DEEPCRAFT™ Ready Model for Fall Detection

Introduction

In this document, we describe the DEEPCRAFT™ Ready Model for Fall Detection, an accelerometer-based Edge AI model developed by Imagimob, an Infineon Technologies company. This model is specifically designed to detect when a less active individual, like an elderly person, is falling. The user is expected to wear the device running this model on their wrist. We provide details about the technical specifications of this machine learning model, its performance in common scenarios, and various test results for the model as well as instructions on how to test it.

Contents

Introduction....1
Definitions....2
Model Tech Specs and Deployment....2
Dataset....2
Testing....3
Tests on wearable device....3
AI Evaluation Kit Testing Instructions....3
General Board Testing Instructions....4
Appendix I: Sensor Orientation – AI Evaluation Kit....5
Appendix II: Code Sample....5
Appendix III: Sensor Orientation – General Board....6

Definitions

Sensitivity
Sensitivity is a measurement of the likelihood of the model to detect a fall that occurs.

$$Sensitivity = \frac{Falls\ detected}{Total\ falls}$$

False positives, false positive rate, and false positive per user per week
A false positive (FP) happens when the model reports a fall when no fall actually occurred. To get a better view of the frequency of the number of FPs over time, we use the following metric:

$$FP\ per\ user\ per\ week = \frac{FP}{Users * weeks}$$

Model Tech Specs and Deployment

The DEEPCRAFT™ Ready Model for Fall Detection is able to detect a fall from accelerometer sensor data with the following sensor settings:

- Scale = +/- 8g
- Resolution = 12 or 16 bit
- Sampling Frequency = 50 Hz

The model's C library has the following memory footprint:

- Flash: 15 kB
- RAM: 6 kB

Its inference time is about 1 ms when running on a board with PSOC™ 6. The model outputs a prediction every 20 ms, namely 50 times per second.

Dataset

This DEEPCRAFT™ Ready Model was built using positive and negative data. The positive data comes from different individuals simulating falls. The negative data (or non-falls) were collected through a number of different methods; initially sensors were installed on users doing various activities. The model was then improved further by running the model in real-time and using it to curate data and implement continuous learning in order to further improve the model.

Testing

Tests on wearable device

Two tests were performed on a wearable device, a short test and an extended test.

The short test involved falling 30 times with the wearable on, and measuring the sensitivity to make sure that it did not vary too much from the results against the test sets. The falls included falling in different directions, active falls (where the individual was moving) and static falls (where the person was still).

The extended test involved having the wearable on for five days. This was performed with five people. The output of this test is a measurement of the number of false positives per user per week.

Clarification: The false positives per user per week is specified as a range. This is because this is measured continuously, and the result varies depending on how active the users are within the period.

Model version	Sensitivity	False positives per user per week
1.9 (current)	93.3% (28/30)	1.8-2.25
1.8	93.3% (28/30)	1.8-2.25
1.7	90% (27/30)	2.275-2.8
1.6 (baseline)	90% (54/60)	N/A

AI Evaluation Kit Testing Instructions

The model is tailored to detect dangerous falls of elderly people, and it performs best on unpadded falls. However, we recommend testing with a healthy adult and placing pillows to avoid injury - the model works well in these instances too. The model is designed to have a very low FP per user per week; for that reason, it requires the person to lay still for 7-10 seconds after the fall.

To test this model on the PSOC™ 6 Artificial Intelligence Evaluation Kit, follow the steps below:

1. Download the hex file.
2. Connect your board to a PC using the USB-C J1 port, which is used for flashing the board.
3. Flash the hex file onto the board.
    - Download ModusToolbox™ Programming Tools.

- Choose the correct 'Programmer' at the top.
- Choose the CY8KIT-062S2-AI 'Board'.
- Select the hex file.
- Press 'Connect'.
- Press 'Program'. You can now close ModusToolbox™ Programming Tools.

4. Disconnect the board from the PC and connect a Lithium Ion Battery rated 3.6V-4.2V with JST-PH connector.
5. Attach them to your wrist with orientation as per Appendix I — using strong tape to hold it in place is recommended.
6. Fall as realistically as possible in a safe manner; make sure to lie still for 10 seconds after the fall.
7. If a second red LED on the front of the board lights up, a fall has been detected. It stays lit for 10 seconds.

