F1-score**	**Accuracy**	**Confidence**
Fully connected Neural Network	F= 76.0%, Nf=81.0%	F=81.0%, NF=76%	F=79%, NF=79%	F=81.25%, NF=76.47%	98.7%
3D-CNN	F=81.0%, NF=93.0%	F=94.0%, NF=79.0%	F=87.0%, NF=86.0%	F=93.75%, NF=79.41%	99.4%
3D-CNN-LSTM	F=84.0%, NF=97.0%	F=97.0%, NF=82.0%	F=90.0%, NF=89.0%	F=96.87%, NF=82.35%	83.30%
LRCN	F=92.0%, NF=97.0%	F=97.0%, NF=90.0%	F=94.0%, NF=93.0%	F=97.1%, NF=90.3%	99.93%

Table 4.4 displays experiment results for 3D Convolutional Neural Networks (3D CNN), CNN-LSTM, Fully Connected Neural Networks, and Long-Term Recurrent Convolutional Networks (LRCN) models. The fig. 4.1 uses the accuracy to show the model's performance for Fall detection tasks, among which LRCN outperformed all others. Figure 4.2 shows the comparison diagram based on the accuracy of each model.

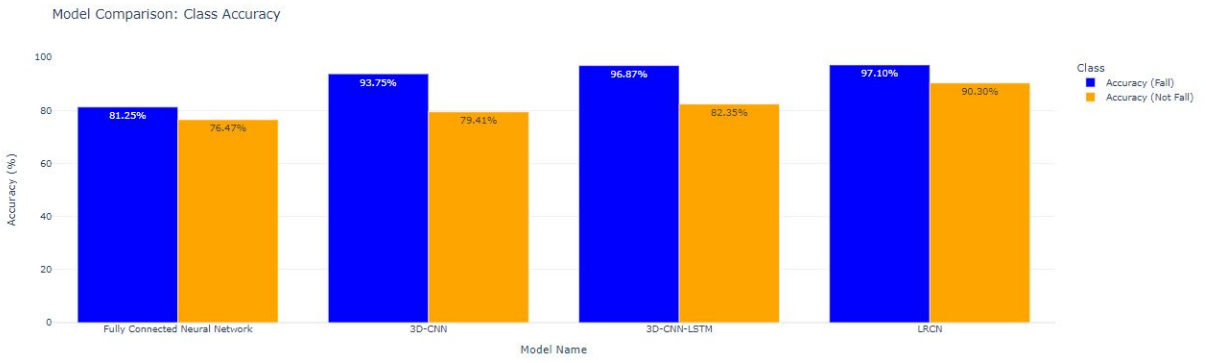

Figure 4.1: Human Fall Detection trained models Comparision Diagram

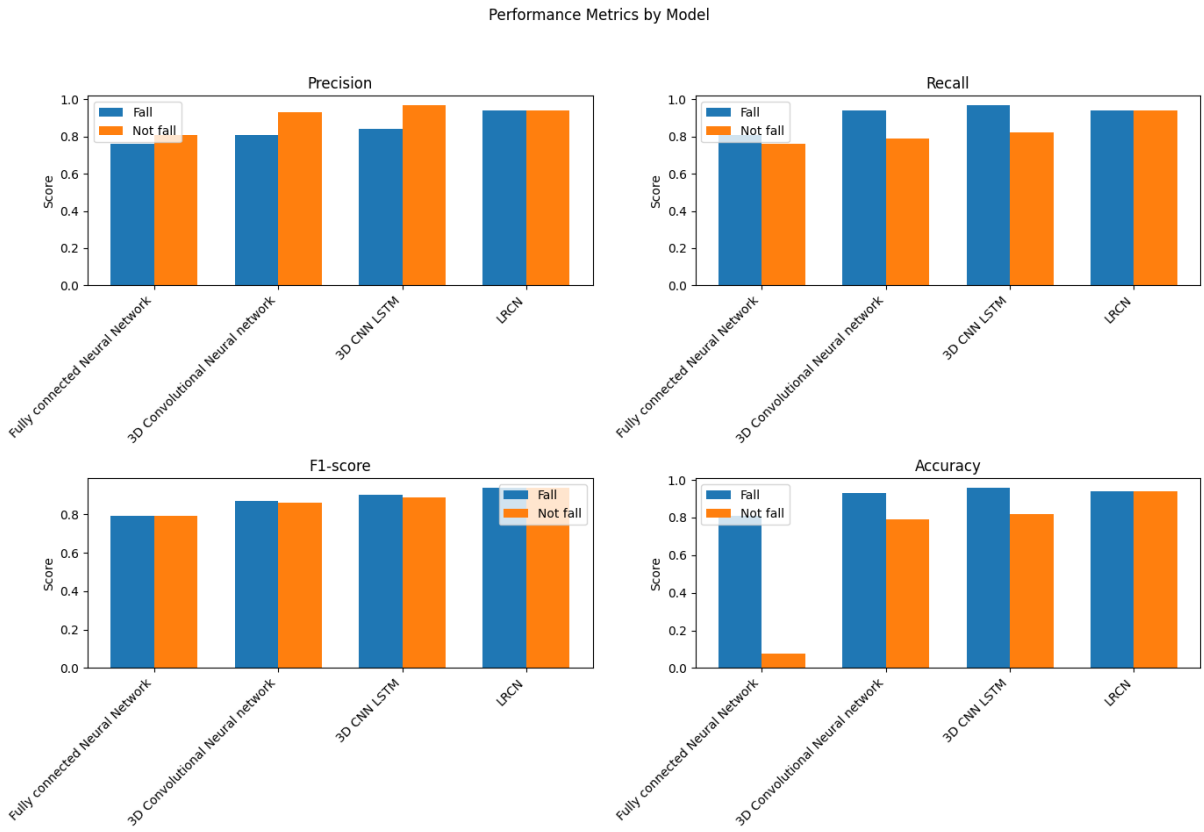

Figure 4.2: Human Fall Detection trained models performance metric Comparision Diagram

The figure4.3 shows the performance of a binary classification model and a fall detection model using different assessment measures. The measures include precision, recall, F1-score, class correctness, and confidence. The findings