Project Stage-II

on

Real-Time Human Fall Detection using YOLOv8 and OpenCV

Submitted

in partial fulfilment of the requirements for

the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

(Artificial Intelligence and Machine Learning)

By

ABHIJITH KORUKONDA (20261A6602)

IRAGAVARAPU SRI RAMA SAKETH (20261A6626)

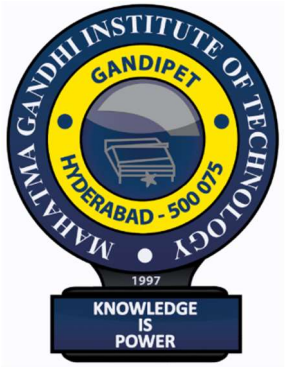
Under the Guidance of

Dr. M. Rama Bai

Professor

Ms. J. Sreedevi

Asst. Professor



DEPT. OF EMERGING TECHNOLOGIES

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY(Autonomous)

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad)

Gandipet, Hyderabad-500075, Telangana, INDIA

2023 – 2024

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY(A)

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)
GANDIPET, HYDERABAD, TELANGANA– 500 075 (INDIA)

CERTIFICATE

This is to certify that the project entitled **Real-Time Human Fall Detection using YOLOv8 and OpenCV** is being submitted by **Mr. ABHIJITH KORUKONDA** bearing **Roll No: 20261A6602** and **Mr. IRAGAVARAPU SRI RAMA SAKETH** bearing **Roll No: 20261A6626** in partial fulfilment for the award of **Bachelor of Technology** in **Computer Science and Engineering (Artificial Intelligence and Machine Learning)** to **Jawaharlal Nehru Technological University Hyderabad** is a record of bonafide work carried out by them under our guidance and supervision.

The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

Project Guides

Dr. M. Rama Bai
Professor and HoD

Ms. J. Sreedevi
Assistant Professor

Head of the Department

Dr. M. Rama Bai
Professor

EXTERNAL EXAMINER

DECLARATION

This is to certify that the work reported in this project titled **Real-Time Human Fall Detection with YOLOv8 and OpenCV** is a record of work done by us in the Department of Emerging Technologies, Mahatma Gandhi Institute of Technology, Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred to in the text. The report is based on the work done entirely by us and not copied from any other source.

ABHIJITH KORUKONDA
(20261A6602)

I. SRI RAMA SAKETH
(20261A6626)

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. G. ChandraMohan Reddy, Principal MGIT**, for providing the working facilities in college.

We wish to express our sincere thanks and gratitude to **Dr. Rama Bai, Professor and HoD**, Department of ET, MGIT, for all the timely support and valuable suggestions during the period of project.

We are also extremely thankful to our Project Coordinators, **Dr. D. Koteswara Rao, Assistant Professor,** Dept. of ET and **Mr. R. Srinivas, Assistant Professor,** Dept.of ET**,** for their valuable suggestions and interest throughout the course of this project

We are extremely thankful to our internal guides **Dr. Rama Bai, Professor and HoD**, Dept. of ET, MGIT and **Ms. J. Sreedevi, Assistant Professor,** Department of ET, MGIT for their constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank all the faculty and staff of the ET Department who helped us directly or indirectly, for completing this project.

ABHIJITH KORUKONDA
(20261A6602)

I. SRI RAMA SAKETH
(20261A6626)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Falls among individuals pose serious risks to their health and independence, underscoring the importance of effective fall detection solutions. This study aims to address this critical issue by proposing a novel approach that integrates Computer Vision and Deep Learning for Real-time fall detection and assistance. Traditionally, fall detection systems have relied on wearable sensors, which, despite their widespread use, often suffer from drawbacks such as false alarms and discomfort for the wearer. In response to these limitations, this project introduces an efficient solution by leveraging Computer Vision in conjunction with Arduino microcontroller. The core of this innovative system lies in the integration of the YOLOv8 (You Only Look Once) which is a cutting-edge, real-time object detection algorithm that uses Convolutional Neural Network (CNN) to predict the bounding boxes and class probabilities of objects in input images with Computer Vision. YOLOv8, a variant of the YOLO object algorithm series, has demonstrated superior performance in identifying various objects, and therefore has been used in detecting fall events, with remarkable accuracy and efficiency.

Complementing the advanced capabilities of the YOLOv8 algorithm, Arduino provide a versatile and cost-effective platform for implementing real-time monitoring and response mechanisms. In this study, Arduino board serve as a processing unit that receive input from cameras, analyse the data using the YOLOv8 algorithm, and trigger appropriate actions, such as buzzing a buzzer and displaying text on an LCD display to signal a fall event. By combining the strengths of YOLOv8, Computer Vision and Arduino, this innovative solution promises to revolutionize fall detection technology. This improves accuracy and reliability in identifying falls and enhances the user experience by offering timely assistance, ensuring individuals' safety and well-being.

1. INTRODUCTION

The incidence of falls presents a noteworthy obstacle for a variety of demographic groups in the healthcare sector and beyond. Accidental falls can affect people of all ages and in a variety of contexts, resulting in serious injuries and health issues. Falls may cause severe morbidity and mortality and are a major contributor to unintentional injury worldwide.

On a global scale, falls are the second most common cause of accidental deaths. Falls are also considered the most likely reason for older adults to develop traumatic brain injuries. On an annual basis, around 30% of people fall at least once a year, and the rate of falling is expected to increase to 42% over the next few years [1]. Every 11 s, a person who has fallen has to be taken to the emergency room. Every 19 min, an older adult dies from a fall in the US. There are over 3 million cases of emergency, 800,000 hospitalizations, and more than 32,000 deaths from falls every year. This rate of hospitalizations and deaths from falls is expected to increase. It is predicted that by 2030, every hour, seven older adults will die in the US after having experienced a fall [1]. The primary factor contributing to this alarming statistic is the lack of immediate and timely attention. Timely attention to falls is paramount in preventing further complications.

In response to this critical issue, the development of real-time monitoring for human fall detection represents a pivotal advancement in safety technology. The initiative seeks to tackle the universal requirement for efficient fall detection systems capable of promptly recognizing falls and triggering immediate alerts.

The applications of such technology span across different sectors and environments. Healthcare facilities, including hospitals, clinics, and rehabilitation centres, can benefit from fall detection systems to enhance patient safety and enable quick response to medical emergencies. Similarly, industrial and workplace safety can be greatly improved by deploying fall detection technology in factories, construction sites, and manufacturing facilities to protect workers from accidents and reduce occupational hazards.

In remote and hazardous environments such as mines or offshore rigs, fall detection systems can aid in ensuring the safety of workers and facilitating rapid rescue operations. Moreover, individuals receiving home care, regardless of age, can be monitored using this technology, offering peace of mind to caregivers and family members [2].

Implementing effective fall detection measures can substantially reduce the incidence of fatal and severe injuries, ultimately improving the overall quality of life. By providing timely assistance and intervention, we aim to address this pressing issue and contribute to a safer and healthier environment.

1.1 Existing System

Existing fall detection systems have primarily relied on wearable sensors, such as accelerometers and gyroscopes, embedded in devices like wristbands, pendants, or belts worn by individuals. These sensors detect changes in acceleration and orientation, which are indicative of a fall. However, these systems often suffer from limitations such as high false alarm rates due to non-fall activities that mimic fall-like movements, as well as issues related to user compliance and comfort. Additionally, wearable devices may not be suitable for individuals with cognitive impairments or those who forget to wear or charge their devices regularly.

Some commercially available fall detection systems utilize machine learning algorithms trained on data collected from wearable sensors to improve accuracy in distinguishing between falls and non-fall activities. However, these systems still face challenges related to false alarms and may not always provide timely assistance in critical situations. Moreover, the reliance on wearable devices restricts the mobility and independence of individuals, as they may feel stigmatized or uncomfortable wearing conspicuous devices.

One notable example of an existing fall detection system is the Mobile Health Assistance Unit (MAHU) developed by researchers at the University of Missouri. MAHU employs a combination of accelerometers and gyroscopes integrated into a smartphone to detect falls. The system uses machine learning algorithms to analyze motion patterns and distinguish falls from other activities [3]. When a fall is detected, MAHU automatically sends alerts to designated caregivers or emergency services, providing timely assistance to the user. While MAHU represents a significant advancement in fall detection technology, it still faces challenges related to false alarms and user acceptance [3].

1.2 Problem Statement

Traditional wearable sensor-based systems exhibit drawbacks including high false alarm rates and user discomfort. This research proposes an innovative approach integrating YOLOv8 with Computer Vision and Arduino microcontrollers to address these limitations and improve fall detection accuracy and user experience.

1.3 Proposed System

The proposed fall detection system integrates computer vision and microcontroller technology to address elderly falls efficiently. Central to this solution is the YOLOv8 computer vision algorithm, renowned for accurately identifying various objects, including falls, with precision and efficiency. Leveraging YOLOv8 enables real-time fall detection, minimizing false alarms and ensuring prompt assistance.

