

HUMAN ACTIVITY RECOGNITION ON EDGE DEVICE

Sanjana Hubballi (01fe21bcs022), Rakshita Wagh (01fe21bcs280),
Suman Patil (01fe21bcs281) , Srajana Naik (01fe21bcs364)

Under the guidance of
Dr. Padmashree Desai

KLE Technological University, Vidyanagar, Hubballi-580031, Karnataka, India

June 27, 2024



- Introduction
- Motivation
- Literature Survey
- Problem Statement and Objectives
- Approach / Solution
- System Design
- Implementation
- Intermediate Results
- Module Testing
- References

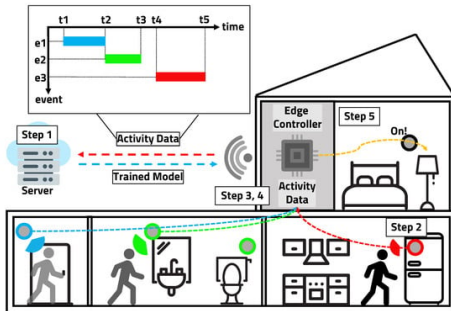
- Activity recognition is a versatile technology with applications in various fields, such as healthcare, activity monitoring and more.
- As people get aged, their physical abilities change, leading to an increased risk of accidents, falls and the need for ongoing care.
- Our proposed work focuses on the development and deployment of algorithms that can accurately monitor daily activities of older individuals.
- Accelerometer data of senior citizens used to recognize activities such as walking, standing, shuffling, ascend and descent stairs, lying and sitting.

- Need of efficient Human Activity Recognition systems such as:
 - Improve the quality of life for humans.
 - Reduce their burden and provide a foundation for healthcare professionals.
- Accelerometer data of human activities enhances the classification perception.
- Deploying activity recognition model on edge device ensures real-time health monitoring which can improve the quality of life.
- Using low-power device motivates the need for energy efficiency, portability, cost considerations.

Literature Survey

Low-Power On-Chip Implementation of Enhanced SVM Algorithm for Sensors Fusion-Based Activity Classification in Lightweight Edge Devices : (Electronics 2022) [?]

This paper proposed a low-power, memory-efficient, high-speed ML algorithm for smart home activity data classification suitable for resource-constrained environment.



Edge AI Framework for Healthcare Applications : (IJCAI 2021) [?]

The paper presents Kestrel, a patient monitoring application for healthcare facilities like hospitals, deployed on Nvidia Jetson NX. Kestrel employs body pose detection to track patient movements.

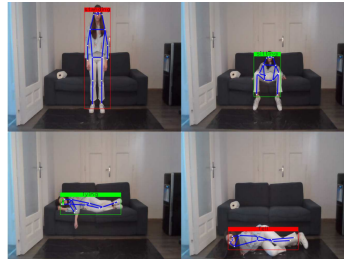
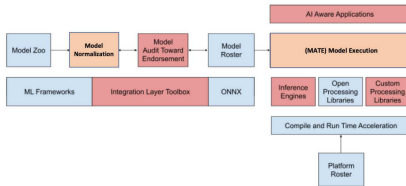


Figure: Body pose detection

Literature Survey

Edge2Train: A Framework to Train Machine Learning Models(SVMs) on Resource-Constrained IoT Edge Devices : (International Conference on the Internet of Things 2020) [?]

This paper introduces an Edge2Train framework for offline machine learning model training on IoT device microcontrollers, enhancing adaptability without cloud connectivity.

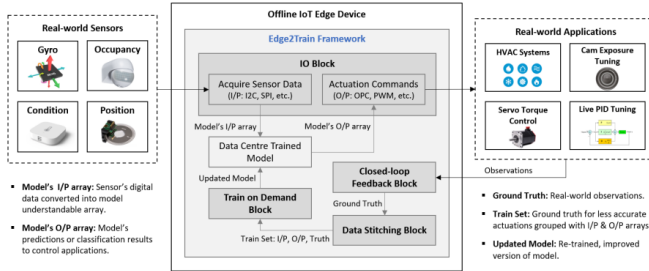


Figure: Architecture and components of Edge2Train

AHAR: Adaptive CNN for Energy-efficient Human Activity Recognition in Low-power Edge Devices:(International Conference on the Internet of Things 2021) [?]

