Title page: EEDG/CE 6302 Lab Report 3
CE6302, Embedded Systems, 302 Laboratory - 83204
Lab Topic: TinyML - Continuous Motion Recognition
Yuyang Hsieh, YXH230019
Lab partner name: Xiongtao Zhang
Group Number 4
Instructor name: Tooraj Nikoubin
TA name: Seyed Saeed
Date: 9/15/2024

1. Objective:

Purpose: The purpose of this lab is to introduce TinyML and its applications on low-power embedded systems. It covers basic ML algorithms and demonstrates their deployment on the TI Launchpad (CC1352P) with Booster Sensors. The lab focuses on real-time gesture recognition and TinyML's role in IoT and edge computing.

Methods used: In lab 3, the method is using TinyML to implement machine learning in embedded microcontrollers. There are mainly three segments, data collection, model training and verifying, model deploying. Data collection is based on TI Launchpad (CC1352P) with Booster Sensors which collect data and send it to the cloud, and Edge Impulse which is a cloud based platform storing collected data. Model training and verifying are all performed on Edge Impulse. Model deploying is to deploy the trained model from the cloud to the device.

Results: After collecting the data, generating the model, and deploying the model, the device can detect its movement behavior among up down, wave, sneak, and idle, with corresponding value.

Hardware used: TI Launchpad (CC1352P) and Booster Sensors.

Software used: Edge Impulse CLI, Texas Instruments UniFlash, and Edge Impulse.

Major Conclusions: Implementing TinyML can make microcontrollers utilize machine learning. With cloud based platform Edge Impulse, model training and other high-computation tasks are pre-completed, which make machine learning related applications achieve low-power and low-cost targets on the microcontrollers.

2. Introduction:

2.1 Hardware and Software Background Information

Hardware Background Information:TI Launchpad (CC1352P):This LaunchPad speeds development on devices with integrated power amplifier and multi-band radio support for concurrent Sub-1Ghz and 2.4-GHz operation. Protocols supported include Bluetooth® Low Energy, Sub-1 GHz, Thread, Zigbee®, 802.15.4, and proprietary RF with the compatible CC13x2-CC26x2 SDK. Available variations feature different RF matching networks and power levels.

The Sensors BoosterPackTM kit (BOOSTXL-SENSORS) is an easy-to-use plug-in module for adding digital sensors to your LaunchPadTM development kit design. SimpleLinkTM microcontroller (MCU) LaunchPad development kit developers can use this BoosterPack plug-in module to start developing sensor applications using the onboard gyroscope, accelerometer, magnetometer, pressure, ambient temperature, humidity, ambient light, and infrared temperature sensors.

Software Background Information: Edge Impulse is ushering in the future of embedded machine learning by empowering developers to create and optimize solutions with real-world data. It makes the process of building, deploying, and scaling embedded ML applications easier and faster than ever, unlocking massive value across every industry, with millions of developers making billions of devices smarter.

2.2 Purpose of the Experiment

The purpose of lab 3 is using TinyML to perform machine learning on microcontrollers. It includes collecting data from embedded device and upload to cloud platform, Edge Impulse, through serial ports; Processing signals and distinguishing between the four (idle, snake, wave, updown) classes with designing an impulse on cloud; Generating features based on previous data and training neural network model; After that, classifying the new Data with trained model to evaluate its performance, and performing anomaly detection to improve it; Finally, deploying the model back to device.

2.3 Summary of the Experiment

In lab 3, the flow of implementing TinyML on microcontroller is demonstrated. From data collection, signal processing to generating features, training model, verifying the model, and deployment, the procedures of applying TinyML are completed. By applying TinyML, computation-intensive and memory-intensive applications are performed on the cloud, models and related applications are optimized to be able to run on the microcontrollers and embedded systems with low-power and real-time features.

2.4 Findings of the experiment

During data collection, we find that the rate of training data and test data are needed to split into proper proportions(e.g. 8:2), so that the neural network model can be trained and verified accurately. Overall, by demonstrating the implementation of TinyML on microcontroller, we learn how to perform ML with high accuracy in low-cost and low-power embedded device.