Complementing computer vision, Arduino microcontrollers serve as central processing units, offering versatility and cost-effectiveness. They receive input from cameras or sensors, process data using YOLOv8, and trigger actions based on detected fall events. Arduino boards activate LED indicators and emit audible alerts, enhancing notification reliability even in noisy environments.

The system deploys cameras strategically in areas prone to falls, such as living spaces or healthcare facilities. These sensors capture real-time video, processed by YOLOv8 on computational units like computers or embedded systems. Processed data, indicating detected fall events, is transmitted to Arduino microcontrollers for immediate response.

By merging cutting-edge computer vision with versatile microcontrollers, this system promises to revolutionize fall detection. It offers improved accuracy and reliability, enhancing safety and well-being. Real-time monitoring ensures timely assistance, bolstering the independence and quality of life for individuals.

1.4 Scope of the work

The scope of this system also includes the deployment of the fall detection system in diverse environments such as assisted living facilities, construction sites, and recreational facilities where individuals may be at risk of falls. The system will be designed to operate effectively in

both indoor and outdoor settings, accommodating different lighting conditions and environmental factors. These efforts aim to establish a comprehensive and adaptable solution that enhances safety and promotes independence in various settings and demographics.

1.5 Significance

This system represents a transformative initiative in fall detection technology, delivering cutting-edge solutions that prioritize user comfort and privacy while enhancing safety across diverse sectors. By leveraging advanced technologies like computer vision and deep learning, the proposed system sets new standards in real-time fall detection across healthcare, industrial safety, and home care services. The goal is to empower individuals of all backgrounds to lead more secure and independent lives, fostering a societal shift towards proactive safety measures and inclusive technological solutions.

1.6 Requirements Specification

1.6.1 Software Requirements

Language	: Python
Operating system	: Windows or Linux or MAC
IDE	: Jupyter Notebook, Google Colab, Arduino IDE

1.6.2 Hardware Requirements

RAM	: 16GB minimum
Processor	: Intel Core-i5 minimum
ROM	: 256 GB
GPU Model	: NVIDIA T4 GPU
IoT Devices	: Arduino UNO R3
	16x2 LCD Display with I2C
	Sound Buzzer
	Jumper Wires

2. LITERATURE SURVEY

Potential research work carried out on various techniques for Fall Detection has been discussed in this section. Various researchers have employed different mechanisms for detecting fall events.

(i) A Study of Fall Detection in Assisted Living: Identifying and Improving the Optimal Machine Learning Method [1] by Thakur and Han explores the application of machine learning for fall detection in assisted living environments. Leveraging a K-Nearest Neighbors (KNN) classifier with AdaBoost on accelerometer data, the study demonstrates improved accuracy in detecting falls. Notably, the authors emphasize the consideration of real-world implementation factors, addressing the practical challenges associated with deploying fall detection systems. However, the study also highlights dependencies on wearable devices like accelerometers and the inherent difficulty in data preprocessing as potential disadvantages, indicating the need for further research to enhance the system's usability and efficiency in real-world settings.

(ii) A Machine Learning Approach for Fall Detection and Daily Living Activity Recognition [2] by Chelli and Pätzold present a machine learning-based approach utilizing QSVM (Quantum Support Vector Machine) and EBT (Extreme Learning Machine Boosting Tree) algorithms for fall detection and daily living activity recognition. The incorporation of these algorithms enhances the accuracy of the system, contributing to the robustness of fall detection. The authors particularly emphasize the advantages of real-time monitoring in enhancing safety, providing timely alerts and interventions. However, the study acknowledges certain limitations, including a dependency on wearable devices such as accelerometers and the inherent difficulty in data preprocessing. These considerations highlight the need for further exploration and optimization to address challenges and ensure the practical feasibility of their machine learning approach in real-world scenarios.

(iii) Deep Learning for Vision-Based Fall Detection System: Enhanced Optical Dynamic Flow [3] authored by Chhetri et al., the authors propose a vision-based fall detection system leveraging deep learning techniques, specifically Support Vector Machine (SVM) and Convolutional Neural Network (CNN). The integration of these algorithms results in a substantial improvement in accuracy. Notably, the system exhibits a notable advantage by eliminating dependence on wearable devices, a common constraint in fall detection systems.

However, the study acknowledges the difficulty associated with dimensionality reduction, posing a challenge in streamlining and optimizing the computational complexity of the proposed vision-based approach. These findings underscore the significant advancements in accuracy and user-friendliness, while also highlighting areas for further research and refinement in addressing computational complexities.

(iv) Combining Domain Knowledge and Machine Learning for Robust Fall Detection [4] authored by Violeta Mirchevska, Mitja Luštrek, and Matjaž Gams, the authors explore a holistic approach by integrating domain knowledge with machine learning techniques to enhance the robustness of fall detection systems. The incorporation of domain knowledge, which encompasses a deep understanding of the specific environment and context in which the system operates, is emphasized as a key element in achieving robust results. This synergy between domain knowledge and machine learning aims to address the complexity and computational demands inherent in fall detection scenarios. The study advocates for a comprehensive strategy that leverages both the inherent understanding of the domain and the power of machine learning algorithms to create a more effective and resilient fall detection system. The incorporation of domain knowledge contributes to a nuanced understanding of the intricacies involved in fall events, while the integration with machine learning techniques seeks to optimize the computational efficiency of the overall system.

(v) Transfer Learning and Information Retrieval Applied to Fall Detection [5] by Mirko Fañez, José R. Villar, Enrique de la Cal, Javier Sedano, and Victor M. González, the authors delve into the application of transfer learning and information retrieval techniques for fall detection. The work introduces the use of Symbolic Aggregate approximation, a method that leverages symbolic representations for time series data. The incorporation of this technique contributes to achieving high detection rates, enhancing the sensitivity and accuracy of fall detection. However, the study acknowledges the computational complexity associated with the proposed methods, highlighting the need for considerations and optimizations in handling large-scale datasets and real-time monitoring scenarios. These findings underline the advancements in detection rates while emphasizing the importance of addressing computational challenges for practical deployment in fall detection applications.

(vi) A Systematic Review on Machine Learning for Fall Detection System [6] by Shikha Rastogi and Jaspreet Singh provide a comprehensive overview of the application of machine learning techniques in fall detection systems. The review focuses on Support Vector Machine (SVM) and Convolutional Neural Network (CNN) as prominent algorithms. The highlighted outcomes include an improved accuracy in fall detection, emphasizing the effectiveness of machine learning in enhancing system performance. However, the study notes the inherent challenge of complex data processing associated with the implementation of these algorithms. This systematic review serves as a valuable resource for understanding the landscape of machine learning in fall detection, shedding light on both the advantages and challenges in the pursuit of more accurate and reliable fall detection systems.

(vii) Deep Learning Based Systems Developed for Fall Detection: A Review [7] authored by M. M. Islam and colleagues, the authors conduct a comprehensive review of deep learning-based systems designed for fall detection. The review emphasizes the application of Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN) architectures. Notably, these deep learning approaches contribute to an improved accuracy in fall detection, showcasing the effectiveness of advanced neural network models. However, the study acknowledges certain challenges, including the dependency on wearable devices like accelerometers, which may introduce constraints related to user comfort and compliance. Additionally, the review highlights the difficulty associated with data preprocessing, suggesting complexities in preparing and refining the data for optimal model training. These insights from the review contribute to a holistic understanding of the advancements and challenges in leveraging deep learning for fall detection systems.

(viii) A Multimodal Approach Using Deep Learning for Fall Detection [8] by Yves M. Galvão, Janderson Ferreira, Vinícius A. Albuquerque, Pablo Barros, and Bruno J.T. Fernandes suggests a multimodal approach incorporating a Custom Convolutional Neural Network (CNN) is proposed for fall detection. The utilization of this custom CNN results in an improved accuracy in detecting falls, showcasing the efficacy of the multimodal deep learning strategy. However, the study notes dependencies on wearable devices such as accelerometers, which may introduce challenges related to user adoption and comfort. Additionally, the research acknowledges the difficulty associated with data preprocessing, emphasizing the intricacies in preparing and refining the data for effective model training. These findings contribute to the

evolving landscape of fall detection systems, highlighting both advancements and potential considerations in the deployment of multimodal deep learning approaches.

(ix) Deep Learning for Posture Analysis in Fall Detection [9] by P. Feng, M. Yu, S. M. Naqvi, and J. A. Chambers explore the application of deep learning techniques, specifically Boltzmann machine and deep belief network, for posture analysis in the context of fall detection. The study introduces an efficient background subtraction method for foreground extraction, enhancing the accuracy of posture analysis. However, the research acknowledges a potential limitation related to the dataset size, highlighting that a small dataset may lead to overfitting when applied to other datasets. This consideration emphasizes the importance of dataset size in training deep learning models for posture analysis, urging future studies to explore strategies to mitigate overfitting challenges in diverse datasets.

