Vishnuvaradhan Moganarengam (VXM210090)

EEDG6302: Microprocessor and Embedded Systems

Wednesday Lab Report

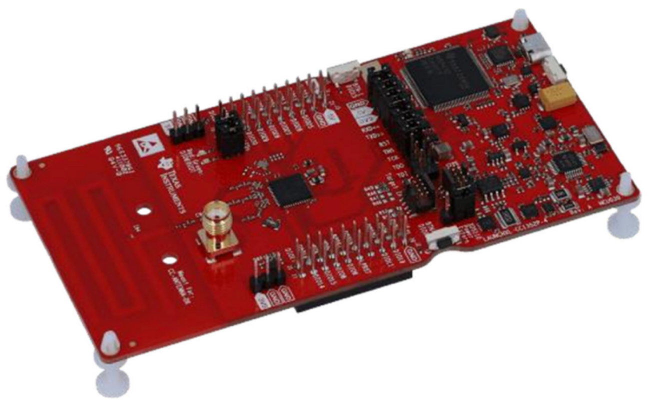
Project 3 Task 1: To understand the concept of TinyML, ML algorithms and TI Launchpad and Booster Sensors.

Introduction:

TinyML is a rapidly emerging field that combines Machine Learning and Embedded Systems, bridging the gap between these two domains. Embedded systems are computing devices that are typically small or tiny, yet capable of performing computations and operating for days or even weeks. Edge computing, on the other hand, involves bringing computation and data storage closer to the origin of data. Many edge devices that are integrated with IoT-based ecosystems are primarily designed to collect sensor data and transmit it to a nearby or remote cloud.

Texas Instruments CC1352P LaunchPad:

This LaunchPad accelerates development on devices that have integrated power amplifier and multi-band radio support for concurrent Sub-1Ghz and 2.4-GHz operation. It supports various protocols such as Bluetooth Low Energy, Sub-1 GHz, Thread, Zigbee, 802.15.4, and proprietary RF with the compatible CC13x2-CC26x2 SDK. Additionally, it features broad band antenna support for Sub-1 GHz frequency bands at 868 MHz, 915 MHz, and 433 MHz, as well as 2.4 GHz frequency bands.



Edge Impulse:

This is a cloud service designed for developing machine learning models specifically for TinyML targeted edge devices. It supports AutoML processing for edge platforms and enables deployment of learning models to a variety of boards including smartphones. The training process is carried out on the cloud platform, and the trained model can be exported to an edge device by following a data forwarder enabled path. The impulse can then be run locally using the in-built C++, Node.js, Python, and Go SDKs. Additionally, impulses can be deployed as a WebAssembly library.

Data Sampling:

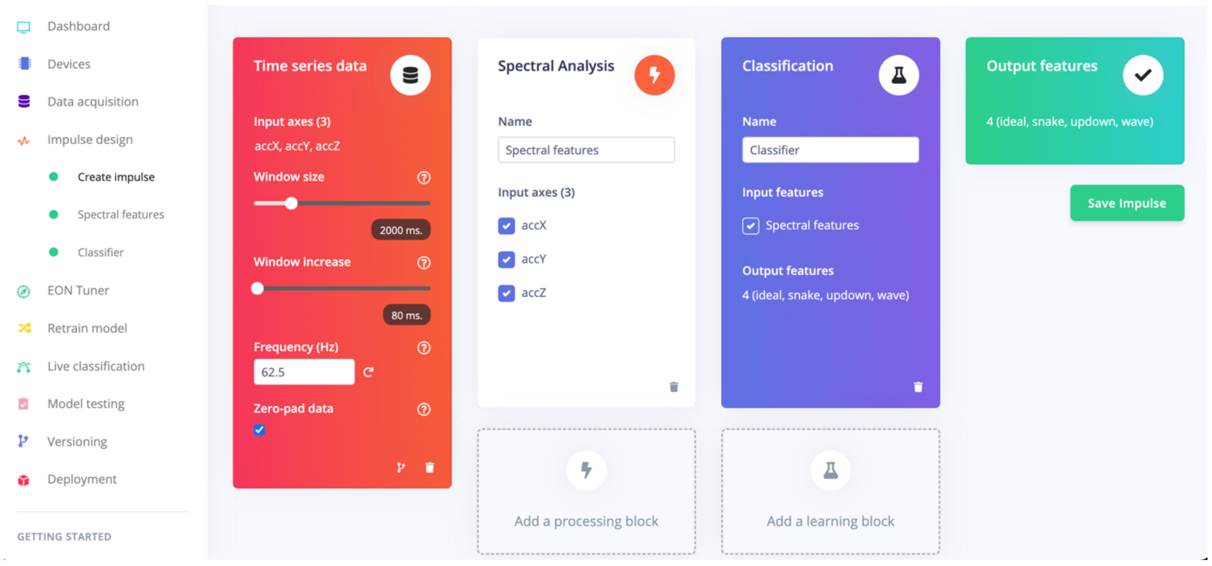
As machine learning thrives on large datasets, a single sample would not be sufficient. Therefore, it is crucial to begin constructing your own dataset. The following four classes require recording approximately three minutes of data:

Ideal: Just leave the device rest on table without any movement
Snake: Move the device like snake movement
Updown: Move the device Up and Down
Wave: Wave the device Left and Right

Designing an Impulse:

Once you have established the training set, you can proceed with designing an impulse. An impulse function by taking raw data and dividing it into smaller windows, utilizing signal processing blocks to extract features, and employing a learning block to classify new data. Signal processing blocks return consistent values for the same input, which simplifies the processing of raw data, while learning blocks learn from previous experiences.