3. Explanation:

3.1 Experiment procedure

First, download and install Python 3 and Node.js, then install the Edge Impulse CLI tools. After creating an Edge Impulse account, use the CLI to login and connect with Edge Impulse Studio. Next, install Texas Instruments Uniflash to manage the firmware flashing process. To interface the CC1352P Launchpad with sensor hardware, connect the BOOSTXL-SENSORS to collect accelerometer data. Download the latest Edge Impulse firmware and open flash_windows.bat to flash and reset the firmware. Once the firmware is updated, set up the keys using edge-impulse-daemon, which will launch a wizard prompting you to log in and select an Edge Impulse project. To switch projects, run the command with the --clean flag. Connect to the serial port corresponding to the Application/User UART to start recording data. In the Edge Impulse data acquisition page, set the sample length to 10,000ms, choose the built-in accelerometer as the sensor, and set the frequency to 62.5Hz, indicating you want to record data for 10 seconds. Record data for four classes: updown, wave, snake, and idle, with 10 sets for each class. Next, design an impulse using the 'Spectral Analysis' signal processing block, which applies a filter, performs spectral analysis, and extracts frequency and spectral power data. This block will help the classifier distinguish between the four classes (idle, snake, wave, and updown). Configure the spectral analysis block, and the Feature Explorer will display extracted features, enabling you to visually separate the data. Then, configure the neural network, which will take the signal processing data as input and map it to one of the four classes. After that, implement anomaly detection, which creates clusters around familiar data and flags samples as anomalies if the distance from a cluster is too large. Finally, deploy the model back to the CC1352P Launchpad by downloading the necessary files, flashing, and resetting the device. Open

a terminal and run edge-impulse-run-impulse to see output samples for the four motion classes based on your movements.