This paper proposes an Adaptive CNN for energy-efficient Human Activity Recognition (AHAR) suitable for low-power edge devices.

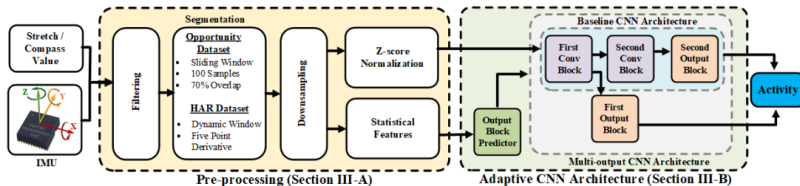


Figure: Overview of proposed AHAR methodology

A Survey on Optimization Techniques for Edge Artificial Intelligence (AI)

This paper discusses challenges associated with edge AI implementation, including cost, security, privacy, flexibility, and data integration.

The optimization techniques discussed in the paper include:

Pruning - Involves removing unimportant connections, neurons, or weights, leading to a smaller and more efficient model.

Quantization - Reduces the precision of weights and activations, typically from 32-bit to lower-bit representations like 8-bit integers.

Knowledge Distillation - Involves transferring knowledge from a larger, complex model to a smaller, simpler model.

Weight sharing - Reduce the number of parameters in a neural network by having multiple connections share the same weight value.

Hyperparameter Optimization - Involves techniques such as Grid Search, Random Search, and Bayesian Optimization to find the best set of hyperparameters for the model.

Optimization Techniques and Evaluation for Building an Integrated Lightweight Platform for AI and Data Collection Systems on Low-Power Edge Devices

The paper discusses the development and implementation of an integrated lightweight platform for AI and data collection systems on low-power edge devices.

Model used: Integration-learning AI modeling which involves combining multiple machine learning models to create a more robust and efficient system suitable for low-power edge devices.

The process used to assess the performance of optimization techniques are:

Offline Learning Evaluation - Trains the AI model using historical data when the system is idle. This process aids the AI in learning from past experiences to better handle future tasks. Assessing the model's training phase helps measure its real-world effectiveness.

Online Learning Evaluation - Tests how well the AI model adapts and learns from new data as it arrives in real-time. It checks how quickly and accurately the model can update itself based on the latest information it receives.

Problem Statement and Objectives

Problem Statement

Development and Deployment of an Optimized Human Activity Recognition Model on Low Power Edge Device (Adafruit feather m0 Adalogger).

Objectives

- Develop and optimize a human activity recognition model for efficient performance on low power edge devices.
- Deploy the optimized model on the Adafruit Feather M0 Adalogger.
- Interface the Adafruit Feather M0 Adalogger with accelerometer sensor(MPU6050) to collect real time data.

Dataset Description

The dataset consists of 15 older-adults wearing 2 accelerometers for 40 minutes on their right thigh and back.

Attributes:

thigh_x: acceleration in x-direction (down) in the unit g.

thigh_y: acceleration in y-direction (right) in the unit g.

thigh_z: acceleration in z-direction (backward) in the unit g.

back_x: acceleration in x-direction (down) in the unit g.

back_y: acceleration in y-direction (left) in the unit g.

back_z: acceleration in z-direction (forward) in the unit g.

label: annotated activity code.

Dataset Description

The dataset contains the following annotated activities with the corresponding coding scheme.

- 1:walking
- 2:shuffling
- 3:stairs(ascending)
- 4:stairs(descending)
- 5:standing
- 6:sitting
- 7:lying

timestamp	back_x	back_y	back_z	thigh_x	thigh_y	thigh_z	label
42:03.8	-0.99902	-0.06348	0.140625	-0.98047	-0.11206	-0.0481	6
42:03.9	-0.98023	-0.07935	0.140625	-0.96118	-0.12158	-0.05176	6
42:03.9	-0.9502	-0.07642	0.140625	-0.94946	-0.08057	-0.06714	6
42:03.9	-0.95483	-0.05908	0.140381	-0.95752	-0.04614	-0.05078	6
42:03.9	-0.97241	-0.04297	0.142822	-0.97705	-0.02368	-0.02661	6
42:03.9	-0.98877	-0.02612	0.157227	-0.98486	-0.04273	-0.03272	6
42:04.0	-1.00195	-0.01611	0.162109	-0.99292	-0.07544	-0.02417	6
42:04.0	-1.00049	-0.0354	0.191406	-0.99634	-0.07275	-0.01343	6
42:04.0	-0.99683	-0.05615	0.1875	-0.97461	-0.0603	-0.01563	6