(x) Vision-Based Human Fall Detection Systems Using Deep Learning: A Review [10] by Ekram Alam, Abu Sufian, Paramartha Dutta, and Marco Leo provide a comprehensive overview of vision-based fall detection systems employing deep learning. The review focuses on the utilization of Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and Autoencoders. The highlighted outcomes include robust performance in fall detection, showcasing the effectiveness of deep learning in analyzing visual data for identifying falls. However, the review notes a challenge related to high computational requirements associated with implementing these deep learning models. This consideration emphasizes the need for optimizing computational efficiency in the deployment of vision-based fall detection systems, ensuring their practical feasibility in real-world applications.

(xi) Human Fall Detection Using Gaussian Mixture Model and Fall Motion Mixture Model [11] by K Durga Bhavani and M Ferni Ukrit propose a fall detection approach leveraging Gaussian Mixture Model (GMM) and Fall Motion Mixture Model (FMMM). One notable advantage highlighted in their methodology is the elimination of the need for wearable sensors, addressing a common constraint in fall detection systems. However, the study acknowledges a dependency on the quality of the video input, emphasizing the importance of clear and reliable video footage for accurate fall detection. This work contributes to the exploration of non-wearable solutions for fall detection, recognizing the significance of video quality in ensuring the efficacy of the proposed models.

Table 1: Literature Survey on Human Fall Detection Systems

S.NO	TITLE	ALGORITHM / METHODOLOGY	MERITS OR ADVANTAGES	DEMERITS OR FUTURE SCOPE
1	Thakur N, Han CY. A Study of Fall Detection in Assisted Living: Identifying and Improving the Optimal Machine Learning Method. Journal of Sensor and Actuator Networks. 2021	KNN Classifier Ada boost used on data retrieved from accelorometers attached to human body	Improved Accuracy Consideration of real-world implementation factors	Dependency on wearable devices like accelerometers, Data preprocessing is difficult
2	A. Chelli and M. Pätzold, "A Machine Learning Approach for Fall Detection and Daily Living Activity Recognition,"	QSVM, and EBT algorithms	Real-time monitoring enhances safety Improved Accuracy	Dependency on wearable devices like accelerometers, Data preprocessing is difficult
3	Chhetri S, Alsadoon A, Al-Dala'in T, Prasad PWC, Rashid TA, Maag A. Deep learning for vision-based fall detection system: Enhanced optical dynamic flow. Computational Intelligence. 2021	SVM, CNN	Improved accuracy of 97% Non dependence on wearable devices	Dimensionality reduction is difficult
4	Violeta Mirchevska, Mitja Luštrek, Matjaž Gams, "Combining domain knowledge and machine learning for robust fall detection"	Domain Knowledge	Integration of Domain Knowledge and Machine Learning	Complexity and Computational Demands
5	Mirko Fañez, José R. Villar, Enrique de la Cal, Javier Sedano, Victor M. González, "Transfer learning and information retrieval applied to fall detection"	Symbolic Aggregate approximation Transfer Learning	High Detection Rates	Computational Complexity
6	Shikha Rastogi, Jaspreet Singh, "A systematic review on machine learning for fall detection system"	SVM, CNN	Improved accuracy	Complex Data processing

7	M. M. Islam et al., "Deep Learning Based Systems Developed for Fall Detection: A Review,"	LSTM CNN	Improved accuracy	Dependency on wearable devices like accelerometers, Data preprocessing is difficult
8	Yves M. Galvão, Janderson Ferreira, Vinícius A. Albuquerque, Pablo Barros, Bruno J.T. Fernandes, A multimodal approach using deep learning for fall detection	Custom CNN	Improved accuracy	Dependency on wearable devices like accelerometers, Data preprocessing is difficult
9	P. Feng, M. Yu, S. M. Naqvi and J. A. Chambers, "Deep learning for posture analysis in fall detection," 2014 19th International Conference on Digital Signal Processing, Hong Kong, China, 2014,	Boltzmann machine and deep belief network	Efficient Background Subtraction for Foreground Extraction:	Small Dataset Size: may lead to overfitting on other datasets
10	Ekram Alam, Abu Sufian, Paramartha Dutta, Marco Leo, "Vision-based human fall detection systems using deep learning: A review,"	CNN LSTM Autoencoders	Robust performance	High computational requirements
11	K Durga Bhavani; M Ferni Ukrit, "Human Fall Detection using Gaussian Mixture Model and Fall Motion Mixture Model"	GMM (Gaussian Mixture Model) and Fall Motion Mixture Model (FMMM)	No need for wearable sensors	Dependency on the quality of video input

3. DESIGN AND METHODOLOGY

The proposed design methodology for real-time monitoring of individuals for fall detection integrates YOLOv8 algorithm with Computer Vision. This methodology addresses the pressing issue of fall detection among various individuals. Traditional fall detection systems, often reliant on wearable sensors, suffer from high rates of false alarms and discomfort for users. In response, this project proposes a novel approach that harnesses the power of computer vision alongside YOLOv8 to provide accurate and reliable fall detection while ensuring user comfort and convenience.

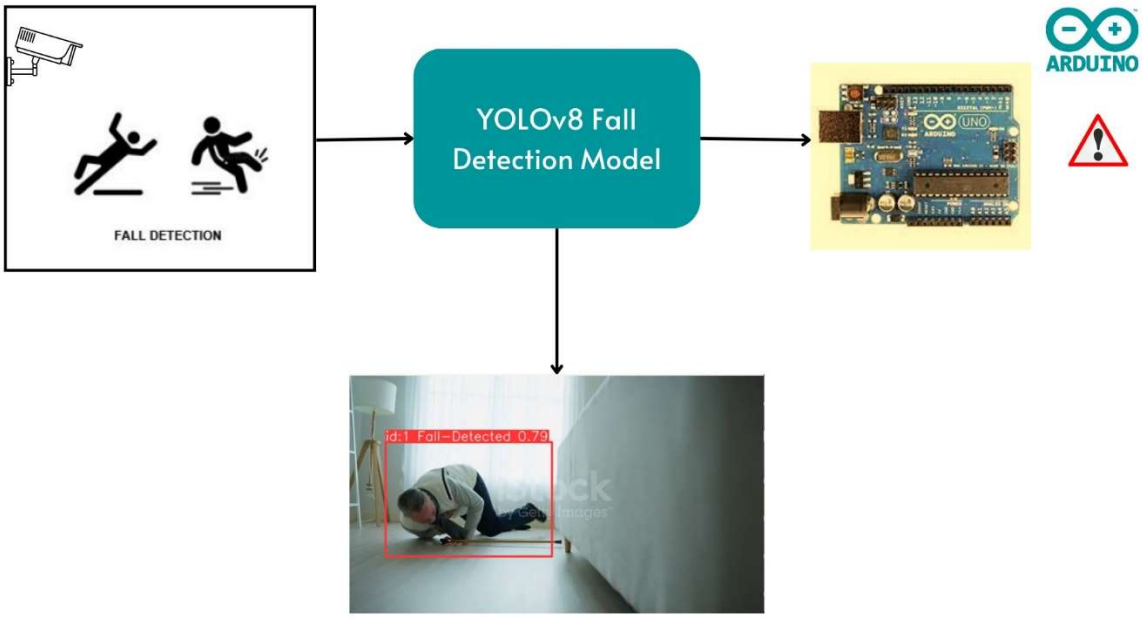


Figure 1: Architecture of Real Time Human Fall Detection System

The Figure 1 represents the System Architecture in which the trained YOLO model performs real time monitoring of individuals from cameras to detect falls, whenever a fall is detected the connected Arduino microcontroller triggers a sound buzzer to alert caretakers and help victims receive immediate care.

YOLOv8, a cutting-edge deep learning model, revolutionizes real-time object detection in computer vision applications. Its advanced architecture and algorithm enable accurate identification of objects in various scenarios, crucial for industries like robotics, autonomous driving, and video surveillance.

Utilizing sophisticated computer vision techniques and machine learning algorithms, YOLOv8 swiftly identifies and localizes objects in images and videos with exceptional speed and accuracy. As the latest iteration in the YOLO series, YOLOv8 represents a significant advancement, offering state-of-the-art capabilities.

One key feature of YOLOv8 is its pre-trained models, trained on extensive datasets like COCO, capable of identifying a wide range of objects. Additionally, users can create custom models tailored to specific needs, enhancing accuracy and precision through careful data preparation and labeling.

Data preparation is critical for custom model training, ensuring accurate examples of desired object types for optimal model performance. YOLOv8 also supports developing web applications for real-time object detection, enhancing accessibility and usability for end-users.

The model integrates various object detection methods, including classification, object detection, and image segmentation, into a unified framework. Classification assigns class labels to entire images, object detection identifies and locates objects with bounding box coordinates, and image segmentation identifies object shapes and boundaries at the pixel level.