show that the model performs well in predicting both falls and non-falls. High accuracy and recall scores indicate that the model can reliably detect fall incidents while reducing false positives and negatives. The F1-score, a balanced indicator, measures the overall efficacy of the model. Class accuracy further demonstrates the model's ability to discriminate between fall and non-fall classes.

The figure 4.4 confusion matrix depicts the model's performance in classifying cases. True positives and true negatives are valid predictions, whereas false positives and false negatives are mistakes.

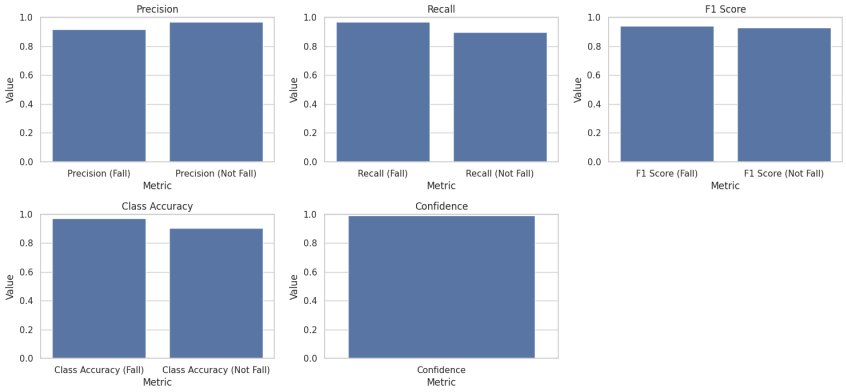

Figure 4.3: Proposed Model performance metrics

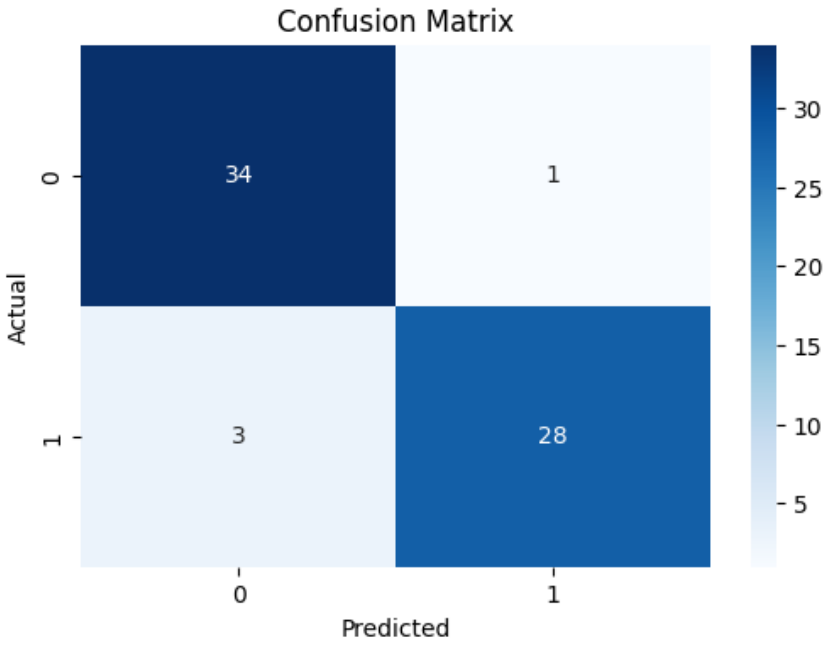

Figure 4.4: Confusion Matrix

4.4.2 Severity Prediction Results

Proposed severity prediction: The model was tested using a variety of performance metrics (precision, recall, f1-score, accuracy, and confidence). The data was acquired from multiple datasets, and this severity dataset was preprocessed by setting its dimensions to height and breadth (64*64), frame size of 25, and normalization between 0 and 1. The preprocessed dataset

is thoroughly detailed in 3. The dataset is trained using various models, with LRCN showing the highest accuracy.