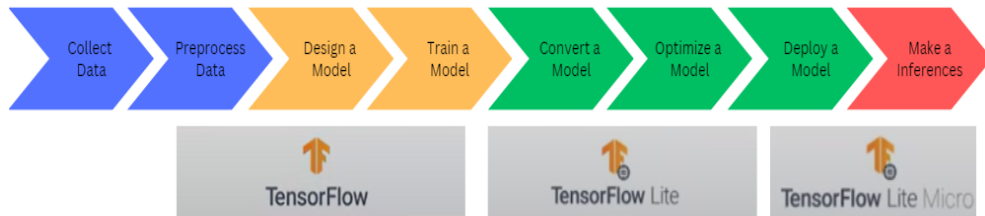


Figure: System flow

Implementation Modules:

- Development Module
- Optimization Module
- Deployment Module

Implementation: Development Module

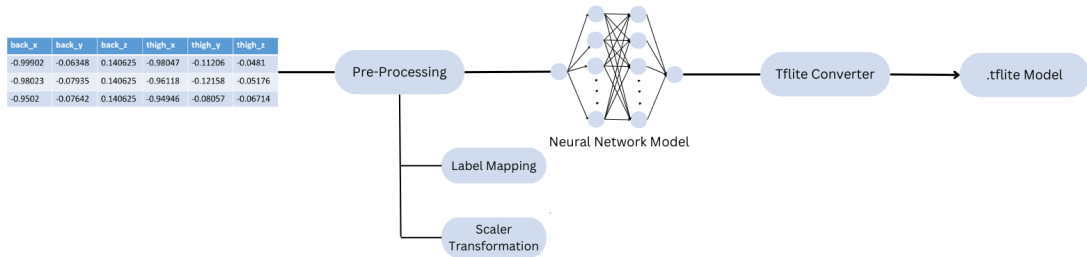


Figure: System flow

Implementation: Optimization Module

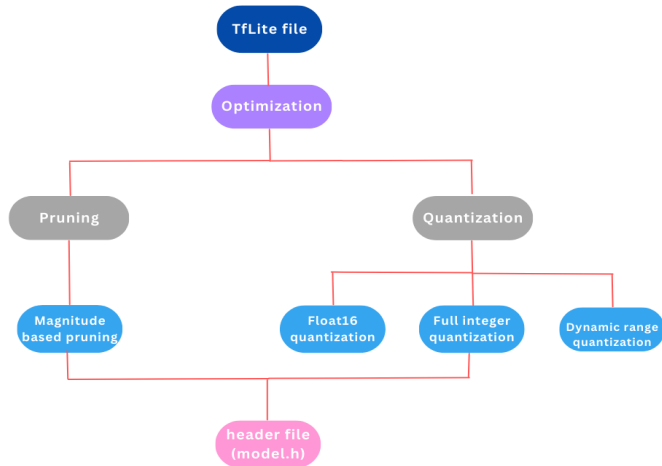


Figure: Optimal tflite conversion

Implementation: Deployment Module

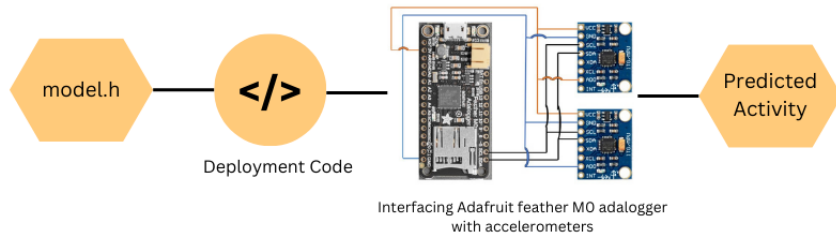
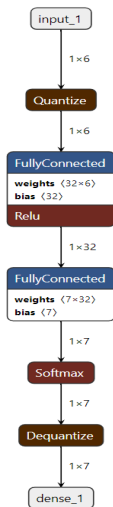