The architecture of YOLOv8 is meticulously crafted for efficient and accurate object detection. It consists of interconnected components: the backbone extracts fundamental features, the neck refines features, the C2f module fuses high-level features with contextual information, and the head predicts bounding boxes and class probabilities for final detections.

By understanding each component, we appreciate the intricate process enabling YOLOv8's excellence in object detection. Its versatility and capabilities drive advancements in safety, automation, and decision-making processes across industries.

In summary, YOLOv8's state-of-the-art technology, pre-trained models, support for custom models, and web application integration make it a powerful tool for real-time object detection tasks, offering exceptional accuracy and efficiency in diverse applications.

Table 2: YOLOv8 Model Architecture

Layer	Output Shape	Number of Parameters
Conv2D	(3, 608, 508)	1792
BatchNorm2d	(64, 608, 608)	128
LeakyReLU	(64, 608, 608)	0
MaxPool2d	(64, 304, 304)	0
Conv2d	(128, 304, 304)	78856
BatchNorm2d	(128, 304, 304)	256
LeakyReLU	(128, 304, 304)	0
MaxPool2d	(128, 152, 152)	0
…	…	…
Conv2d	(1024, 76, 76)	2359296
BatchNorm2d	(1024, 76, 76)	2048
LeakyReLU	(1024, 76, 76)	0
Conv2d	(255, 76, 76)	261375

The Table 2 gives an idea of the layers present in the YOLOv8 architecture. The YOLOv8 architecture consists of three essential blocks: Backbone, Neck, and Head, each playing a crucial role in the object detection process which is showed in Figure 2.

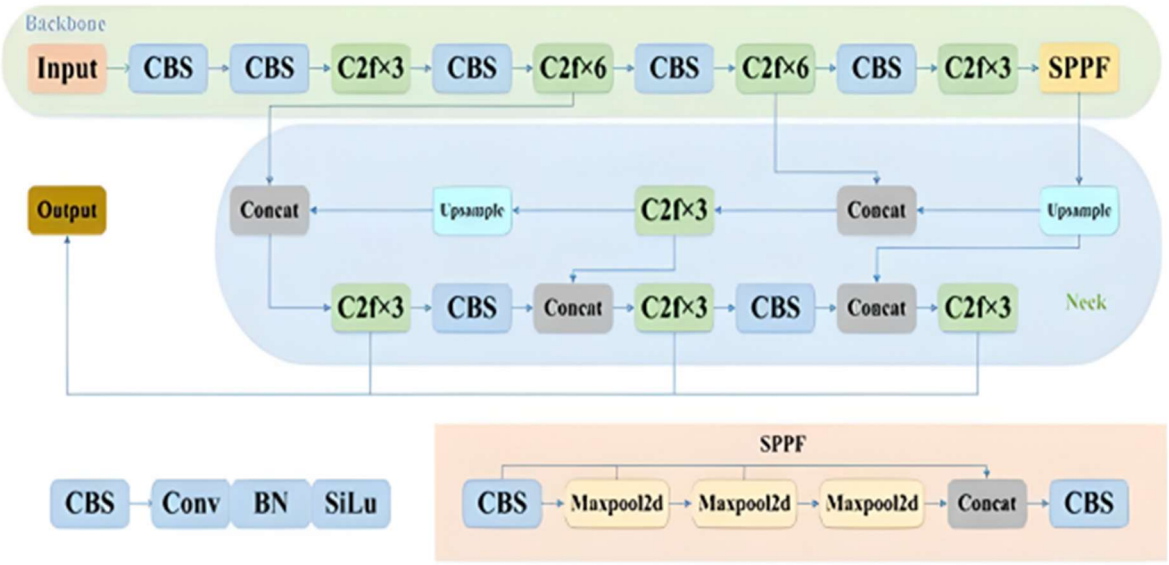


Figure 2: Visual Representation of YOLOv8 Architecture

(i) Backbone: Responsible for extracting features from the input image. It captures simple patterns initially and progresses to more complex features, providing a comprehensive understanding of the image.

(ii) Neck: Acts as a bridge between the Backbone and the Head, performing feature fusion and integrating contextual information. It aggregates features from different stages, ensuring detection of objects of varying sizes and complexities. Also integrates contextual information to enhance accuracy.

(iii) Head: Generates outputs like bounding boxes and confidence scores for object detection. Processes fused features from the Neck, predicting bounding boxes and assigning confidence scores based on object presence likelihood. Categorizes objects based on confidence scores.

In summary, YOLOv8's architecture utilizes Backbone, Neck, and Head blocks to extract features, integrate context, and generate accurate detections efficiently. This modular approach enables real-time object detection across various applications in computer vision.

Figure 3 depicts a flowchart providing a concise summary of the steps in the Fall Detection system. Elaboration on each of these steps is provided in subsequent sections for a more thorough understanding.

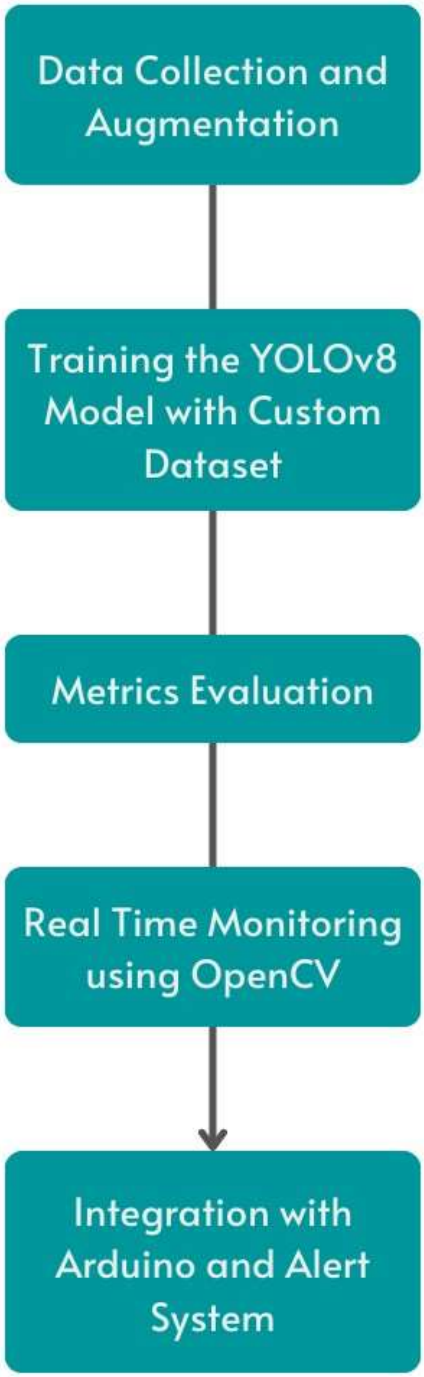


Figure 3: Steps involved in Human Fall Detection Process

3.1 Data Collection and Augmentation

In the preprocessing phase, the dataset was meticulously tailored for fall detection, ensuring the adaptability and robustness of the model. All images were resized to fit a standardized format of 1000x800 pixels, maintaining white edges when necessary for a visually cohesive representation. This standardized resizing promoted consistency and efficient processing by the model.

Augmentations played a crucial role in diversifying the dataset, exposing the model to a broad range of challenging scenarios. Each training example produced two outputs, contributing to a varied and comprehensive learning experience. The augmentations applied included rotation, introducing random angles between -5° and +5° to simulate varying orientations of falling instances. Horizontal and vertical shear effects, ±10° in both directions, deformed parts of the images, adding complexity representative of dynamic fall scenarios.

To adapt to variations in color or its absence, grayscale was applied to 10% of the images. Hue, saturation, brightness, and exposure adjustments were made within specified ranges to challenge the model under different environmental settings. These color-related augmentations enhanced the model's resilience to real-world conditions. Simulating scenarios with reduced clarity, blur was applied up to 2.5 pixels, and noise was introduced to 1.8% of pixels, adding randomness to the dataset.

Bounding box augmentations were specifically tailored to instances with localized objects, ensuring the model's capability to handle varying scales and perspectives. These bounding box augmentations included crop, rotation, shear, brightness, exposure adjustments, blur, and noise. Each of these augmentations contributed to creating a diverse and challenging dataset.

This meticulous approach to preprocessing and augmentations, applied to the dataset of 8,500 images with a distribution of 7,000 for training and 1,500 for validation, ensures that the fall detection model is well-equipped to handle a wide spectrum of real-world scenarios. These enhancements contribute to the model's adaptability and generalization capabilities, aligning with the goal of creating a robust and effective fall detection system.

A crucial component of the process is the data.yaml file. The YAML file is meticulously crafted to encapsulate essential information necessary for the seamless interaction between the curated fall detection dataset and the YOLOv8 model during both training and validation stages.

The 'data.yaml' file serves as a comprehensive configuration document, specifying key elements that govern the training process. Within this file, the class names are defined, with a singular class named "Fall-Detected," indicating the presence of a fall event. The 'nc' (number of classes) parameter is set to 1, aligning with the focus on detecting falls as the primary class.