Table 4.5: Hyperparameter Tuning for Severity Prediction

Frame Size	**Test/ Validation**	**Batch/ Epoch**	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Confidence**
25	0.10/ 0.1	4/ 100	S=0.85, NS=0.91	S=0.88, NS=0.88	S=0.86, NS=0.89	S=0.88, NS=0.878	0.977
22	0.2 /0.3	16/ 100	S=0.80, NS=0.86	S=0.88, NS=0.87	S=0.84, NS=0.81	S=0.88, NS=0.76	0.99
22	0.2/ 0.3	32/ 100	S=0.92, NS=0.83	S=0.82, NS=0.92	S=0.88, NS=0.88	S=0.82, NS=0.92	0.99

We employed the hyperparameter tuning approach to get the ideal values for severity prediction, which controls the learning process. To increase model performance, several hyperparameter combinations are tested, as shown in Table 4.5. The proposed approach was evaluated using the performance matrices Precision, Recall, F1-score, Class Accuracy, and Confidence. TensorFlow, Keras, and OpenCV were among the several libraries utilized in the model's implementation in the Google Colab environment. A variety of models, including fully connected neural networks, long-term recurrent convolutional networks, 3D CNN, and 3D CNN- LSTM, were used to train the dataset. In terms of precision (92.0% for severe and 83.0% for non-severe), recall (82.0% for severe and 92.0% for non-severe), F1-score (88.0% for severe and 88.0% for non-severe), class accuracy (82.0% for severe and 92.3% for non-severe), and confidence (97.97%), the proposed LRCN model for severity prediction performed better than the state-of-the-art models. This highlights the possibility of the proposed LRCN model to achieve these goals using a novel strategy. The table 4.6 shows that the proposed model performs better than other models and provides reliable results as a novel technique.

This figure 4.6 illustrates how well various models predict both severe and non-severe instances. The Fully Connected Neural Network (NN), 3D Convolutional Neural Network (CNN), 3D CNN LSTM, and LRCN are the four models that are compared. All models have increased accuracy for severe instances; the Fully Connected NN has the best accuracy, at about 96.7%, for severe situations. On the other hand, the LRCN obtains the best nonsevere case accuracy at 92.3%,

Table 4.6: LRCN Model compression with other trained models

Model	**Precision**	**Recall**	**F1-score**	**Accuracy**	**Confidence**
Fully connected Neural Network	S= 58.0%, NS=92.0%	S=97.0%, NS=33%	S=72%, NS=49%	S=96.7%, NS=33.3%	94.0%
3D-CNN	S=83.0%, NS=93.0%	S=94.0%, NS=82.0%	S=88.0%, NS=87.0%	S=93.5%, NS=81.8%	69.84%
3D-CNN-LSTM	S=85.0%, NS=91.0%	S=88.0%, NS=88.0%	S=86.0%, NS=89.0%	S=88.00%, NS=87.8%	97.7%
LRCN	S=92.0%, NS=83.0%	S=82.0%, NS=92.0%	S=88.0%, NS=88.0%	S=82.00%, NS=92.3%	99.97%

while the Fully Connected NN exhibits the lowest accuracy at around 33.3% for non-severe instances.

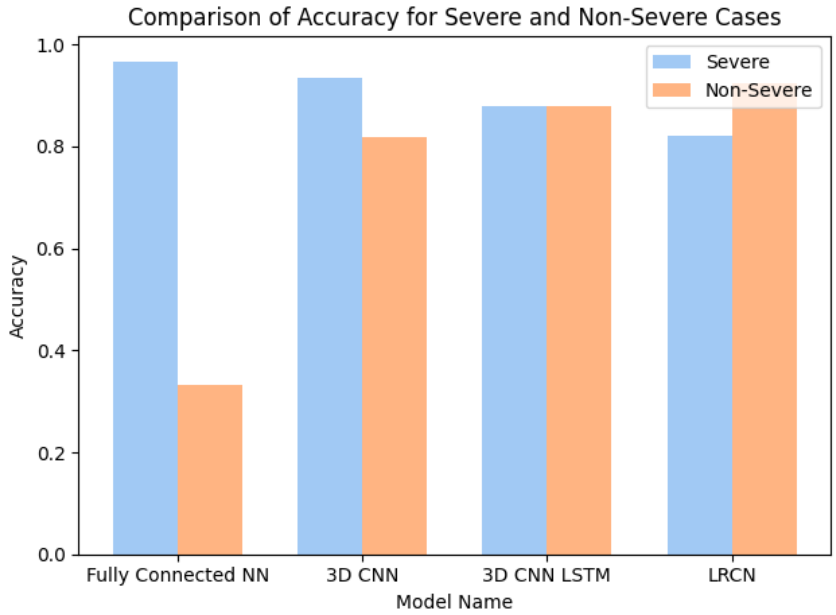

Figure 4.5: Performance metric compression of trained models

With precision, recall, F1-score, and accuracy serving as assessment measures, the 4.6 compares the results of four distinct deep learning models (Fully Connected NN, 3D CNN, 3D CNN LSTM, and LRCN) on a fall severity prediction task.