Figure: Optimal tflite conversion

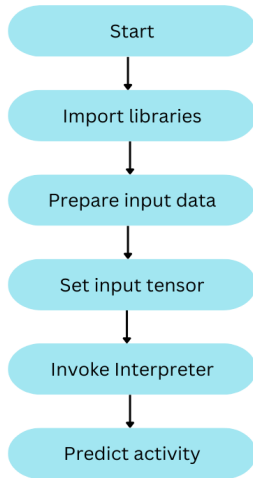
Intermediate Results:



- **Input:** 1×6 features.
- **Quantization:** Reduce computation and memory costs.
- **Layer 1:** Fully connected with **32 neurons**, ReLU activation, output 1×32 .
- **Layer 2:** Fully connected with **7 neurons**, softmax activation, output 1×7 .
- **Dequantization:** Output is dequantized to higher precision.
- **Output:** Dense layer produces 1×7 vector.

Figure: TfLite visualization

Software Module testing:



- Initially, we evaluated the performance of the Tensorflow model.
- Converted the model to TFLite format and observed that the model size was 3500 bytes.
- Applied quantization techniques, which reduced the model size to 2888 bytes.
- Optimization:**
 - To make the model more efficient in terms of size and speed.
- Tested the optimized TFLite model using Interpreter and it accurately predicted the correct labels.

Figure: Software testing

Intermediate Results:

```
to_predict = np.array([[-0.980225, -0.079346, 0.140625, -0.961182, -0.121582, -0.051758]], dtype=np.float32)
to_predict_scaled = scaler.transform(to_predict)
interpreter.set_tensor(input_details[0]['index'], to_predict_scaled)
interpreter.invoke()
output = interpreter.get_tensor(output_details[0]['index'])
predicted_class = np.argmax(output)
print("Predicted Class:", predicted_class)
```

Predicted Class: 4

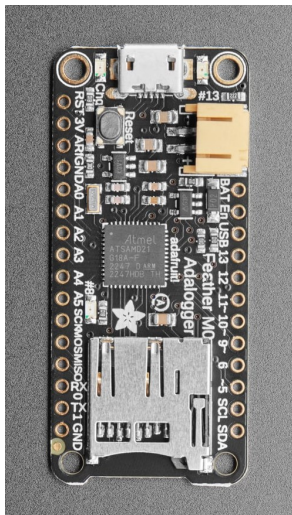
Figure: TFLite Prediction

The above figure showcases the predictions made by our TFLite model.

Expected Class : 4-stairs(descending)

Predicted Class : 4-stairs(descending)

Hardware: Adafruit Feather M0 Adalogger



- Measures 2.0" x 0.9" x 0.28" (51mm x 23mm x 8mm)
- 256KB of FLASH + 32KB of RAM
- 3.3V regulator with 500mA peak current output
- Hardware Serial, hardware I2C, hardware SPI support
- 10 x analog inputs
- Pin 13 red LED for general purpose blinking

Figure: Adafruit Feather M0 Adalogger

Hardware: Accelerometer MPU6050

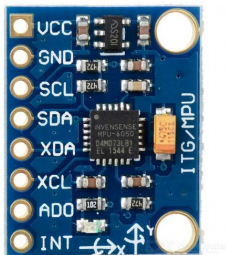


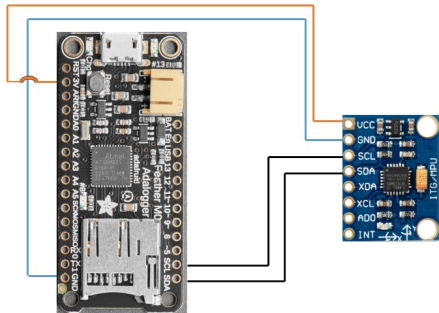
Figure: Accelerometer MPU6050

- MPU6050 has a 3-axis gyroscope, 3-axis Accelerometer and a Digital motion processor integrated on a single chip.
- Power supply of 3V-5V.
- I2C protocol for communication and transfer of data.
- MPU6050 operates on 500microA of current.
- MPU6050 also has an in-built temperature sensor.