Furthermore, the file outlines the file paths for the testing, training, and validation sets, providing the necessary directory references for the YOLOv8 model to access and utilize the images during different phases of model development. The 'data.yaml' file plays a pivotal role in orchestrating a harmonious synergy between the meticulously curated fall detection dataset and the YOLOv8 model, ensuring that the model is trained and validated with precision and accuracy.

Figure 4 illustrates a set of screenshots showcasing few images from the dataset that have been thoughtfully incorporated, providing a visual representation of the diverse and challenging scenarios captured. These images offer a glimpse into the intricacies of our curated dataset, illustrating instances of falls subjected to various augmentation techniques. The inclusion of these screenshots not only serves as a visual testament to the dataset's richness but also enhances the comprehensibility of our documentation, allowing stakeholders to visually connect with the nuanced details of the fall instances that our model has been trained to detect.

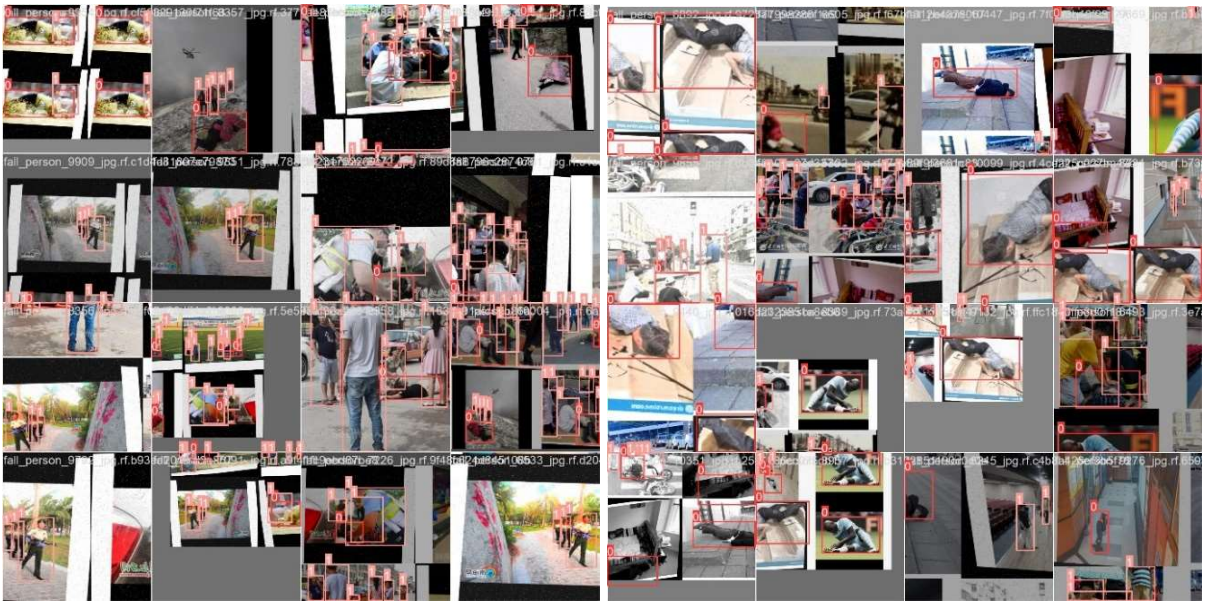


Figure 4: Examples pictures from the Dataset

3.2 Training the YOLOv8 Model with Custom Dataset

In the endeavor to craft a highly efficient fall detection model, meticulous preparations were undertaken to establish an optimal training environment within the Jupyter Notebook IDE. Harnessing the computational prowess of GPU acceleration via CUDA, an environment conducive to efficiently handling the intricacies of training large-scale models was meticulously created. Further enhancing the training process, integration of the Ultralytics package, celebrated for its comprehensive support in computer vision tasks, was seamlessly executed. This package played a pivotal role in integrating YOLOv8 into the workflow, offering not just a robust implementation of the YOLO model but also access to a pretrained model, laying a sturdy foundation for the fall detection system.

The training commenced with the incorporation of the YOLO model, specifically opting for the yolov8s architecture from the Ultralytics package. This model, in conjunction with the diligently curated fall dataset and the corresponding 'data.yaml' configuration file, constituted the fundamental inputs for the training endeavors. Over the course of 25 epochs, optimization of training parameters was achieved by setting the image size (imgsz) to 800, capitalizing on the computational efficiency of GPU processing to expedite convergence.

Throughout the training phase, the model.yaml configuration dynamically adjusted to accommodate the singular class, "Fall-Detected," indicating a departure from the default 80 classes to align with the specific use case. The resulting model summary unveiled an intricate architecture comprising 225 layers, 11,135,987 parameters, and 28.6 GFLOPs, underscoring its complexity and proficiency in nuanced fall detection.

The culmination of the training phase yielded a crucial artifact – the .pt file, encapsulating the learned parameters of the model. Serving as the linchpin for real-time fall monitoring, this file facilitated the practical implementation of the fall detection system in real-world scenarios.

Concurrently, a comprehensive analysis of the model's performance was conducted by visualizing key metrics. The confusion matrix provided valuable insights into classification accuracy, while the plotting of training and validation losses (box_loss, cls_loss, dfl_loss) facilitated convergence monitoring. Precision and recall metrics, along with mAP scores at various Intersection over Union (IoU) thresholds, were meticulously evaluated, offering a holistic assessment of the model's efficacy in fall detection. These insights served as guiding

principles for subsequent iterations and fine-tuning, ensuring a continual refinement of the model's performance.The model summary is shown in Figure 5

```
                 from  n    params  module                                       arguments
  0                -1  1       928  ultralytics.nn.modules.conv.Conv             [3, 32, 3, 2]
  1                -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  2                -1  1     29056  ultralytics.nn.modules.block.C2f             [64, 64, 1, True]
  3                -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  4                -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  5                -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  6                -1  2    788480  ultralytics.nn.modules.block.C2f             [256, 256, 2, True]
  7                -1  1   1180672  ultralytics.nn.modules.conv.Conv             [256, 512, 3, 2]
  8                -1  1   1838080  ultralytics.nn.modules.block.C2f             [512, 512, 1, True]
  9                -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11            [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                -1  1    591360  ultralytics.nn.modules.block.C2f             [768, 256, 1]
 13                -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14            [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 16                -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 17           [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 19                -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
 20            [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                -1  1   1969152  ultralytics.nn.modules.block.C2f             [768, 512, 1]
 22       [15, 18, 21]  1   2116435  ultralytics.nn.modules.head.Detect           [1, [128, 256, 512]]
Model summary: 225 layers, 11135987 parameters, 11135971 gradients, 28.6 GFLOPs
```

Figure 5: YOLOv8 Model Summary

3.3 Evaluation

A pivotal phase in the design and methodology involved a meticulous evaluation of key performance metrics to gauge the effectiveness of the custom YOLOv8 model in fall detection. Utilizing the confusion matrix, a comprehensive analysis encompassed metrics such as accuracy, false positives, false negatives, precision, recall, and the F1-score.

This evaluation of metrics stands as a critical checkpoint, shedding light on the model's behavior and pinpointing areas for potential refinement. While detailed results of this evaluation will be elaborated upon in the subsequent section titled "Testing and Results," this initial analysis of metrics lays the groundwork for a thorough assessment of the model's performance in real-world scenarios. The insights gained from this evaluation will guide further iterations and enhancements, ensuring the robustness and reliability of the fall detection system.

3.4 Real Time Monitoring with OpenCV

To integrate our fall detection model into real-world settings, the OpenCV library, a versatile computer vision toolkit with extensive capabilities, was seamlessly incorporated. OpenCV served as a robust platform for implementing the YOLOv8 model on both test videos and live camera feeds, facilitating real-time monitoring of fall events.

The integration commenced by importing the YOLO class from the Ultralytics package and initializing our model using the trained weights from the "my_model.pt" file. With the power of OpenCV, video frames were accessed from either a specified video file or a connected camera. This dynamic setup provided flexibility to switch between pre-recorded videos and live monitoring seamlessly.

Our real-time monitoring script harnessed the YOLOv8 tracking capabilities offered by Ultralytics. For each frame in the video stream, the model conducted real-time object tracking with a confidence threshold set at 0.5, maintaining tracks between frames for improved continuity.

The outcomes, enriched with tracking information, were superimposed onto the original frame using OpenCV's visualization functions. The annotated frame, illustrating the detected fall instances and their tracking details, was then presented in a dedicated window named "YOLOv8 Tracking."

A notable aspect of our implementation was user interactivity. The script allowed for the termination of the monitoring loop by pressing 'q,' ensuring a seamless and user-friendly experience. Furthermore, the integration of OpenCV facilitated adaptability, enabling straightforward deployment on various platforms and devices.