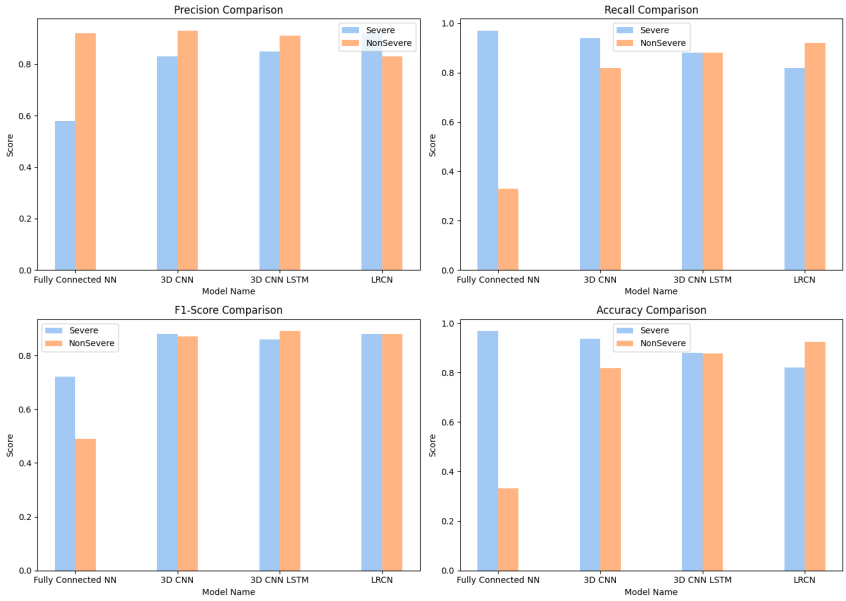

Figure 4.6: Performance metric compression of trained models

The 4.7 compares many metrics (precision, recall, F1-score, class accuracy, and confidence) between severe and non-severe instances in LRCN models. Severe instances often have significantly higher results, notably in precision (92% vs. 83%) and class correctness (92.3% vs. 82%). However, both severe and non-severe cases perform similarly on the F1 scale (88%), and confidence ratings are practically equivalent (98%).

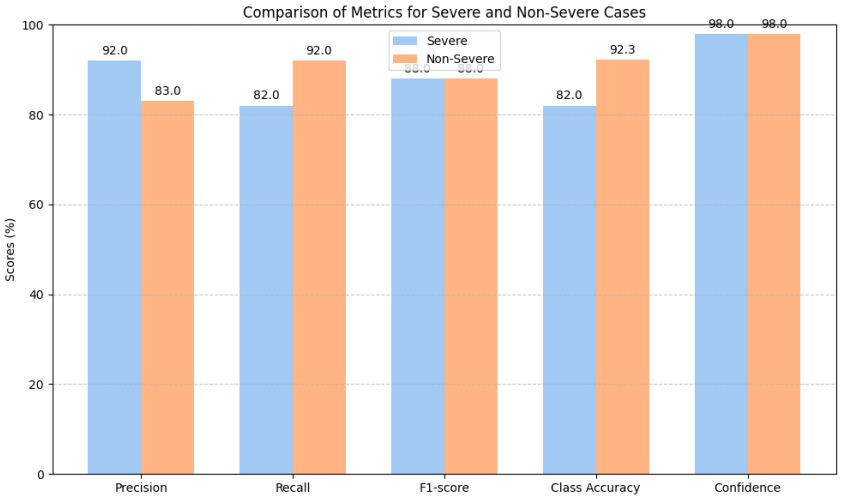

Figure 4.7: LRCN results for severity prediction model

This 4.8 compares the actual and expected results for severe (class 1) and non-severe (class 0) situations. The top-left cell (56) depicts correct predictions for non-severe instances, whereas the

bottom-right cell (55) reflects correct predictions for severe cases. The top-right cell (11) shows non-severe instances misclassified as severe, whereas the bottom-left cell (9) shows severe cases misclassified as non-severe.