Hardware Module Testing:

- **Initial Setup with Adafruit Feather M0 Adalogger:**

- Tested the Adafruit Feather M0 Adalogger by programming it to blink its inbuilt LED every 5 seconds to ensure proper functionality.



- VCC of accelerometer to 3.3V on the Adafruit board.
- GND of accelerometer to GND on the Adafruit board.
- SCL (clock line) of accelerometer to SCL on the Adafruit board.
- SDA (data line) of accelerometer to SDA on the Adafruit board.

Figure: Interfacing the First Accelerometer:

Hardware Module Testing:

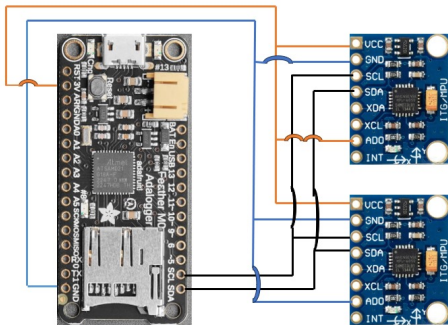


Figure: Interfacing the Second Accelerometer:

- VCC of accelerometer to 3.3V on the Adafruit board.
- GND of accelerometer to GND on the Adafruit board.
- SCL (clock line) of accelerometer to SCL on the Adafruit board.
- SDA (data line) of accelerometer to SDA on the Adafruit board.
- AD0pin of the first accelerometer is connected to GND on the adafruit board, the address is 0x68.
- AD0pin of the second accelerometer is connected to VCC on the adafruit board, the address is 0x69.

Implementation:

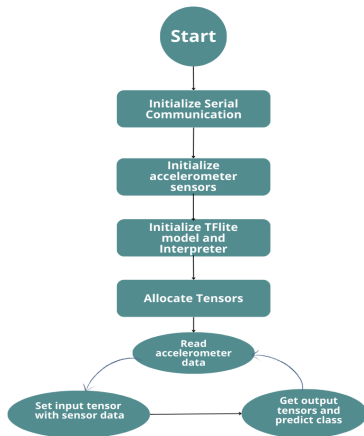
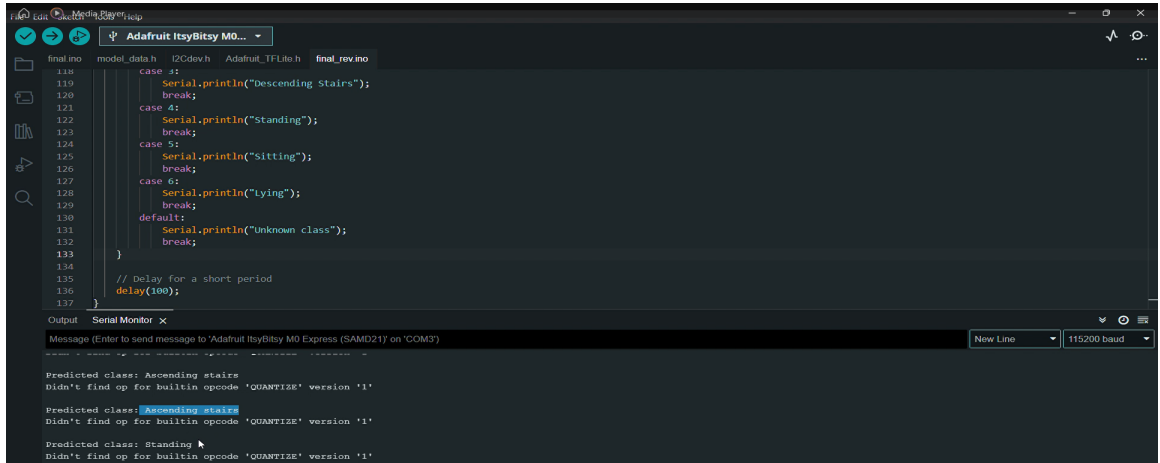


Figure: Deployment Code

• Header Files Used:

- **"Wire.h"** : Initializing and managing communication with the MPU6050 sensors.
- **"I2Cdev.h"** : Provides an easier interface to initialize the sensor and read data from it.
- **"MPU6050.h"** : Initialize the MPU6050 sensors.
- **"AdafruitTFLite.h"** : Initialize the TensorFlow Lite model, manage tensors, and perform inference using the model to classify activities based on the sensor data.
- **"modeldata.h"** : Contains the pre-trained TensorFlowLite model.