3.2 Experiment Images

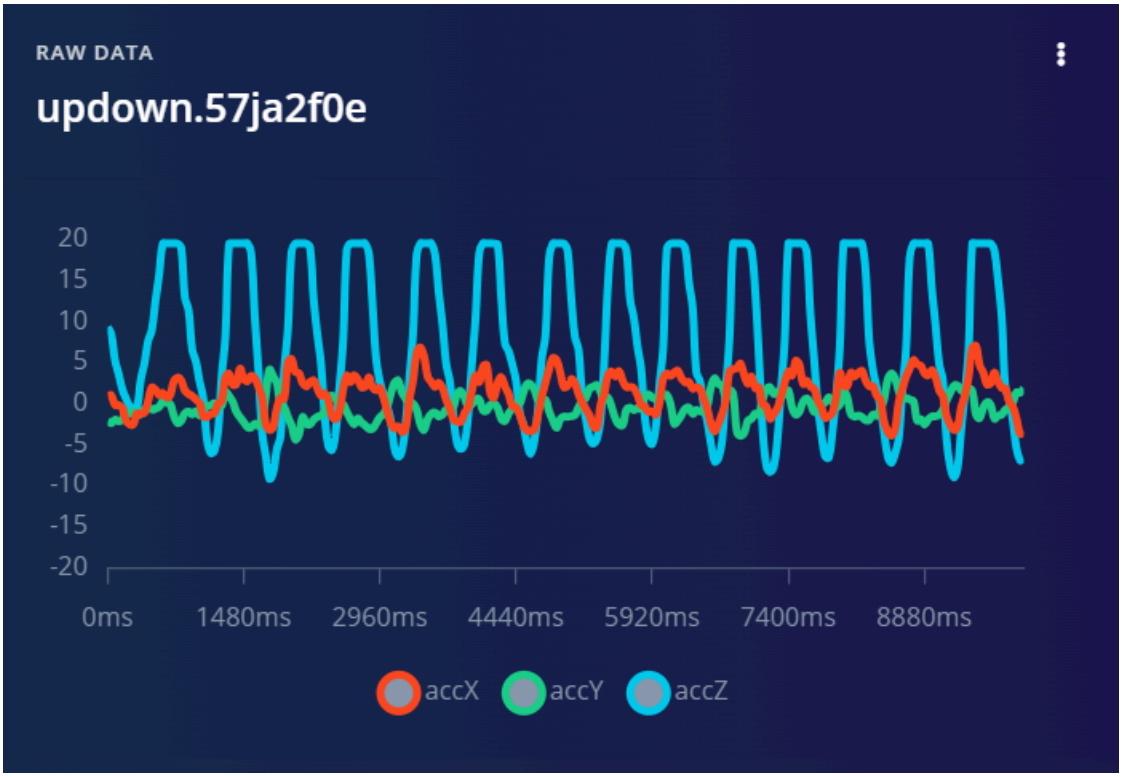


Image 3.2-1: updown data

