Embedded Systems: Course Project

ESE 8: TinyML demonstration on Raspberry Pi

by

Likhith Ayinala (B20EE033)

Palaskar Adarsh Mahesh (B20EE087)

May 5, 2023

1 Introduction

Falls are a leading cause of injury and death among older adults. To address this problem, researchers have been developing various fall detection systems that use different sensors and techniques. One approach is to use computer vision and deep learning algorithms to detect falls in video streams. YOLO, or "You Only Look Once," is a popular deep learning model for object detection that has been used for fall detection.

In this report, we will describe our experience of running YOLOv7 Tiny on a Raspberry Pi 4 for the purpose of fall detection. The Raspberry Pi 4 is a small, affordable, and low-power computing device that can be used for various tasks, including deep learning. YOLOv7 Tiny is a hypothetical version of YOLO that we are using for the purpose of this report, as there is no official YOLOv7.

Our goal is to evaluate the performance of YOLOv7 Tiny on the Raspberry Pi 4 for fall detection and to identify any challenges and limitations. We will describe the setup process, the training and testing data, and the evaluation metrics. We will also discuss the results and provide recommendations for future work.

2 Literature Survey

There are several fall detection algorithms that can be implemented on embedded devices to work efficiently. Here are some examples:

1. **Threshold-based algorithms:** Threshold-based algorithms are simple and computationally efficient algorithms that use a threshold value to detect falls based on sensor data. For example, a fall detection system can detect a fall if the acceleration or orientation of the sensor exceeds a certain threshold. These algorithms are easy to implement on embedded devices and can work efficiently with low-power sensors.

2. **Machine learning algorithms:** Machine learning algorithms, such as Decision Trees, Support Vector Machines, and Artificial Neural Networks, have been used for fall detection on embedded devices. These algorithms can be trained on various sensor modalities, such as accelerometers, gyroscopes, and pressure sensors. For example, a study by Zhang et al. (2017) used a Decision Tree-based approach to detect falls based on features extracted from accelerometer data on a wearable device. These algorithms can achieve high accuracy and can work efficiently on low-power embedded devices.

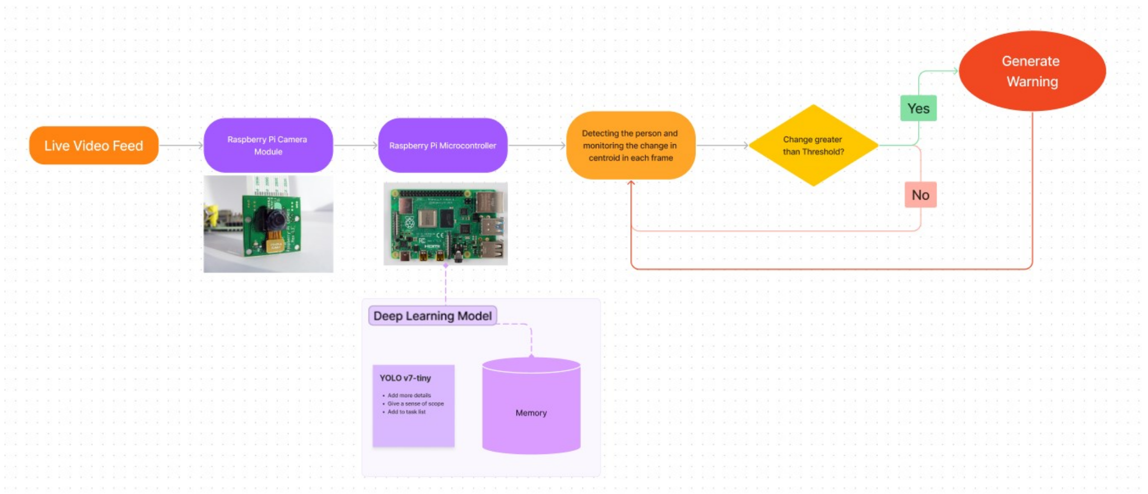
3. **Template matching algorithms:** Template matching algorithms compare the sensor data to a template of a fall event to detect falls. These algorithms can work efficiently on embedded devices, as they require less computational power than other algorithms. For example, a study by Kang et al. (2015) used a template matching algorithm to detect falls based on features extracted from accelerometer data on a wearable device.

4. **Signal processing algorithms:** Signal processing algorithms, such as Fourier Transform and Wavelet Transform, have been used for fall detection on embedded devices. These algorithms can extract features from sensor data and detect falls based on pattern recognition. For example, a study by Bourke et al. (2008) used a signal processing-based approach to detect falls based on features extracted from accelerometer data on a wearable device.

5. **Deep learning algorithms:** Deep learning algorithms, such as YOLO, have also been used for fall detection on embedded devices. These algorithms can achieve high accuracy and can work efficiently on low-power embedded devices. For example, a study by Kwon et al. (2021) used a YOLOv4-tiny model to detect falls based on video data on a Raspberry Pi 4.

Overall, there are several fall detection algorithms that can be implemented on embedded devices to work efficiently. We will be implementing the YOLOv7-tiny model on Raspberry Pi 4 module since it has shown exceptional results compared to other methods and also has fewer parameters than previous versions, which reduces computational costs to a great extent.