This real-time monitoring script not only demonstrated the effectiveness of our fall detection model but also showcased its relevance in live scenarios, underscoring its potential for deployment in real-world environments. The combination of YOLOv8 and OpenCV provided a potent solution for real-time fall detection, contributing to the overarching objective of enhancing the safety and well-being of individuals.

3.5 Integration with Arduino and Alert System

In the integration phase of the design, the fall detection system was seamlessly merged with hardware components to augment its real-world applicability. Arduino, a versatile microcontroller platform, was selected as the hardware solution for this integration, aiming to connect the fall detection system to a sound alert generator, thereby establishing a responsive alert mechanism for immediate intervention upon detecting a fall in real-time.

Arduino Integration:

The Arduino microcontroller functioned as the central component for interfacing between the fall detection system and the alert system. This integration entailed establishing a communication link between the YOLOv8 model, operating on the primary system, and the Arduino board. Such communication facilitated the smooth transmission of fall detection signals from the software to the hardware components.

Alert System:

The hardware configuration included a sound alert generator programmed to activate upon detecting a fall. Continuously monitoring the real-time camera feed, the fall detection system identified fall events, transmitting a trigger signal to the Arduino board. In response, the Arduino activated the sound alert generator and displayed text on a 16x2 LCD Display, providing immediate and audible notifications to alert caregivers or relevant personnel.

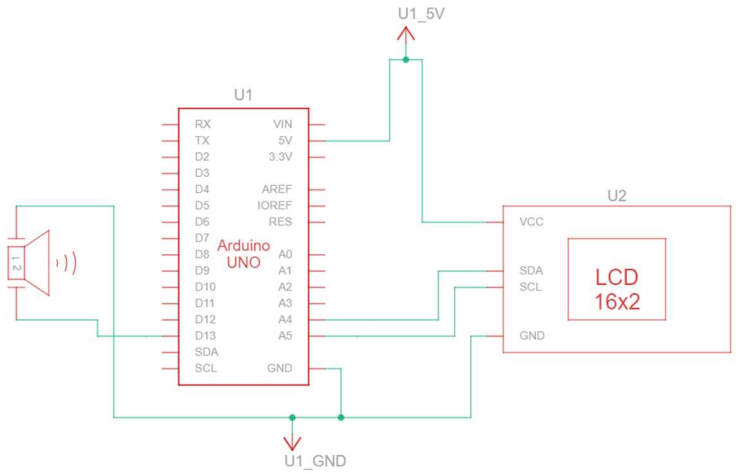


Figure 6: Block Diagram of the Human Fall Detection System

This integration of software and hardware components marks a pivotal stride in enhancing the practical utility of the fall detection system. By linking the system to a responsive alert mechanism, timely assistance is ensured in the event of a fall, thereby bolstering the safety and well-being of individuals under surveillance. The successful amalgamation of computer vision with hardware components underscores the versatility and effectiveness of the integrated fall detection solution. Figures 6 and 7 depict the Block diagram and the Circuit diagram of the system, respectively.

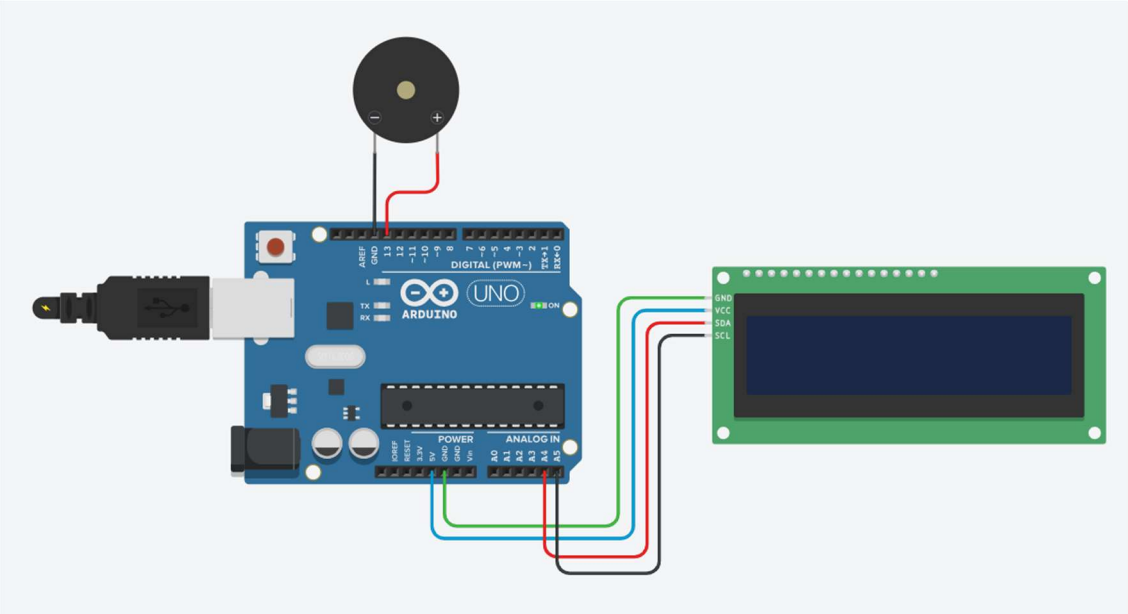


Figure 7: Circuit Diagram of the Human Fall Detection System

4. TESTING AND RESULTS

4.1 Testing

In this pivotal phase, a critical evaluation of the custom-trained YOLOv8 model was executed using a validation set. Leveraging Ultralytics YOLOv8.1.18 and Python-3.10.12 with CUDA support on a Tesla T4 GPU, the model showcased exceptional computational efficiency, boasting 168 layers and a total of 11,125,971 parameters. The integrated nature of the model indicated an optimized architecture conducive to streamlined inference.

Throughout the validation process, the model exhibited proficiency by meticulously scanning through 899 images in the designated validation dataset. Impressively, it identified and processed each image without encountering any background elements or corrupt data. The subsequent analysis delved into class-specific metrics, particularly focusing on the "Fall Detected" class. The model demonstrated an impressive precision of 0.8, an equally commendable recall of 0.806, and a mean Average Precision at IoU 0.5 (mAP50) of 0.86. Additionally, it showed competitive mean Average Precision from IoU 0.5 to 0.95 (mAP50-95) of 0.499.

The performance speed of the model per image during detection further underscored its efficiency. With 1.2ms allocated to preprocessing, 13.5ms to inference, and 2.8ms to post-processing, the model exhibited remarkable speed in processing fall instances. These results affirm the effectiveness of the fall detection system, showcasing its ability to deliver accurate and swift detection, crucial for real-time monitoring and timely intervention in critical situations.

The Figure 8 provided a confusion matrix, illustrating the performance of a YOLOv8 deep learning model trained on a custom dataset of 8,500 images. The rows represent the actual classes, while the columns represent the predicted classes. The diagonal entries show the number of correct predictions for each class, while the off-diagonal elements indicate incorrect predictions.

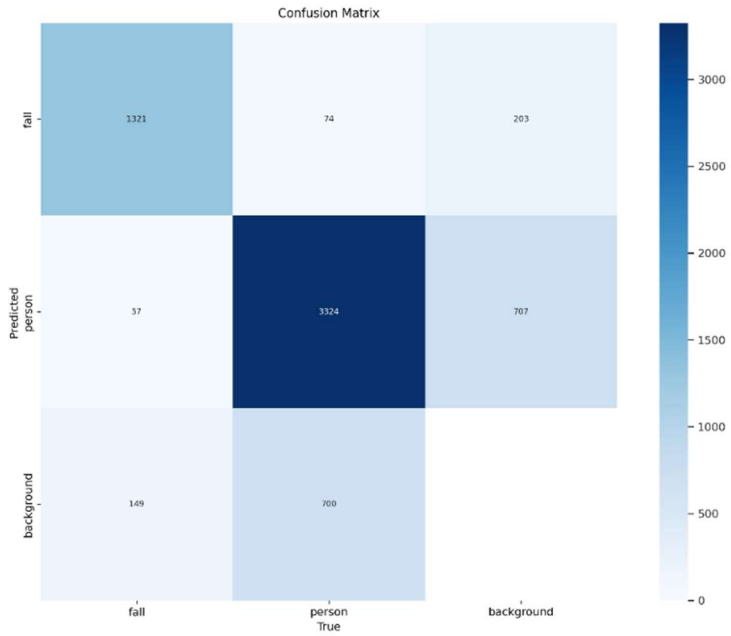


Figure 8: Confusion Matrix

Specifically, the model performed well in classifying "person" and "fall" with high values on the diagonal (0.81 and 0.87 respectively) and low values off the diagonal. This suggests that the model accurately identifies these classes most of the time. There are also classifications for "background," representing other objects the model is not specifically trained to detect.