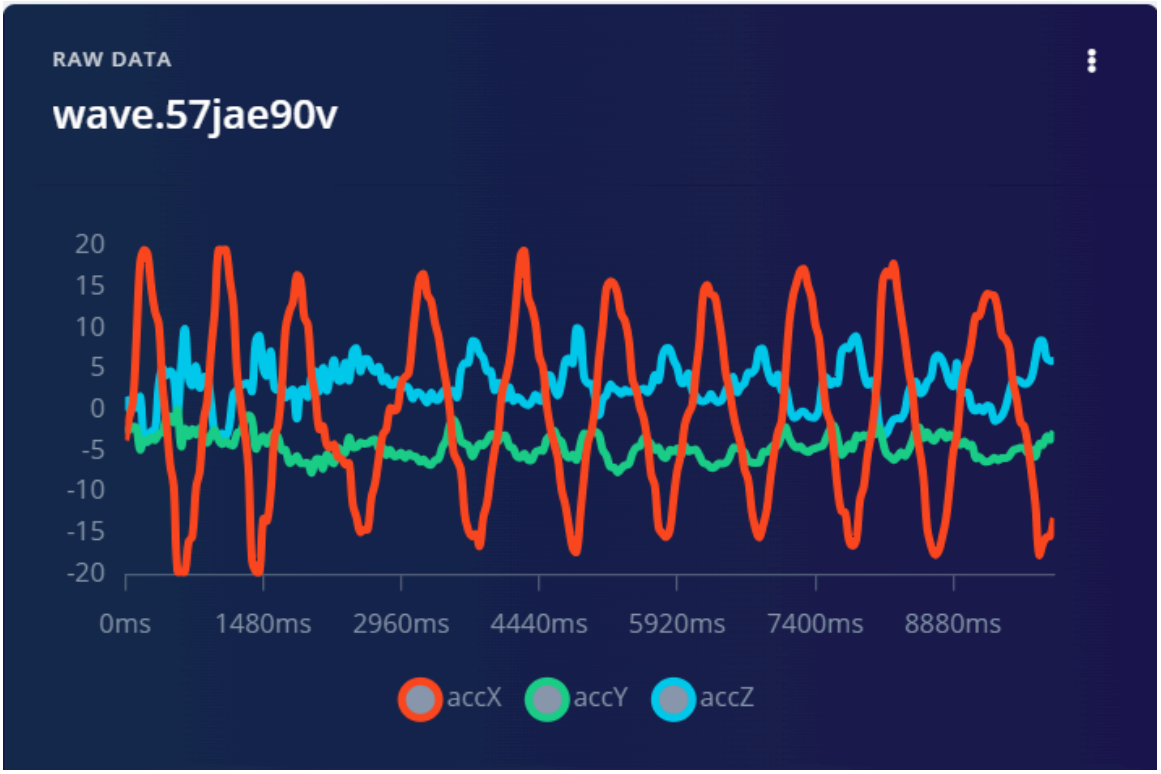


Image 3.2-2: wave data

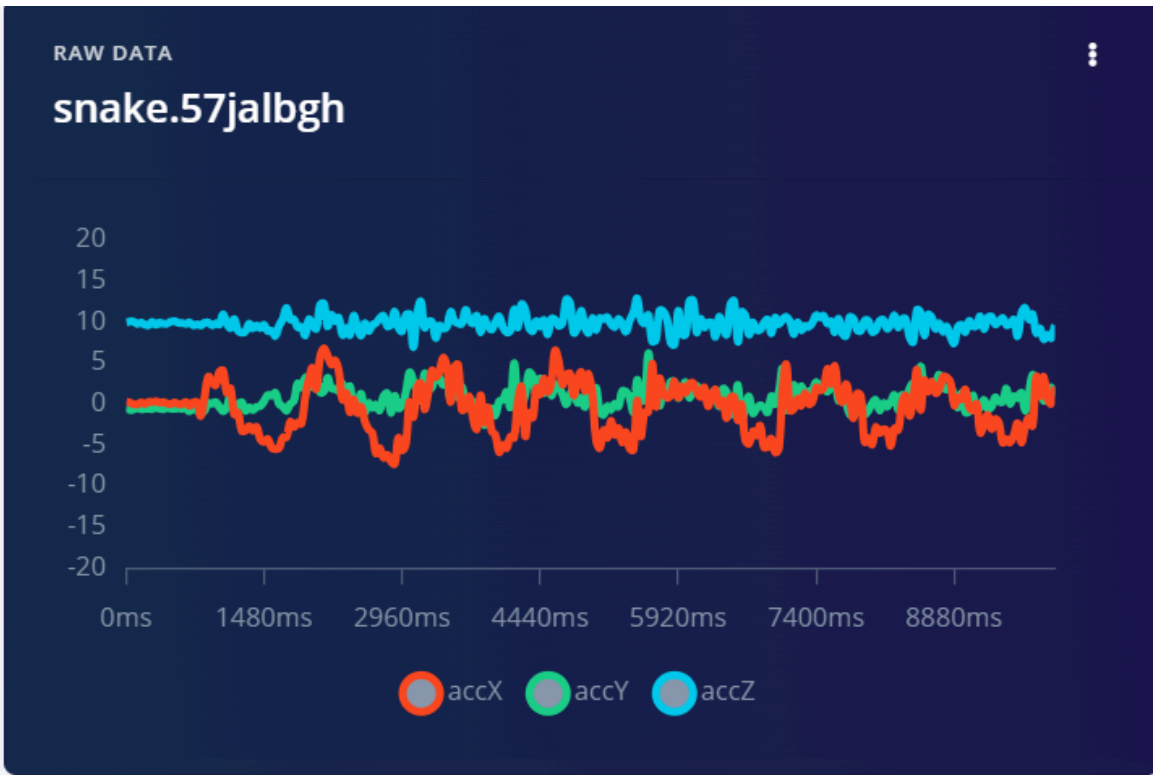