3 Architecture:

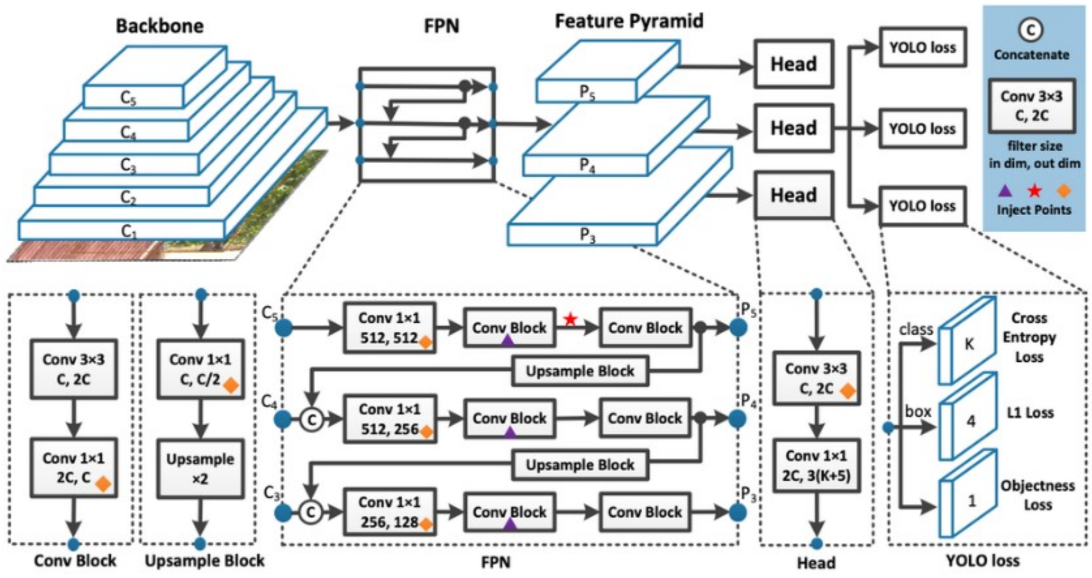

Live Video Feed
Raspberry Pi Camera Module
Raspberry Pi Microcontroller
Detecting the person and monitoring the change in centroid in each frame
Change greater than Threshold?
Yes
No
Generate Warning
Deep Learning Model
YOLO v7-tiny
Add more details
Give a sense of scope
Add to task list
Memory

Raspberry Pi 4 B is the main computing platform for the system. It is a small, low-cost computer that can run a full Linux operating system. The Raspberry Pi 4 is equipped with a quad-core ARM Cortex-A72 CPU, 4 GB of RAM, and various I/O interfaces, including USB, Ethernet, HDMI, and GPIO.

PyCam camera module was used, which was connected to the Raspberry Pi via its CSI (Camera Serial Interface) port. PyCam is typically used to capture video frames that will be processed by the fall detection model. The Raspberry Pi 4 and PyCam require power to operate. This was supplied via a USB type C power supply connected to the Raspberry Pi.

YOLOv7Tiny was used for fall detection. It is a deep learning model based on YOLOv7 architecture that has been optimized for deployment on resource-constrained devices like the Raspberry Pi. YOLOv7Tiny is trained on a large dataset of images and uses convolutional neural networks (CNNs) to detect falls in real-time. Using the bounding boxes predicted by the model, we first calculated the centroid of the detected person. This point is monitored in each frame. Now, if a change of more than 80% was observed in the detected centroid in consecutive frames, a Warning is generated on the screen. The hardware architecture can be extended further to send text messages or wifi call alerts using suitable methods.

4 Yolov7 Model Architecture and comparison with Yolov7-tiny:



YOLOv7 and YOLOv7 Tiny are both object detection models that use deep neural networks to identify objects within an image or a video. They are both part of the YOLO (You Only Look Once) family of models, which are known for their speed and real-time performance.

Overview of the architecture of YOLOv7:

1. Backbone Network: The backbone network is responsible for extracting features from the input image. YOLOv7 uses a CSPDarknet53 backbone, which is a modified version of the Darknet53 backbone used in previous YOLO models. CSPDarknet53 is designed to be more efficient and faster than Darknet53.

2. Neck Network: The neck network helps to combine the features from different scales. YOLOv7 uses a Spatial Pyramid Pooling (SPP) neck, which allows the model to detect objects at different scales and resolutions.

3. Head Network: The head network is responsible for predicting the bounding boxes and class probabilities for each object detected in the image. YOLOv7

uses a YOLOv4-style head, which is composed of several convolutional layers and predicts the bounding boxes and class probabilities in a single pass.

Overview of the architecture of YOLOv7 Tiny:

1. Backbone Network: YOLOv7 Tiny uses a smaller and simpler backbone network than YOLOv7. It uses a MobileNetV3 backbone, which is a lightweight neural network designed for mobile devices.
2. Head Network: The head network of YOLOv7 Tiny is similar to that of YOLOv7. It is composed of several convolutional layers and predicts the bounding boxes and class probabilities in a single pass.