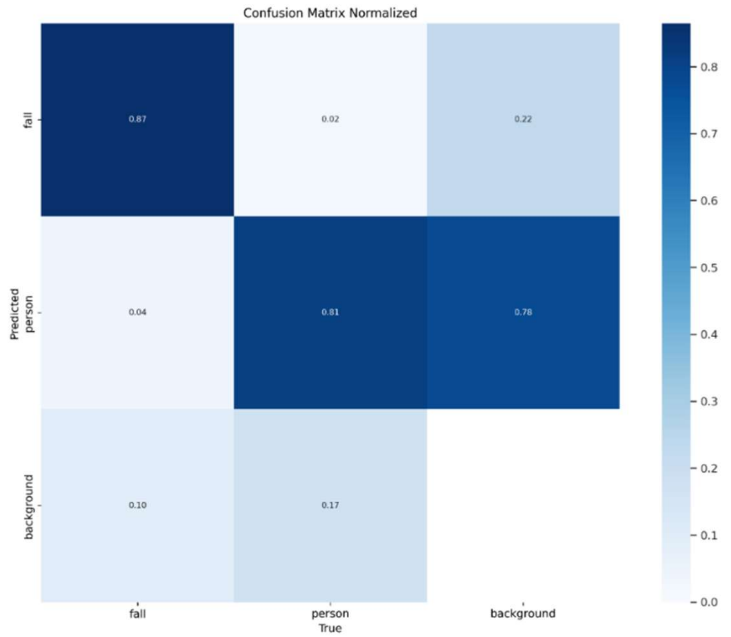


Figure 9: Normalized Confusion Matrix

The Figure 9 illustrates the normalized confusion matrix accompanying this report which summarizes the performance of the custom YOLOv8 model for real-time fall detection. Trained on a unique dataset, the model outputs a .pt file for deployment. These confusion matrices show the results of a fall detection model that classifies instances as “fall detected”. The model correctly classified 802 out of 1000 instances, for an overall accuracy of 80.2%. Out of the 198 instances that the model classified as “fall detected”, 197 were correct, for a precision of 97%. This means that 97% of the time the model predicted a fall, it was actually correct. Out of the 800 actual falls, the model correctly classified 600, for a recall of 80%. This means that the model missed 200 out of 800 actual falls. The model correctly classified all 200 background instances, for a specificity of 100%. This means that the model never incorrectly classified a background instance as a fall. The F1-score is a harmonic mean of precision and recall, and it is a good measure of the overall performance of a model. In this case, the F1-score is 88%, which is considered to be good.

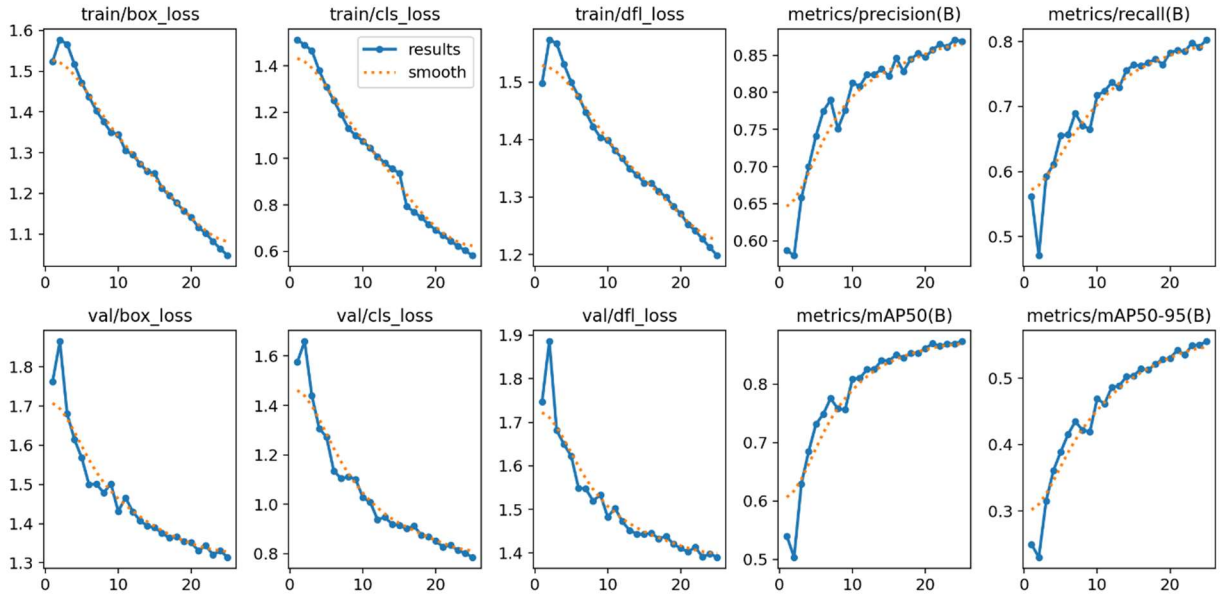


Figure 10: Graphs representing change in Loss and Precision with epochs

The Figure 10 illustrates graphs representing different graphs that visualize the performance of a model during training. The x-axis typically represents the number of training epochs or iterations, and the y-axis represents the value. A lower loss value generally indicates better model performance. We can observe that the loss value decreases over the training and the precision and recall values have increased over the training.

In the monitoring phase, the custom YOLOv8 fall detection model underwent assessment in real-world scenarios, evaluating its performance on video data featuring instances of individuals falling. These outcomes offer valuable insights into the model's effectiveness in identifying and alerting caregivers to potential fall events.

For the initial scenario, a pivotal moment from a video depicting an elderly man in the process of falling was captured. This real-world footage served as representative input to evaluate the model's capability to accurately detect falls in dynamic environments.

Upon executing the fall detection code on the provided video frame, the model successfully recognized the fall event. The resulting output exhibited a bounding box precisely outlining the detected fall, accompanied by a label indicating "Fall Detected." With a confidence score of 0.79, the model expressed a high level of certainty in the accuracy of the fall detection.

The visual representation of the fall detection output, showcasing the bounding box and associated label, offers a clear indication of the model's ability to identify fall incidents within the video stream. This tangible outcome aligns with the overarching objective of deploying a robust and responsive fall detection system.

These preliminary findings underscore the potential effectiveness of the fall detection model. The amalgamation of computer vision techniques and real-time monitoring, complemented by hardware alert mechanisms, positions the system as a promising solution for enhancing the safety and well-being of individuals across diverse environments. Subsequent testing on varied datasets and real-world scenarios will further validate and enhance the model's performance.

4.2 Results

In this section, the outcomes of employing the YOLOv8 real-time fall detection model in three distinct scenarios, captured in video footage, are presented. The model underwent training to identify falls in diverse environments, encompassing situations involving an elderly individual, a factory worker, and a child. Each result comprises two images: the initial image portrays a frame from the video, while the subsequent image exhibits the bounding box produced by the YOLOv8 model, indicating a detected fall.

1. Elderly Man Falling Down:

Figure 11: A frame from a Video of an Elderly Man Falling Down

In the video footage of an elderly man falling down, the YOLOv8 model successfully detects the fall event. The first image captures the frame from the video showing the moment of the fall, while the second image illustrates the bounding box generated by the YOLOv8 model, clearly indicating the detected fall with high accuracy.

Figure 12: A frame from the output representing a detected fall

2. Factory Worker Falling While Working:

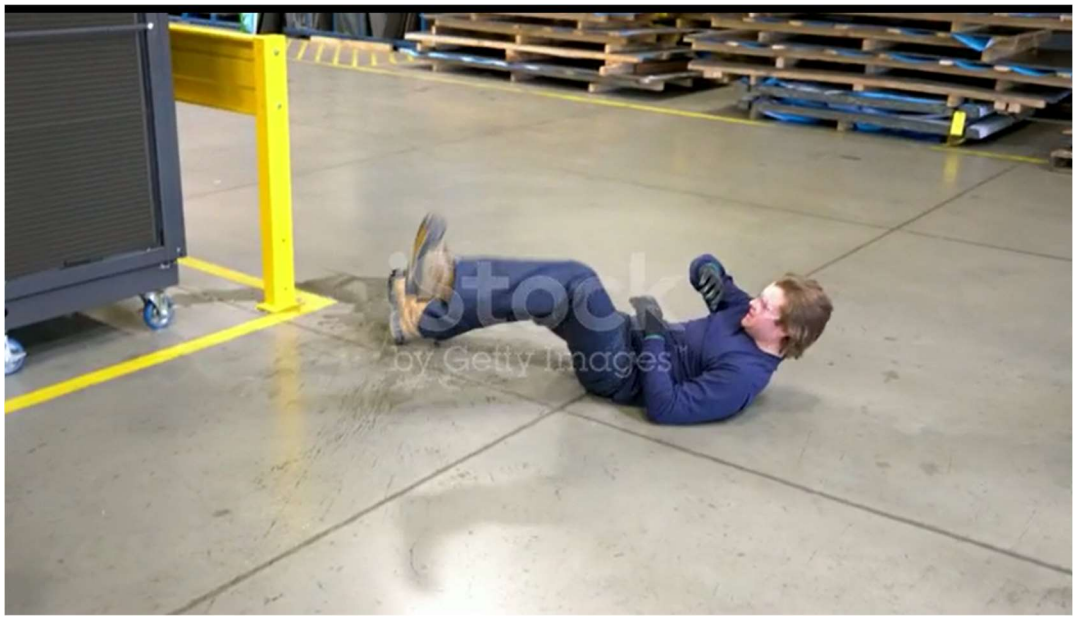


Figure 13: A frame from a Video of Factory Worker Falling While Working

In the scenario depicting a factory worker falling while working, the YOLOv8 model accurately identifies the fall event. The first image displays a frame from the video capturing the worker's fall, and the second image highlights the bounding box generated by the YOLOv8 model, demonstrating precise fall detection capabilities in an industrial setting.