Image 3.2-3: snake data

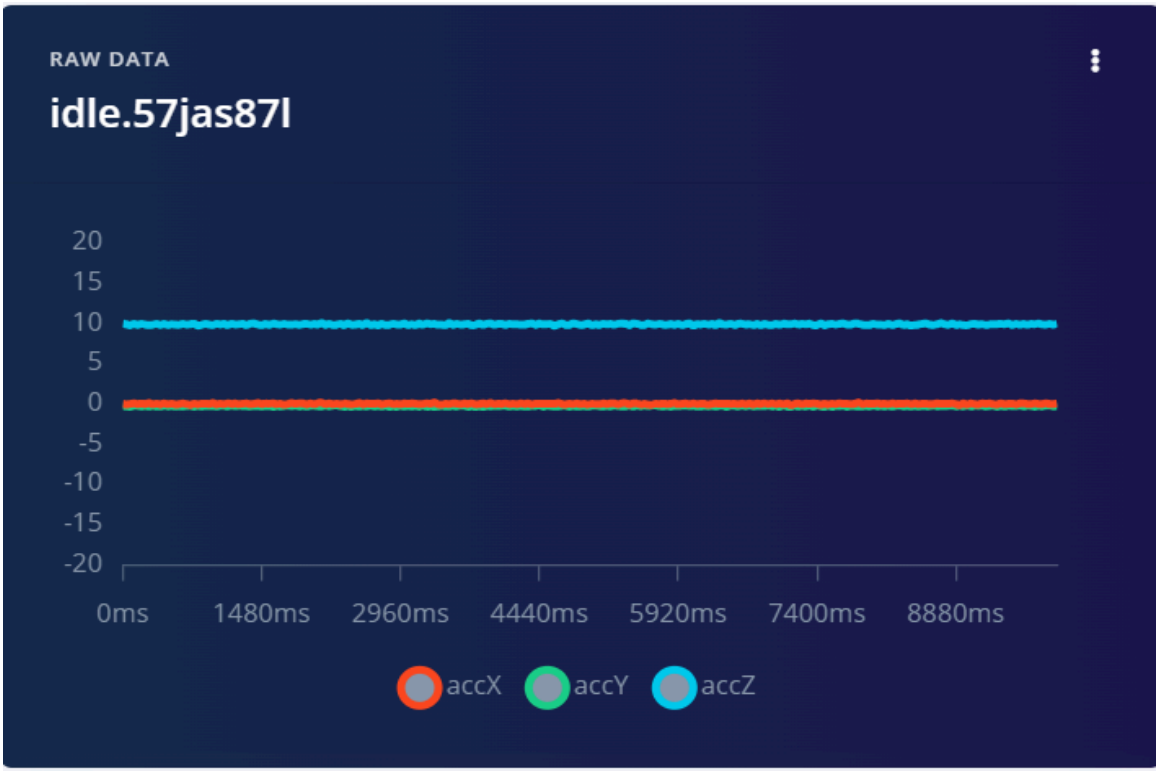


Image 3.2-4: idle data