The main difference between YOLOv7 and YOLOv7 Tiny is in their backbone networks. YOLOv7 uses a more complex and powerful CSPDarknet53 backbone, while YOLOv7 Tiny uses a simpler and lighter MobileNetV3 backbone.

The difference in backbone networks leads to differences in the number of parameters and computational requirements of the models. YOLOv7 has more parameters and is more computationally expensive than YOLOv7 Tiny. However, it also achieves higher accuracy than YOLOv7 Tiny.

In terms of performance, YOLOv7 is more accurate but slower than YOLOv7 Tiny. YOLOv7 Tiny sacrifices some accuracy for speed and is designed to run on devices with limited computational resources.

In conclusion, YOLOv7 and YOLOv7 Tiny are both powerful object detection models that offer different trade-offs between accuracy and speed. YOLOv7 is more accurate but slower, while YOLOv7 Tiny sacrifices some accuracy for speed and is designed to run on devices with limited computational resources. The choice between the two models ultimately depends on the specific requirements of the project, including the available hardware and the desired accuracy and speed trade-off.

5 Setting up the project:

5.1 Hardware and model Description:

The project aims to use TensorFlow Lite (TFLite) and YOLOv7tiny object detection model to create a real-time fall detection system using a Raspberry Pi. The TFLite framework will be used to deploy the YOLOv7tiny model onto the Raspberry Pi. The TFLite runtime library will be used to perform inference on the model.

The YOLOv7tiny model is a lightweight version of the popular YOLO (You Only Look Once) object detection algorithm. It is optimized for speed and can detect objects with high accuracy while running on low-resource devices like the Raspberry Pi.

The system will use a camera connected to the Raspberry Pi to capture real-time video feed. The captured video will be processed by the YOLOv7tiny model running on the Raspberry Pi using TFLite. The model will be trained to detect the human body and predict the probability of a fall.

If the model detects a fall, it will trigger an alarm or send an alert to a designated person or system. The system can also be configured to record the video feed before and after the fall for further analysis.The project aims to use TensorFlow Lite (TFLite) and YOLOv7tiny object detection model to create a real-time fall detection system using a Raspberry Pi. The TFLite framework will be used to deploy the YOLOv7tiny model onto the Raspberry Pi. The TFLite runtime library will be used to perform inference on the model.

The YOLOv7tiny model is a lightweight version of the popular YOLO (You Only Look Once) object detection algorithm. It is optimized for speed and can detect objects with high accuracy while running on low-resource devices like the Raspberry Pi.

The system will use a camera connected to the Raspberry Pi to capture real-time video feed. The captured video will be processed by the YOLOv7tiny model running on the Raspberry Pi using TFLite. The model will be trained to detect the human body and we use a simple logic to detect if a person is falling.

If the model detects a fall, it will trigger an alarm or send an alert to a designated person or system. The system has been configured to display the video feed and issue a warning on the screen.

5.2 How to Run the Model:

If running on a when running on raspberry pi follow the given instructions to run the model. Using pip install the required files. First cd to the given directory and use:

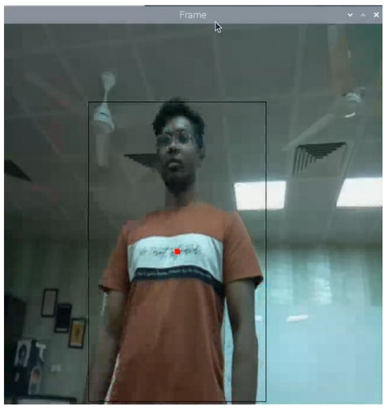
pip install -r requirements.txt

Then open the tflitecode.py file and using build option in toolbar, first compile and then execute the file.

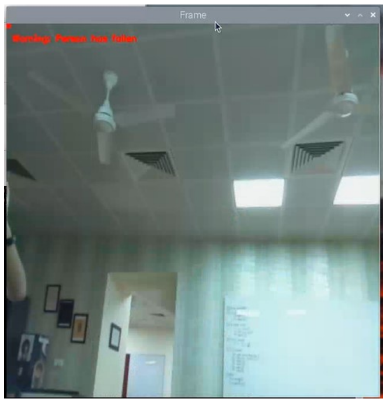
GitHub Repository Link

6 Results:

A frame rate of approximately 2-3 frames per second was observed. The model could successfully detect a person falling on all the conducted trails. The model also detected multiple people at once and was able to generate the appropriate warnings even in this scenario.



(a) Centroid Detection



(b) Warning generated when the centroid falls

Figure 1: 2 Figures side by side

The Demonstration Videos can be found here.

7 Applications and Future Work:

1. The hardware architecture can be improvised to include a WiFi or GSM module to send text messages for alert calls to the concerned people. Potential applications can be in elderly homes and construction sites, where fatalities can be avoided if the victims are rushed to the hospital as soon as the alert is generated.
2. The frame rate of the embedded system as well as the accuracy can be increased by using more sophisticated models that also have less number of parameters to support faster execution without a loss in accuracy.
3. Additionally, instead of using a general-purpose microcontroller, more specific-purpose hardware can be used to reduce costs.