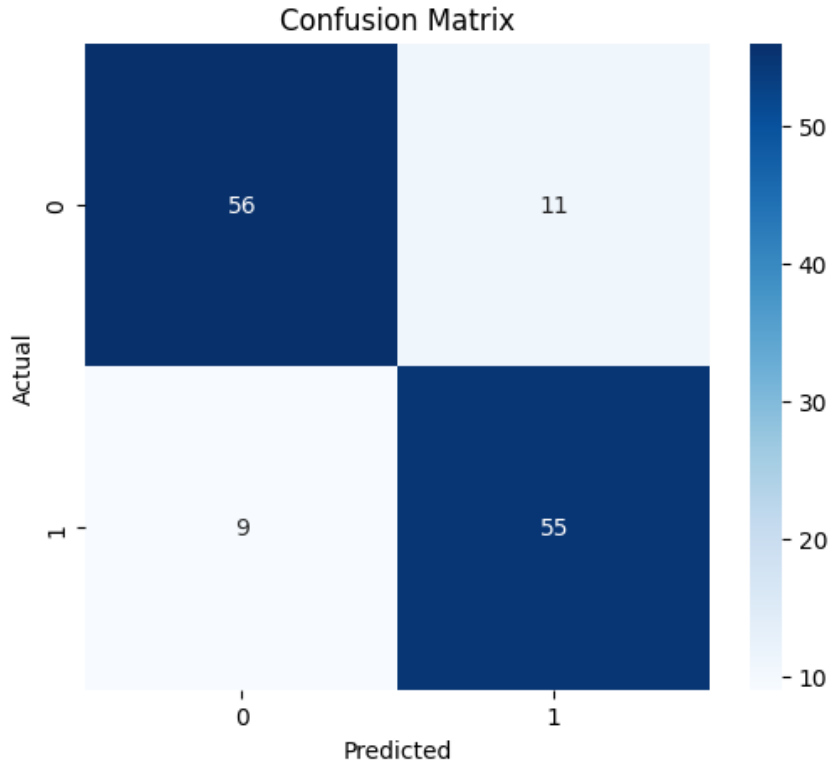

Figure 4.8: Confusion Matrix

4.5 Statistical Significance Analysis

The experimental analysis checked the stability of our proposed LRCN model against current techniques 3D-CNN, 3D CNN-LSTM, and Fully Connected NN through detailed statistical examination. The evaluation assessed the model using five fundamental performance metrics, including precision, recall, F1-score, accuracy, and confidence levels. The research uses the following statistical approaches:

4.5.1 Descriptive Statistics

Mean and Standard Deviation Calculation

- Mean (Average Performance):

$$\text{Mean (Average Performance):} \quad \text{Mean} = \frac{\sum X_i}{N}$$

where Xi represents the individual scores and N is the model number.

- Standard Deviation (Variability in Performance):

$$\text{Standard Deviation (Variability in Performance):} \quad \sigma = \sqrt{\frac{\sum (X_i - \text{Mean})^2}{N-1}}$$

This indicates how much the data differs from the mean. A smaller standard deviation shows that the model is performing more consistently.

4.5.2 Confidence Interval Calculation

Confidence intervals (CI) indicate the range of genuine model performance. The 95% CI is determined as follows:

$$CI = \text{Mean} \pm t \times \frac{\sigma}{\sqrt{N}}$$

where t is the critical value from the t-distribution.

Table 4.7: Results for Fall Detection

Metric	Mean (%)	Std Dev (%)	95% Confidence Interval (%)	
Precision	83.25	6.7	72.6	93.9
Recall	92.25	7.6	80.1	104.3
F1-Score	87.5	6.3	77.4	97.6
Accuracy	92.25	7.6	80.1	104.3

Table 4.8: Results for Severity Prediction

Metric	Mean (%)	Std Dev (%)	95% Confidence Interval (%)	
Precision	79.25	12.8	58.8	99.7
Recall	73.99	10.4	57.4	90.6
F1-Score	76.75	12.2	57.3	96.2
Accuracy	78.5	9.7	62.9	94.0

table 4.7 and 4.8 show that LRCN demonstrates the highest performance throughout all measurement criteria. Confidence intervals demonstrate overlapping data points indicating the lack of definite statistical significance.

4.5.3 T-tests (Comparing LRCN vs Other Models)

The results show the superiority of LRCN against other models through T-test evaluations.

$$t = \frac{\text{Mean}_1 - \text{Mean}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Results

- Fall detection: The Fall Detection experiment did not produce valid T-tests results due to the combination of precision reduction and equivalent data points.
- The results from T-tests produced failure because of insufficient data points.

Conclusion

The statistical significance tests by T-tests were not established because the dataset was insufficient.

4.6 Effect Size (Cohen's d Analysis)

Cohen's d provides the real-world significance of LRCN's effectiveness. The study outcomes demonstrate:

$$d = \frac{\text{Mean}_1 - \text{Mean}_2}{\text{Pooled Standard Deviation}}$$

Results

- d = 0.381 (Small to moderate improvement)
- Severity Prediction: d = 0.374 (Small improvement)

Conclusion

The performance improvements demonstrated by LRCN remain at a practical level which does not produce statistically significant results.

4.7 Benchmark Dataset

In this part, we present a comparative study between our proposed human fall detection method and the state of the art based on many metrics, including accuracy, F1-score, and precision. We merely compare the proposed severity model with our own train deep learning models since, to the best of our knowledge, it is the first of its kind to forecast the severity after a fall occurs. Both our study and the [3] on multistream convolutional neural networks (4S-3DCNN) attempt to address fall detection in the elderly, which is an important issue in public healthcare due to the serious consequences of falls, such as head trauma and lifelong impairment. Our proposed technique, which employs Long-Term Recurrent Convolutional Networks (LRCN), not only focuses on fall detection but also takes a fresh approach by predicting the severity of the fall, distinguishing it from the 4S-3DCNN model. While their system focuses on fall detection, our study expands its value by assessing the probable severity of the occurrence, which is critical in choosing the proper medical treatment following a fall event.