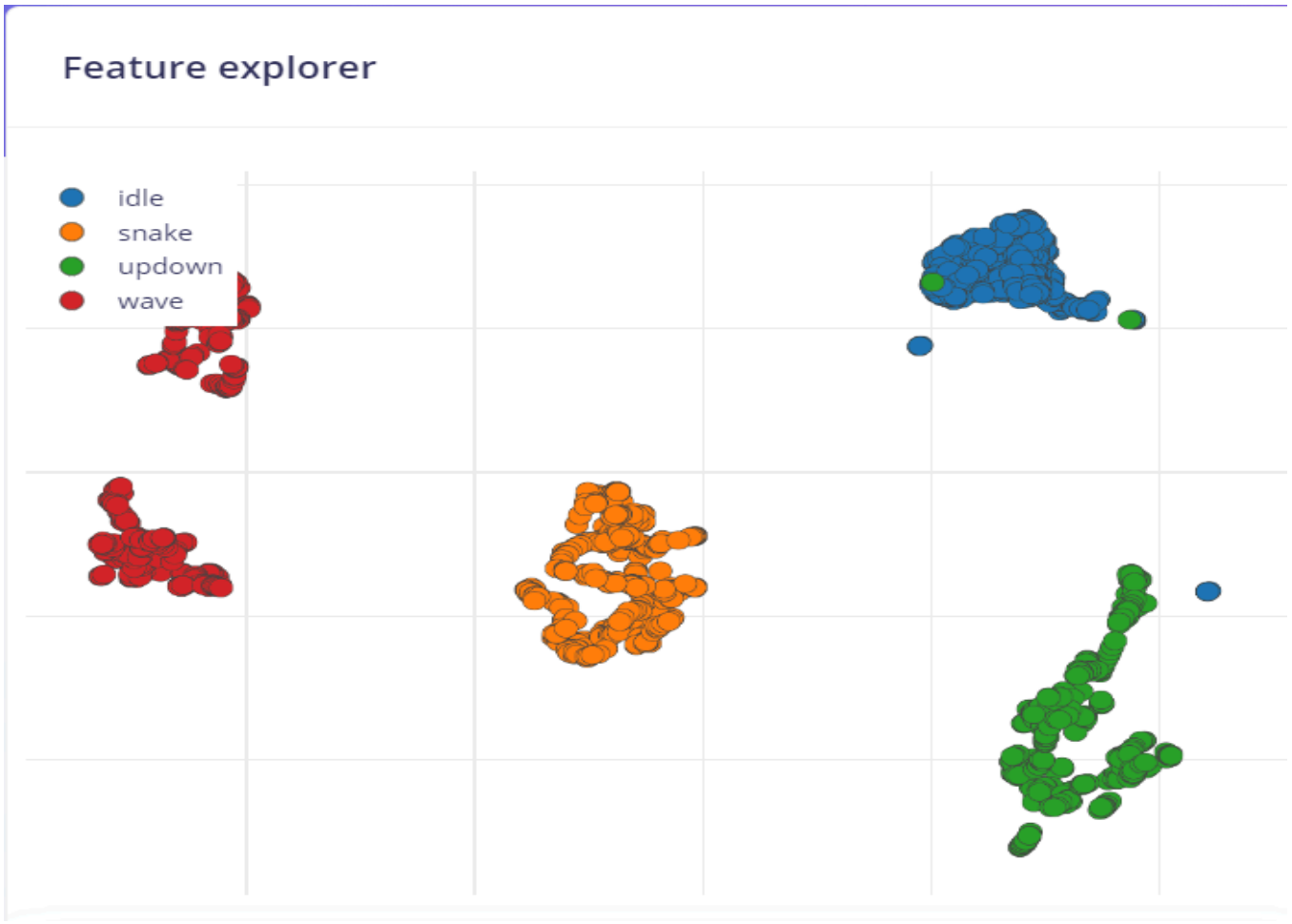


Image 3.2-5: full dataset using the feature explorer