Results:



```
final.ino  model_data.h  I2Cdev.h  Adafruit_TFLite.h  final_rev.ino
118         case 3:
119             Serial.println("Descending Stairs");
120             break;
121         case 4:
122             Serial.println("Standing");
123             break;
124         case 5:
125             Serial.println("Sitting");
126             break;
127         case 6:
128             Serial.println("Lying");
129             break;
130         default:
131             Serial.println("Unknown class");
132             break;
133     }
134
135     // Delay for a short period
136     delay(100);
137 }
```

Output Serial Monitor x

Message (Enter to send message to 'Adafruit ItsyBitsy M0 Express (SAM21)' on 'COM3')

New Line 115200 baud

Predicted class: Ascending stairs
Didn't find op for builtin opcode 'QUANTIZE' version '1'

Predicted class: Ascending stairs
Didn't find op for builtin opcode 'QUANTIZE' version '1'

Predicted class: Standing
Didn't find op for builtin opcode 'QUANTIZE' version '1'

Figure: Human Activity Prediction

Comparative Analysis:

- Using TensorFlow Lite (TFLite) without optimization techniques, memory usage improved by 15.41 percent. Our model after applying quantization, has improved memory usage to 17.49 percent[3].

Before	After
3500 Bytes	2888 Bytes

- Network slimming method increases computational work and leads to accuracy loss while TFLite quantization reduces model size by maintaining accuracy.[5]
- For human activity recognition, Nvidia Jetson Xavier NX consumes up to 20 watts, supporting intensive AI and GPU tasks. while The Adafruit Feather M0 Adalogger consumes 0.6-0.9 watts, ideal for low-power IoT applications.[12]

References I



[Why i can't find the arduino_tensorflowlite library in library management?, November 2022.](#)

Accessed: 2024-05-17.



[Adafruit.](#)

[Adafruit feather huzzah with esp8266 - loose headers, 2024.](#)

Accessed: 2024-05-17.



[Juneseo Chang, Myeongjin Kang, and Daejin Park.](#)

[Low-power on-chip implementation of enhanced svm algorithm for sensors fusion-based activity classification in lightweighted edge devices.](#)

Electronics, 11(1), 2022.



[W Cho, H Lee, and J-h Gu.](#)

[Optimization techniques and evaluation for building an integrated lightweight platform for ai and data collection systems on low-power edge devices, 2024.](#)



[DigiKey.](#)

[Tensorflow lite micro - arduino library, 2024.](#)

Accessed: 2024-05-17.



[Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang.](#)

[Learning efficient convolutional networks through network slimming.](#)

arXiv preprint arXiv:1708.06519, 2017.



[Massimo Merenda, Carlo Porcaro, and Demetrio Iero.](#)

[Edge machine learning for ai-enabled iot devices: A review.](#)

Sensors, 20(9), 2020.

References II



Nafiul Rashid, Berken Utku Demirel, and Mohammad Abdullah Al Faruque.

Ahar: Adaptive cnn for energy-efficient human activity recognition in low-power edge devices.
IEEE Internet of Things Journal, 9(15):13041–13051, 2022.



Robocraze.

Mpu-6050 triple axis accelerometer and gyroscope module, 2024.
Accessed: 2024-05-17.



Bharath Sudharsan, John Breslin, and Muhammad Intizar Ali.

Edge2train: A framework to train machine learning models (svms) on resource-constrained iot edge devices.
IEEE Internet of Things Journal, pages 1–8, 2020.
Pages 1–8.



C Surianarayanan, JJ Lawrence, PR Chelliah, E Prakash, and C Hewage.

A survey on optimization techniques for edge artificial intelligence (ai), 2023.



Miljan Vuletić, Vladimir Mujagić, Nikola Milojevic, and Debmalya Biswas.

Edge ai framework for healthcare applications.
IEEE Access, May 2021.



xLAB for safe autonomous Systems.

Tensorflow lite for microcontrollers, 2024.
Accessed: 2024-05-17.