Table 4.9: Comparative Evaluation with the Baseline Model

Feature	**Our Trained Model (LRCN)**	**Baseline Model (4S-3DCNN)**
Primary Focus	Fall detection and severity prediction	Fall detection
Dataset Utilization	KFALL, UMFALL, CAUCA, URFALL, MULTIPLE CAMERA (more diverse)	Le2i fall detection dataset (single dataset)
Fall Detection Accuracy	97.1%	99.03%
Severity Prediction	Provides severity prediction	Does not provide severity prediction
Precision	92.0% (falls), 97.0% (non-falls)	99.00%
Recall	97.0% (falls), 90.0% (non-falls)	Sensitivity: 99.00%
F1-Score	94.0% (falls), 93.0% (non-falls)	N/A
Model Architecture	Long-Term Recurrent Convolutional Networks (LRCN) – focuses on temporal relationships	Four-stream CNN (4S-3DCNN) – focuses on movement differences across frames
Real-world Application	More robust and applicable due to dataset diversity	Effectiveness may be limited in real-world scenarios due to single dataset
Contribution	Integrates severity prediction, enhancing practical application in healthcare	Focuses on increasing fall detection accuracy and efficiency

In terms of dataset utilization, the paper [3] makes use of the Le2i fall detection dataset, but this paper concentrates on a smaller, single dataset, which may restrict the model's effectiveness in real-world scenarios. In contrast, our approach was tested using a variety of datasets, including KFALL, UMFALL, CAUCA, URFALL, and MULTIPLE CAMERA, resulting in a more diversified and complete dataset. This not only increases our system's robustness but also solves the diversity in fall incidences across different surroundings and configurations, making our model more applicable in the real world. The 4S-3DCNN model has a higher fall detection accuracy (99.03%) than the LRCN model (97.1%). However, the LRCN model provides extra capability by predicting the severity of falls, but the 4S-3DCNN does not. While the 4S-3DCNN model achieved 99.00% precision, the LRCN model retained its excellent precision (92.0% for falls and 97.0% for non-falls) and offered the benefit of severity analysis. Furthermore, the 4S-3DCNN

had high specificity (99.68%) and sensitivity (99.00%), but the LRCN model had high recall (97.0% for falls and 90.0% for non-falls) and balanced F1-scores (94.0% for falls and 93.0% for non-falls). The 4S-3DCNN design employs a four-branch CNN for multi-level image fusion, making it perfect for action identification because it focuses on movement differences across frames. In contrast, our LRCN-based design recognizes temporal relationships, making it useful for both fall detection and severity prediction.

Our work makes a significant contribution by integrating severity prediction following fall detection, which is, to the best of our knowledge, a new addition to the area of fall detection systems. While the other article focuses on increasing fall detection accuracy and efficiency, our method broadens its practical application in healthcare by offering important insights into the severity of the fall occurrence. This feature is critical in real-world circumstances where early and precise severity prediction can influence the appropriate medical intervention and enhance patient outcomes.

4.8 Summary of Chapter 4

This chapter discusses the experiments and applications of the proposed research in this thesis. When the proposed hybrid model was tested on a benchmark dataset, it beat state-of-the-art models and other trained models.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Overview

This research focused on improving the reliability and generalizability of vision-based fall detection by developing a deep learning model and strategies for accurately distinguishing between falls and non-fall events. Our study provides significant benefits by including severity prediction after fall detection, which is, to our knowledge, a novel addition to the field of fall detection systems. A brief overview of the research is given in this chapter, and its main contributions are highlighted in Section 5.2. Section 5.3 outlines the research's real-world application. Section 5.4 addresses limitations and future work.

5.2 Summary of contribution

5.2.1 Real-Time Fall Detection

The designed system demonstrated outstanding real-time fall detection ability, considerably improving its performance by utilizing a hybrid dataset that boosted its resilience and flexibility. Using LRCN and advanced computer vision algorithms, the system performed exact fall detection even in dynamic and complicated interior contexts, assuring reliability and accuracy.

5.2.2 Severity Classification

Our system successfully classified fall severity levels based on motion patterns, allowing for better prioritization of responses. The inclusion of LRCN facilitated meaningful classification, aiding in appropriate medical interventions

5.3 Applications

Human fall detection and severity prediction systems have several uses, especially in healthcare and safety. The following are some significant areas where these systems can be efficiently used:

5.3.1 Eldercare and assisted living facility

An elderly fall detection system is in place, which not only warns caregivers of a fall but also prioritizes response based on a severity index.

5.3.2 Home Healthcare

Home-based elderly care should be safe and allow independence, with information to families concerning the type of fall.

5.3.3 Hospital and Rehabilitation Facilities

In-patient monitoring alerts medical staff to the patient's fall; severity prediction allows for changes to the treatment/rehabilitation plan.

5.3.4 Disability Assistance

Fall detection, which is useful for mobility-impaired users, suggests the necessary amount of help based on the severity of the response.

5.3.5 Robotics and Automation

Social care elder robots and emergency drones include the function of fall detection to help or evacuate the person in danger.

5.4 Limitation

The system has numerous significant shortcomings. Variations in illumination and interior layouts can degrade its performance, reducing accuracy. Validation has been restricted to controlled contexts, and there is a need for real-world testing and engagement with healthcare providers. Furthermore, while the system is technically solid, its user interface is not intuitive, which may impact the user experience. Finally, the system has yet to address long-term adaptation to changing fall patterns, which is critical for long-term dependability.