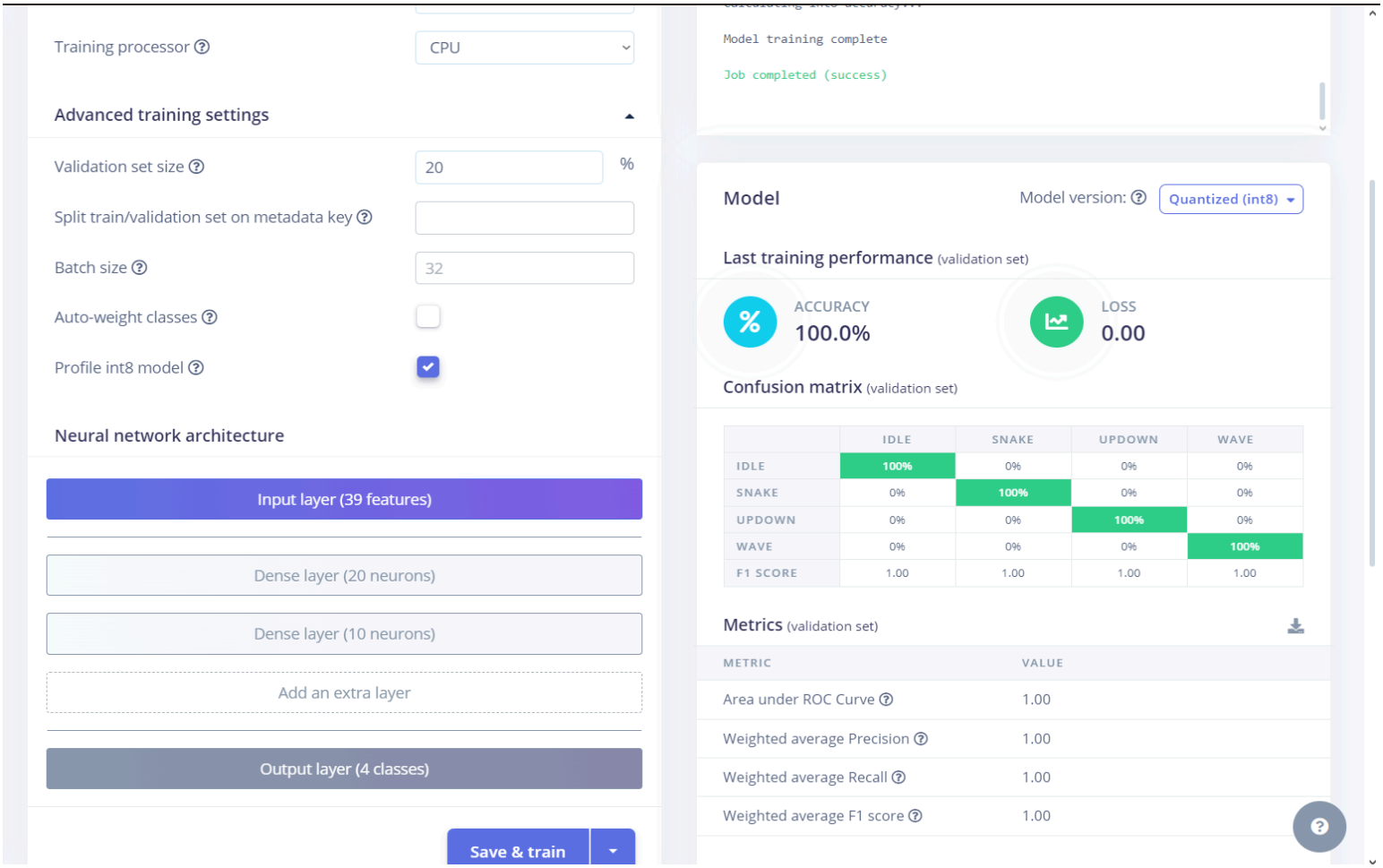
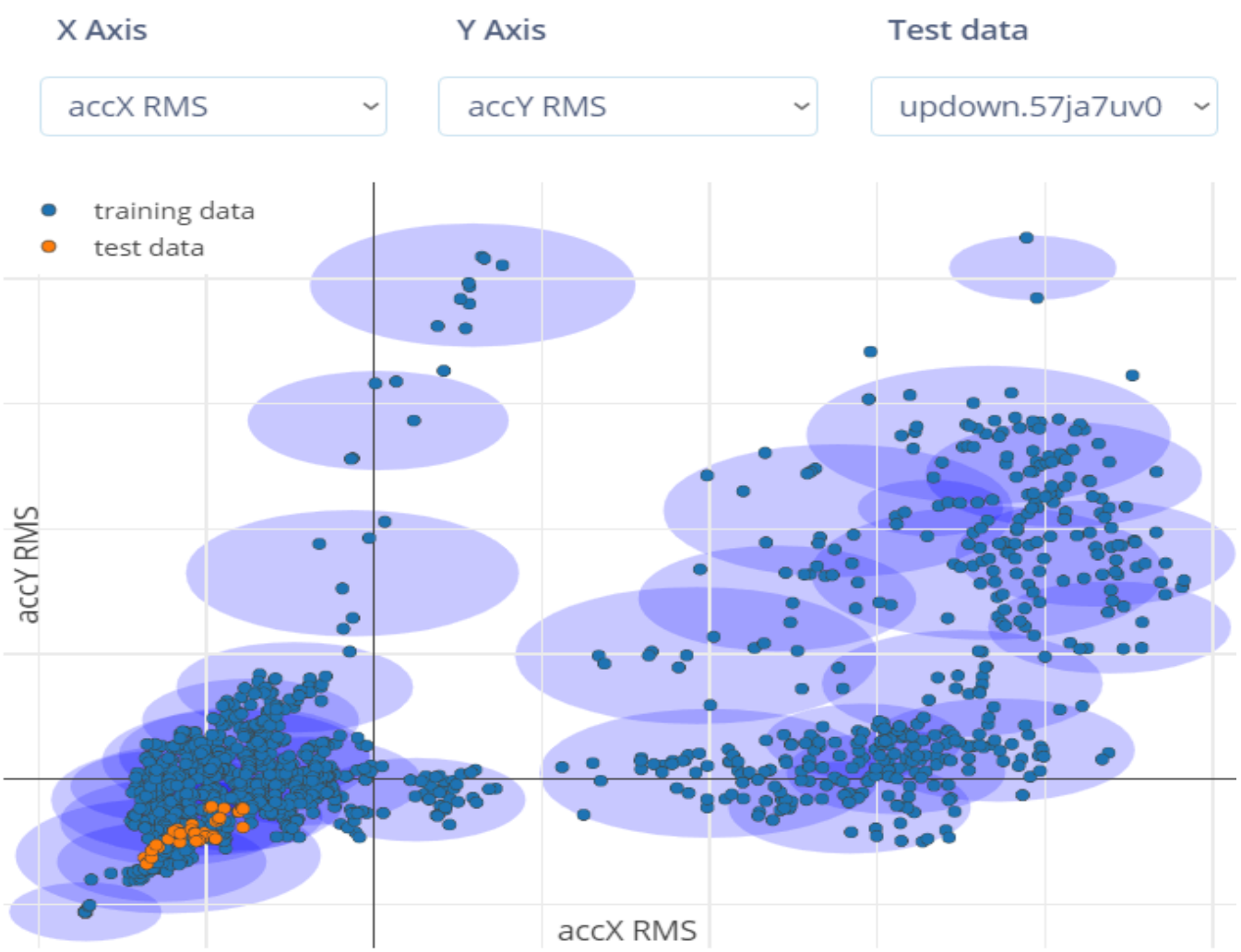