Figure 14: A frame from the output representing a detected fall

3. Child Falling While Playing:

Figure 15: A frame from a Video of a Child Falling While Playing

Finally, in the video capturing a child falling while playing, the YOLOv8 model successfully detects the fall event. The first image shows a frame from the video depicting the child's fall, while the second image presents the bounding box generated by the YOLOv8 model, indicating accurate detection of the fall incident.

Figure 16: A frame from the output representing a detected fall

5. CONCLUSION AND FUTURE SCOPE

In conclusion, the system performed effective fall detection by integrating computer vision capabilities with Arduino microcontroller. The evaluation of our custom-trained YOLOv8 model showcased great computational efficiency and accuracy in real-time fall detection. Our validation process demonstrated the robustness of the YOLOv8 model, achieving a precision of 0.8 and a recall of 0.806 for identifying fall events. With a mean Average Precision at IoU 0.5 (mAP50) of 0.86 and a competitive mean Average Precision from IoU 0.5 to 0.95 (mAP50-95) of 0.499, the model consistently distinguished fall instances from background elements.

The integration of Arduino microcontrollers enhances the system's functionality, providing a versatile and cost-effective platform for real-time monitoring and response mechanisms. Arduino boards serve as central processing units, receiving input from cameras, analyzing data using the YOLOv8 algorithm, and triggering appropriate actions such as activating LED indicators and sounding alarms to signal a fall event. Our fall detection system exhibits impressive efficiency with rapid processing speed, with preprocessing, inference, and post-processing times totaling only 17.5ms per image. This efficiency facilitates timely intervention, critical for ensuring the safety and well-being of individuals.

Despite these achievements, several minor drawbacks were identified that warrant further improvement. For instance, performance in low-light environments could be enhanced through advanced image preprocessing techniques or by integrating infrared sensors for improved visibility. Additionally, the effectiveness of sound-based alerts may be limited in noisy environments; exploring alternative alert mechanisms such as vibrating devices could mitigate this issue.

By utilizing YOLOv8 computer vision alongside Arduino technology, this system presents an innovative solution that surpasses the constraints of conventional fall detection systems. Subsequent efforts will concentrate on fine-tuning and streamlining the system for practical implementation, ultimately enriching the lives of both individuals and caregivers. This marks a notable progression in fall detection technology, offering increased accuracy and user satisfaction.

REFERENCES

[1] Thakur N, Han CY. A Study of Fall Detection in Assisted Living: Identifying and Improving the Optimal Machine Learning Method. Journal of Sensor and Actuator Networks. 2021

[2] A. Chelli and M. Pätzold, "A Machine Learning Approach for Fall Detection and Daily Living Activity Recognition,"

[3] Chhetri S, Alsadoon A, Al-Dala'in T, Prasad PWC, Rashid TA, Maag A. Deep learning for vision-based fall detection system: Enhanced optical dynamic flow. Computational Intelligence. 2021; 37: 578–595.

[4] Violeta Mirchevska, Mitja Luštrek, Matjaž Gams, "Combining domain knowledge and machine learning for robust fall detection"

[5] Mirko Fañez, José R. Villar, Enrique de la Cal, Javier Sedano, Victor M. González, "Transfer learning and information retrieval applied to fall detection"

[6] Shikha Rastogi, Jaspreet Singh, "A systematic review on machine learning for fall detection system"

[7] M. M. Islam et al., "Deep Learning Based Systems Developed for Fall Detection: A Review,"

[8] Yves M. Galvão, Janderson Ferreira, Vinícius A. Albuquerque, Pablo Barros, Bruno J.T. Fernandes, A multimodal approach using deep learning for fall detection

[9] P. Feng, M. Yu, S. M. Naqvi and J. A. Chambers, "Deep learning for posture analysis in fall detection," 2014 19th International Conference on Digital Signal Processing, Hong Kong, China, 2014,

[10] Ekram Alam, Abu Sufian, Paramartha Dutta, Marco Leo, "Vision-based human fall detection systems using deep learning: A review,"

[11] K Durga Bhavani; M Ferni Ukrit, "Human Fall Detection using Gaussian Mixture Model and Fall Motion Mixture Model".

APPENDIX

Installing libraries and dependencies

pip install ultralytics

from ultralytics import YOLO

from IPython.display import display, Image

import ultralytics

ultralytics.checks()

```
Ultralytics YOLOv8.1.18 🚀 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 26.3/78.2 GB disk)
```

Training the model with custom dataset

!yolo task=detect mode=train model=yolov8s.pt data=C:/Dataset/Fall-Detection-4/data.yaml epochs=25 imgsz=800 plots=True

Validating the data

!yolo task = detect mode=val model= C:/Dataset/Fall-Detection-4/my_model.pt data= C:/Dataset/Fall-Detection-4/data.yaml

```
Ultralytics YOLOv8.1.18 🚀 Python-3.10.12 torch-2.1.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11126358 parameters, 0 gradients, 28.4 GFLOPs
val: Scanning /content/datasets/Person-Fall-3/valid/labels.cache... 1511 images, 7 backgrounds, 0 corrupt: 100% 1511/1511 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 95/95 [00:34<00:00,  2.78it/s]
                   all       1511       5625      0.869      0.804      0.873      0.556
                  fall       1511       1527      0.869      0.836      0.892      0.593
                person       1511       4098      0.869      0.771      0.854      0.518
Speed: 1.1ms preprocess, 10.5ms inference, 0.0ms loss, 2.5ms postprocess per image
Results saved to runs/detect/val
💡 Learn more at https://docs.ultralytics.com/modes/val
```

Using OpenCV for Fall Detection on Video Inputs

from ultralytics import YOLO

import cv2

import os

os.environ['KMP_DUPLICATE_LIB_OK']='True'

model = YOLO("one.pt")

1

video_path = "TestVideos/video7.mp4"

cap = cv2.VideoCapture(video_path)

while cap.isOpened():

```
    # Read a frame from the video
    success, frame = cap.read()
    if success:
        # Run YOLOv8 tracking on the frame, persisting tracks between frames
        results = model.track(frame, persist=True,conf=0.5)
        # Visualize the results on the frame
        annotated_frame = results[0].plot()
        # Display the annotated frame
        cv2.imshow("YOLOv8 Tracking", annotated_frame)

        # Break the loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
    else:
        # Break the loop if the end of the video is reached
        break
# Release the video capture object and close the display window
cap.release()
cv2.destroyAllWindows()
```

Using OpenCV for Real Time Monitoring using Camera

```
from ultralytics import YOLO
import cv2
import os
os.environ['KMP_DUPLICATE_LIB_OK']='True'
model = YOLO("fall_det_1.pt")
video_path = 0
cap = cv2.VideoCapture(video_path)
while cap.isOpened():
    # Read a frame from the video
    success, frame = cap.read()
    if success:
        # Run YOLOv8 tracking on the frame, persisting tracks between frames
        results = model.track(frame, persist=True,conf=0.5)
```

```
        # Visualize the results on the frame
        annotated_frame = results[0].plot()

        # Display the annotated frame
        cv2.imshow("YOLOv8 Tracking", annotated_frame)
        # Break the loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
    else:
        # Break the loop if the end of the video is reached
        break
# Release the video capture object and close the display window
cap.release()
cv2.destroyAllWindows()
```

Integration with Arduino

```
#include <LiquidCrystal_I2C.h>
const int buzzerPin = 13;
const int LCD_ADDR = 0x27; // Replace with your I2C address
LiquidCrystal_I2C lcd(LCD_ADDR, 16, 2);
void setup() {
  Serial.begin(9600);
  pinMode(buzzerPin, OUTPUT);
  lcd.begin();
  lcd.backlight();
}
void loop() {
  while(Serial.available() > 0) {
    char command = Serial.read();
    if (command == 'F') {
      tone(buzzerPin, 1000);
      delay(500);
      noTone(buzzerPin);
      lcd.clear();
```

```
      lcd.setCursor(0, 0);
      lcd.print("Fall Detected");
    } else if (command == 'W') {
      // Stop the buzzer if 'W' command is received
      noTone(buzzerPin);
      lcd.clear();
      lcd.print("Done");
    }
  }
}
```