General Board Testing Instructions

To test this model on a general board, follow the steps below:

1. Download ModusToolbox™ and create a new firmware project for this board that gives access to the accelerometer data of the integrated IMU sensor. Use sensor's settings in the 'Model Tech Specs' and 'Deployment' sections.
2. Scale the accelerometer's values to SI metric system.
3. Include in the project the provided library's *fall_lib_eval.a* and *fall_lib.h* files.

In the main loop in *main.c*, add your code to continuously run the accelerometer data through the AI model and print the results to a serial terminal. See Appendix II for an example.

4. Flash the board.
5. Open a serial terminal to observe the prints. Terminal settings:
   a. COM port is dependent on the computer being used, check device manager to find the port number.
   b. Speed: 115200
   c. Data: 8 bit
   d. Parity: none
   e. Stop bits: 1
   f. Flow control: none
8. Wear the board on your wrist making sure to have the right IMU sensor orientation. See Appendix III for more details.
9. Fall as realistically as possible in a safe manner; make sure to lie still for 10 seconds after the fall.

Appendix I: Sensor Orientation – AI Evaluation Kit

To correctly set up the IMU orientation, make sure that the board orientation is as shown in Figure 1 below.

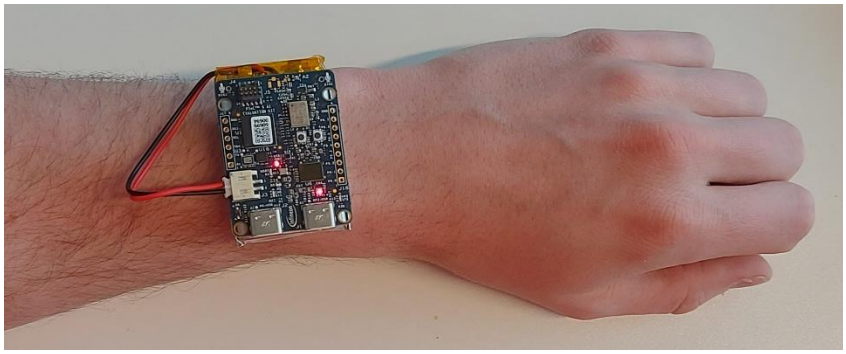

Figure 1: Bird's eye view of arm flat on table.

Appendix II: Code Sample

This is a code sample for running the fall detection model on a general board.

```
// Main loop
for ( ; ; )
        {
        …
        // Get raw accelerometer data from IMU sensor
        …
        int acc_x = < X RAW DATA > ;
        int acc_y = < Y RAW DATA > ;
        int acc_z = < Z RAW DATA > ;

        // Set the SI metric system scaling factor
        float ACC_SCALE = < YOUR SI SCALING FACTOR > ;

        // Convert raw data into SI data
        float acc_x_f = (float) acc_x * ACC_SCALE;
        float acc_y_f = (float) acc_y * ACC_SCALE;
        float acc_z_f = (float) acc_z * ACC_SCALE;
```

```
// Store data in input vector
float dataIn[3] = {acc_x_f, acc_y_f, acc_z_f};

// Enqueue the input data in the AI model buffer
int enq_res = IMAI_FED_enqueue(dataIn);

// Run the input data through the AI model and store
// results in output data vector
int deq_res = IMAI_FED_dequeue(dataOut);

// Print to serial terminal when a fall is detected
if (dataOut[1] > 0)
        printf("Fall Detected!\r\n");  // Fall Detected
...
}
```

Appendix III: Sensor Orientation – General Board

To correctly set up the IMU orientation, make sure that the accelerometer X, Y, Z axis and values are as shown in the figures below:

Figure 2: Y = 1, X, Z = 0 --- hand held up

Figure 3: X = -1, Y, Z = 0 --- hand outstretched, palm facing front

Figure 4: Z = 1, X, Y = 0 --- hand outstretched, palm up

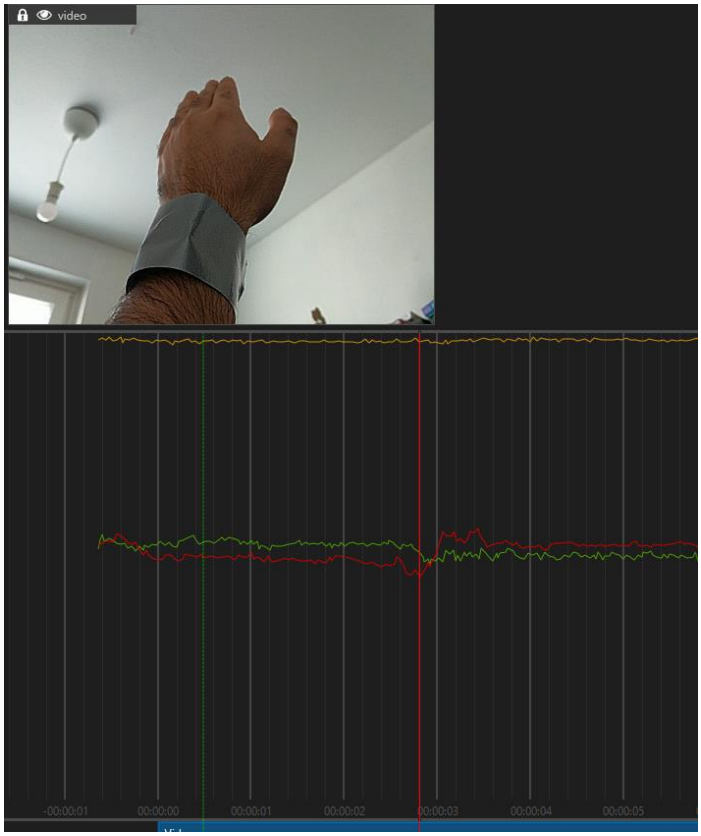

Figure 2

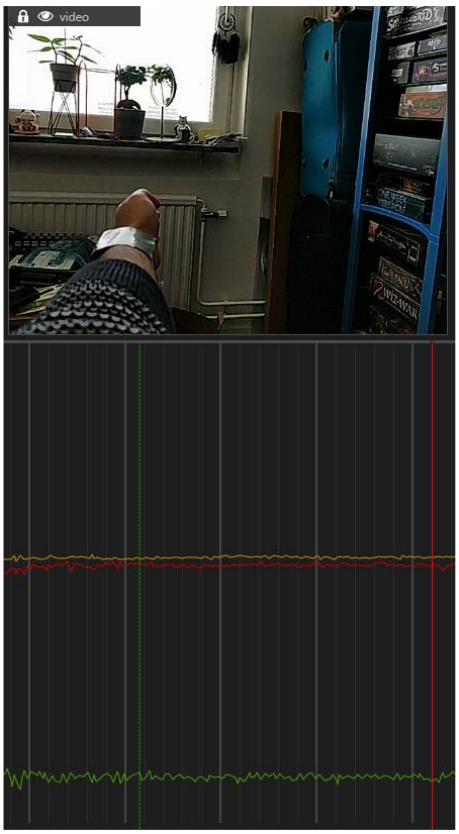

Figure 3

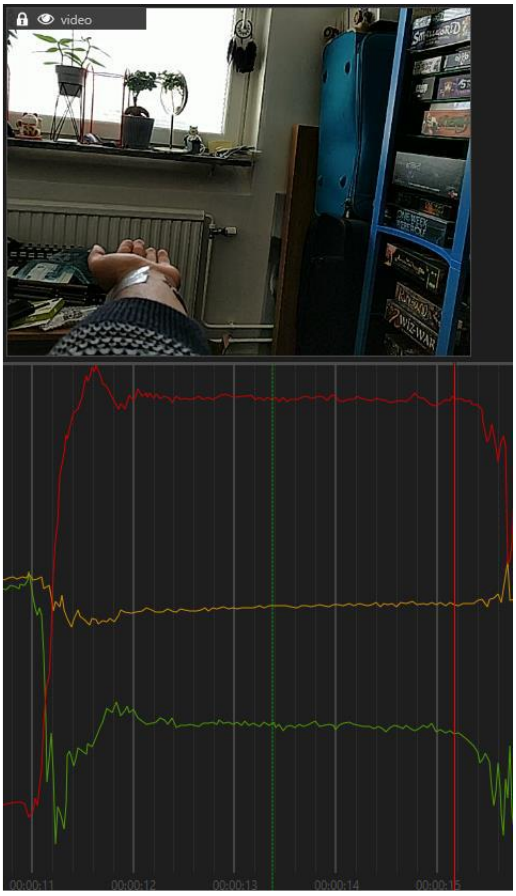

Figure 4