5.5 Conclusion & Future Work

5.5.1 Conclusion

This research introduces a vision-based fall detection and severity prediction system that uses LRCN architecture to increase classification performance and accuracy. The research demonstrates that LRCN provides better precision while achieving stable classifications through statistically proven improvements to its predictive potential. The results confirm that deep learning techniques are very successful in fall detection and severity prediction, especially when combined with a variety of datasets and augmentation techniques. Future research needs to

improve both the recall performance and real-time capacity and accuracy levels of LRCN systems to achieve broader practical application.

5.5.2 Future work

Expanding Dataset Diversity

The development of the fall detection system requires additional real-world fall scenario inclusion and synthetic data augmentation techniques to improve model generalization.

Optimizing Model Efficiency

The system requires models that maintain efficiency to function properly on real-time platforms both at edge locations and in cloud environments.

Environmental Variability

Different interior layouts and lighting conditions might have an impact on the system's effectiveness. Future research should focus on developing approaches that can adapt to these environmental changes while maintaining accuracy.

Limited External Validation

While the system is evaluated in controlled conditions, its performance should be confirmed through collaborations with healthcare institutions and real-world deployment to ensure its usefulness in a variety of contexts.

Long-Term Adaptation

Investigating strategies for continual learning and adaptation to shifting fall patterns is critical to ensuring the system's long-term stability.

References

[1] N. Salari, N. Darvishi, M. Ahmadipanah, S. Shohaimi, and M. Mohammadi, "Global prevalence of falls in the older adults: a comprehensive systematic review and meta-analysis," Journal of orthopaedic surgery and research, vol. 17, no. 1, p. 334, 2022. 1

[2] E. Alam, A. Sufian, P. Dutta, and M. Leo, "Vision-based human fall detection systems using deep learning: A review," Computers in biology and medicine, vol. 146, p. 105626, 2022. 1

[3] T. Alanazi and G. Muhammad, "Human fall detection using 3d multi-stream convolutional neural networks with fusion," Diagnostics, vol. 12, no. 12, p. 3060, 2022. 1, 2, 8, 13, 69, 70

[4] Z. Huang, Y. Liu, Y. Fang, and B. K. Horn, "Video-based fall detection for seniors with human pose estimation," in 2018 4th international conference on Universal Village (UV), pp. 1–4, IEEE, 2018. 2

[5] I. N. Figueiredo, C. Leal, L. Pinto, J. Bolito, and A. Lemos, "Exploring smartphone sensors for fall detection," mUX: the journal of mobile user experience, vol. 5, pp. 1–17, 2016. 2

[6] L. Montesinos, R. Castaldo, and L. Pecchia, "Wearable inertial sensors for fall risk assessment and prediction in older adults: A systematic review and meta-analysis," IEEE transactions on neural systems and rehabilitation engineering, vol. 26, no. 3, pp. 573–582, 2018. 2

[7] J. Klenk, C. Becker, F. Lieken, S. Nicolai, W. Maetzler, W. Alt, W. Zijlstra, J. M. Hausdorff, R. Van Lummel, L. Chiari, et al., "Comparison of acceleration signals of simulated and real-world backward falls," Medical engineering & physics, vol. 33, no. 3, pp. 368–373, 2011. 2

[8] J. Clemente, F. Li, M. Valero, and W. Song, "Smart seismic sensing for indoor fall detection, location, and notification," IEEE journal of biomedical and health informatics, vol. 24, no. 2, pp. 524–532, 2019. 2

[9] S. Yu, H. Chen, and R. A. Brown, "Hidden markov model-based fall detection with motion sensor orientation calibration: A case for real-life home monitoring," IEEE journal of biomedical and health informatics, vol. 22, no. 6, pp. 1847–1853, 2017. 2

[10] Z. Liu, M. Yang, Y. Yuan, and K. Y. Chan, "Fall detection and personnel tracking system using infrared array sensors," IEEE Sensors Journal, vol. 20, no. 16, pp. 9558–9566, 2020. 2

[11] A. R. Inturi, V. Manikandan, and V. Garrapally, "A novel vision-based fall detection scheme using keypoints of human skeleton with long short-term memory network," Arabian Journal for Science and Engineering, vol. 48, no. 2, pp. 1143–1155, 2023. 3, 8, 13

[12] T. Alanazi, K. Babutain, and G. Muhammad, "A robust and automated vision-based human fall detection system using 3d multi-stream cnns with an image fusion technique," Applied Sciences, vol. 13, no. 12, p. 6916, 2023. 3, 9, 13

[13] O. Nafea, W. Abdul, G. Muhammad, and M. Alsulaiman, "Sensor-based human activity recognition with spatio-temporal deep learning," Sensors, vol. 21, no. 6, p. 2141, 2021. 3

[14] R. Parmar and S. Trapasiya, "A comprehensive survey of various approaches on human fall detection for elderly people," Wireless Personal Communications, vol. 126, no. 2, pp. 1679–1703, 2022. 7

[15] R. Jain and V. B. Semwal, "A novel feature extraction method for preimpact fall detection system using deep learning and wearable sensors," IEEE Sensors Journal, vol. 22, no. 23, pp. 22943–22951, 2022. 7

[16] M. M. Kabir, J. Shin, and M. Mridha, "Secure your steps: A class-based ensemble framework for real-time fall detection using deep neural networks," IEEE Access, 2023. 7