Image 3.2-6: Training performance after a single iteration



Anomaly score

`min: -0.3757, max: 0.0407, avg: -0.1899`

Average axis distance

`accX RMS: 0.0640, accY RMS: 0.1052, accZ RMS: 0.1232`

Image 3.2-7: Known clusters in blue, the live input data in orange

```
Administrator: C:\Windows\system32\cmd.exe - "node"  "C:\Users\yuyan\AppData\Roaming
    idle: 0.000000
    snake: 0.000000
    updown: 0.996094
    wave: 0.000000
Anomaly prediction: -0.159021
Sampling...
Predictions (DSP: 79 ms., Classification: 1 ms., Anomaly: 2 ms.):
#Classification results:
    idle: 0.000000
    snake: 0.000000
    updown: 0.996094
    wave: 0.000000
Anomaly prediction: -0.072045
Sampling...
Predictions (DSP: 79 ms., Classification: 1 ms., Anomaly: 2 ms.):
#Classification results:
    idle: 0.000000
    snake: 0.000000
    updown: 0.996094
    wave: 0.000000
Anomaly prediction: -0.062961
Sampling...
Predictions (DSP: 79 ms., Classification: 1 ms., Anomaly: 2 ms.):
```

Image 3.2-8: Sample output of updown type motion

```
Anomaly prediction: 0.348702
Sampling...
Predictions (DSP: 79 ms., Classification: 1 ms., Anomaly: 2 ms.):
#Classification results:
    idle: 0.000000
    snake: 0.097656
    updown: 0.000000
    wave: 0.902344
Anomaly prediction: 2.335543
Sampling...
Predictions (DSP: 79 ms., Classification: 1 ms., Anomaly: 2 ms.):
#Classification results:
    idle: 0.000000
    snake: 0.007812
    updown: 0.000000
    wave: 0.992188
Anomaly prediction: 0.893690
```

Image 3.2-9: Sample output of wave type motion

```
Anomaly prediction: -0.070244
Sampling...
Predictions (DSP: 79 ms., Classification: 1 ms., Anomaly: 2 ms.):
#Classification results:
    idle: 0.000000
    snake: 0.996094
    updown: 0.000000
    wave: 0.000000
Anomaly prediction: -0.173155
Sampling...
Predictions (DSP: 79 ms., Classification: 1 ms., Anomaly: 2 ms.):
#Classification results:
    idle: 0.000000
    snake: 0.996094
    updown: 0.000000
    wave: 0.000000
Anomaly prediction: -0.156452
Sampling...
Predictions (DSP: 79 ms., Classification: 1 ms., Anomaly: 2 ms.):
#Classification results:
    idle: 0.000000
    snake: 0.996094
    updown: 0.000000
    wave: 0.000000
Anomaly prediction: -0.142318
```

Image 3.2-10: Sample output of snake type motion

```
Administrator: C:\Windows\system32\cmd.exe - "node"  "C:\Users\yuyan\AppData\Roaming\
   snake: 0.000000
   updown: 0.000000
   wave: 0.000000
nomaly prediction: -0.219242
ampling...
redictions (DSP: 79 ms., Classification: 1 ms., Anomaly: 2 ms.):
Classification results:
   idle: 0.996094
   snake: 0.000000
   updown: 0.000000
   wave: 0.000000
nomaly prediction: -0.211172
ampling...
redictions (DSP: 79 ms., Classification: 1 ms., Anomaly: 2 ms.):
Classification results:
   idle: 0.996094
   snake: 0.000000
   updown: 0.000000
   wave: 0.000000
nomaly prediction: -0.218762
ampling...
redictions (DSP: 79 ms., Classification: 1 ms., Anomaly: 2 ms.):
Classification results:
   idle: 0.996094
   snake: 0.000000
   updown: 0.000000
   wave: 0.000000
nomaly prediction: -0.184183
ampling...
```

Image 3.2-11: Sample output of idle type motion

4. Discussions and Conclusions:

4.1 Comparison of your experimental results with the research/theory of the concept with reasoning.

In this lab, the experimental results agree with the theoretical ideas of TinyML and real-time motion classification using edge devices. The integration of machine learning with embedded systems theoretically enables efficient on-device data processing, allowing for low-latency, real-time analysis. In practice, the lab successfully demonstrated this by deploying a neural network model onto the CC1352P Launchpad, achieving the expected classification of different motion patterns. Additionally, spectral analysis proved effective in extracting meaningful features from time-series data, such as accelerometer readings, confirming its utility in motion classification tasks, as predicted by theory.

4.2 Learnings from the experiment

In this lab, the process of flashing new firmware into the CC1352P Launchpad and integrating it with Edge Impulse for real-time motion classification was explored. Practical experience was gained in collecting and processing accelerometer data, configuring neural networks, and applying spectral analysis for classifying various motion patterns. Additionally, the implementation of anomaly detection was covered, along with the deployment of a machine learning model back to the hardware. This further deepened the understanding of TinyML and embedded systems.

4.3 Video Link: https://youtu.be/EGOf1pkJG4g

5. References:

Texas Instruments: https://www.ti.com/
Texas Instruments, BOOSTXL-SENSORS Guide: https://www.ti.com/lit/ug/slau666b/slau666b.pdf
Edge Impulse: https://edgeimpulse.com/