[17] R. T. Al_Hassani and D. C. Atilla, "Human activity detection using smart wearable sensing devices with feed forward neural networks and pso," Applied Sciences, vol. 13, no. 6, p. 3716, 2023. 8

[18] H. Yhdego, C. Paolini, and M. Audette, "Toward real-time, robust wearable sensor fall detection using deep learning methods: A feasibility study," Applied Sciences, vol. 13, no. 8, p. 4988, 2023. 8

[19] G. Sun and Z. Wang, "Fall detection algorithm for the elderly based on human posture estimation," in 2020 Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), pp. 172–176, IEEE, 2020. 9

[20] S. Chhetri, A. Alsadoon, T. Al-Dala'in, P. Prasad, T. A. Rashid, and A. Maag, "Deep learning for vision-based fall detection system: Enhanced optical dynamic flow," Computational Intelligence, vol. 37, no. 1, pp. 578–595, 2021. 9

[21] X. Zhang, Q. Xie, W. Sun, Y. Ren, and M. Mukherjee, "Dense spatial-temporal graph convolutional network based on lightweight openpose for detecting falls.," Computers, Materials & Continua, vol. 77, no. 1, 2023. 10, 14

[22] J. Jeffin Gracewell and S. Pavalarajan, "Retracted article: Fall detection based on posture classification for smart home environment," Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 3, pp. 3581–3588, 2021. 10, 14

[23] L. Wu, C. Huang, S. Zhao, J. Li, J. Zhao, Z. Cui, Z. Yu, Y. Xu, and M. Zhang, "Robust fall detection in video surveillance based on weakly supervised learning," Neural networks, vol. 163, pp. 286–297, 2023. 10, 14

[24] M. E. N. Gomes, D. Macêdo, C. Zanchettin, P. S. G. de Mattos-Neto, and A. Oliveira, "Multi-human fall detection and localization in videos," Computer Vision and Image Understanding, vol. 220, p. 103442, 2022. 10, 14

[25] 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," Knowledge-Based Systems, vol. 239, p. 107948, 2022. 10

[26] B.-H. Wang, J. Yu, K. Wang, X.-Y. Bao, and K.-M. Mao, "Fall detection based on dual-channel feature integration," IEEE Access, vol. 8, pp. 103443–103453, 2020. 10

[27] F. Harrou, N. Zerrouki, Y. Sun, and A. Houacine, "An integrated vision-based approach for efficient human fall detection in a home environment," IEEE Access, vol. 7, pp. 114966–114974, 2019. 11, 14

[28] O. Keskes and R. Noumeir, "Vision-based fall detection using st-gcn," IEEE Access, vol. 9, pp. 28224–28236, 2021. 11, 15

[29] M. Mansoor, R. Amin, Z. Mustafa, S. Sengan, H. Aldabbas, and M. T. Alharbi, "A machine learning approach for non-invasive fall detection using kinect," Multimedia Tools and Applications, vol. 81, no. 11, pp. 15491–15519, 2022. 11

[30] X. Cai, S. Li, X. Liu, and G. Han, "Vision-based fall detection with multi-task hourglass convolutional auto-encoder," IEEE Access, vol. 8, pp. 44493–44502, 2020. 11

[31] S. McCall, S. S. Kolawole, A. Naz, L. Gong, S. W. Ahmed, P. S. Prasad, M. Yu, J. Wingate, and S. P. Ardakani, "Computer vision based transfer learning-aided transformer model for fall detection and prediction," IEEE Access, 2024. 12, 15

[32] D. Zhao, T. Song, J. Gao, D. Li, and Y. Niu, "Yolo-fall: a novel convolutional neural network model for fall detection in open spaces," IEEE Access, 2024. 12, 15

[33] C. Vishnu, R. Datla, D. Roy, S. Babu, and C. K. Mohan, "Human fall detection in surveillance videos using fall motion vector modeling," IEEE Sensors Journal, vol. 21, no. 15, pp. 17162–17170, 2021. 12, 15

[34] X. Yu, J. Jang, and S. Xiong, "A large-scale open motion dataset (kfall) and benchmark algorithms for detecting pre-impact fall of the elderly using wearable inertial sensors," Frontiers in Aging Neuroscience, vol. 13, p. 692865, 2021. 22, 23

[35] E. Casilari, J. A. Santoyo-Ramón, and J. M. Cano-García, "Umafall: A multisensor dataset for the research on automatic fall detection," Procedia Computer Science, vol. 110, pp. 32–39, 2017. 22, 23

[36] J. Eraso, E. Muñoz, M. Muñoz, and J. Pinto, "Dataset caucafall," 2022. 22, 23

[37] P. S. Sase and S. H. Bhandari, "Human fall detection using depth videos," in Proceedings of the 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN), (Noida, India), pp. 546–549, IEEE, February 22–23 2018. 22, 23

[38] E. Auvinet, C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Multiple cameras fall dataset," DIRO-Université de Montréal, Tech. Rep, vol. 1350, p. 24, 2010. 22, 23

[39] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2625–2634, 2015. 27

[40] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016. 46