For this project, we will be using the 'Spectral analysis' signal processing block. This block utilizes a filter, performs spectral analysis on the signal, and extracts frequency and spectral power data. We will then use a 'Neural Network' learning block, which takes these spectral features and learns to differentiate between the four classes: idle, snake, wave, and updown.



Time series data

Input axes (3): accX, accY, accZ

Window size: 2000 ms.

Window increase: 80 ms.

Frequency (Hz): 62.5

Zero-pad data: ✓

Spectral Analysis

Name: Spectral features

Input axes (3): ✓ accX, ✓ accY, ✓ accZ

Classification

Name: Classifier

Input features: ✓ Spectral features

Output features: 4 (ideal, snake, updown, wave)

Output features

4 (ideal, snake, updown, wave)

Spectral Analysis Block:

The spectral features block contains the following graphs:

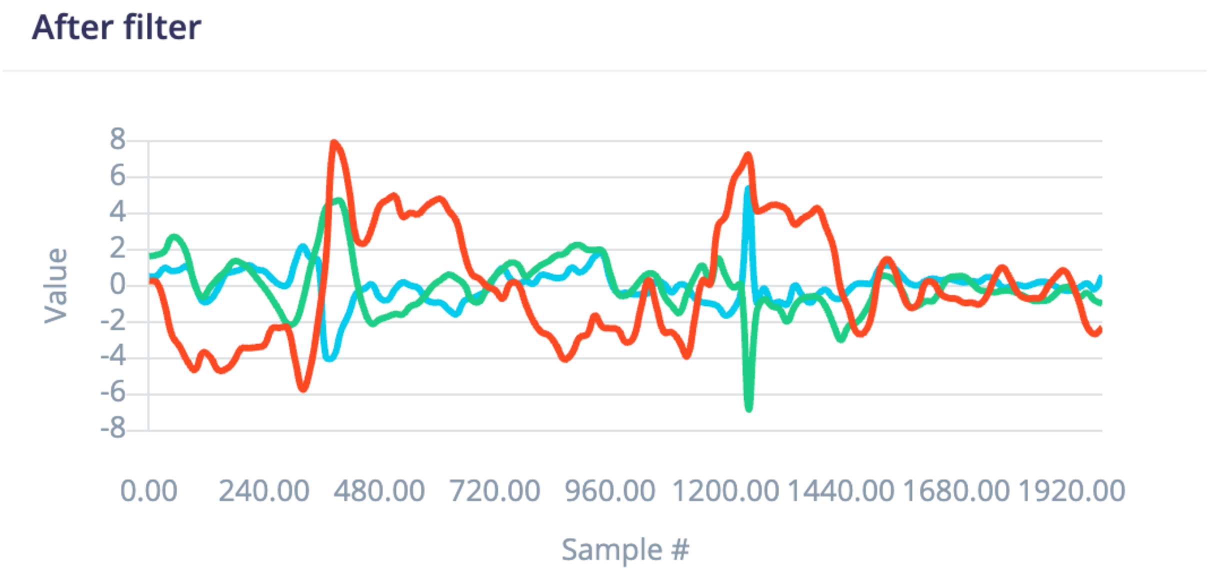


Fig: After filter – the signal after applying a low-pass filter.

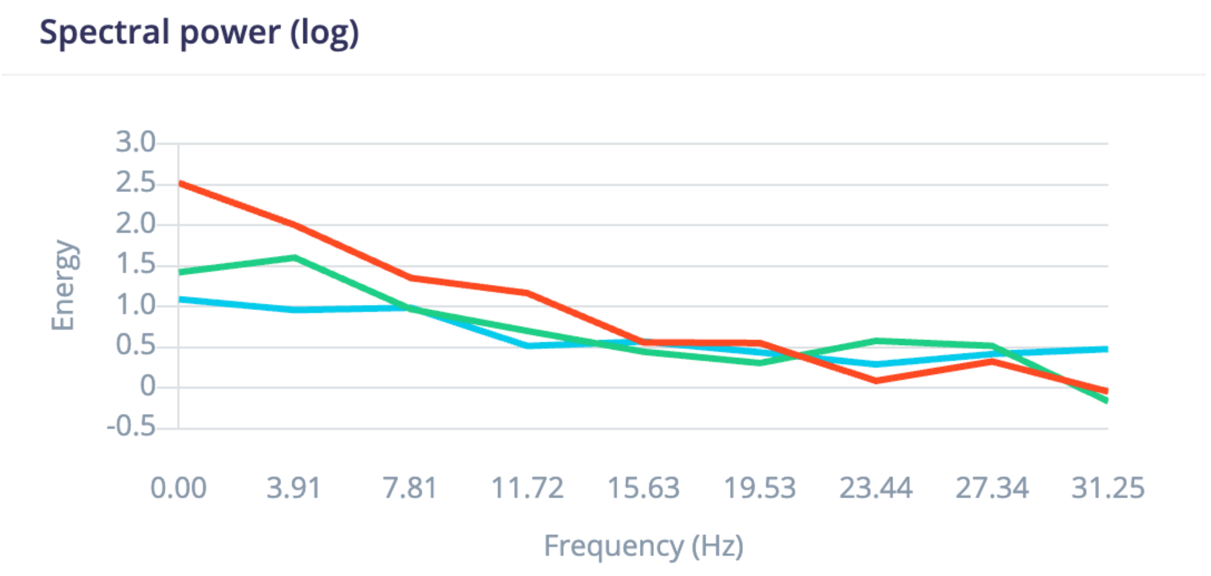


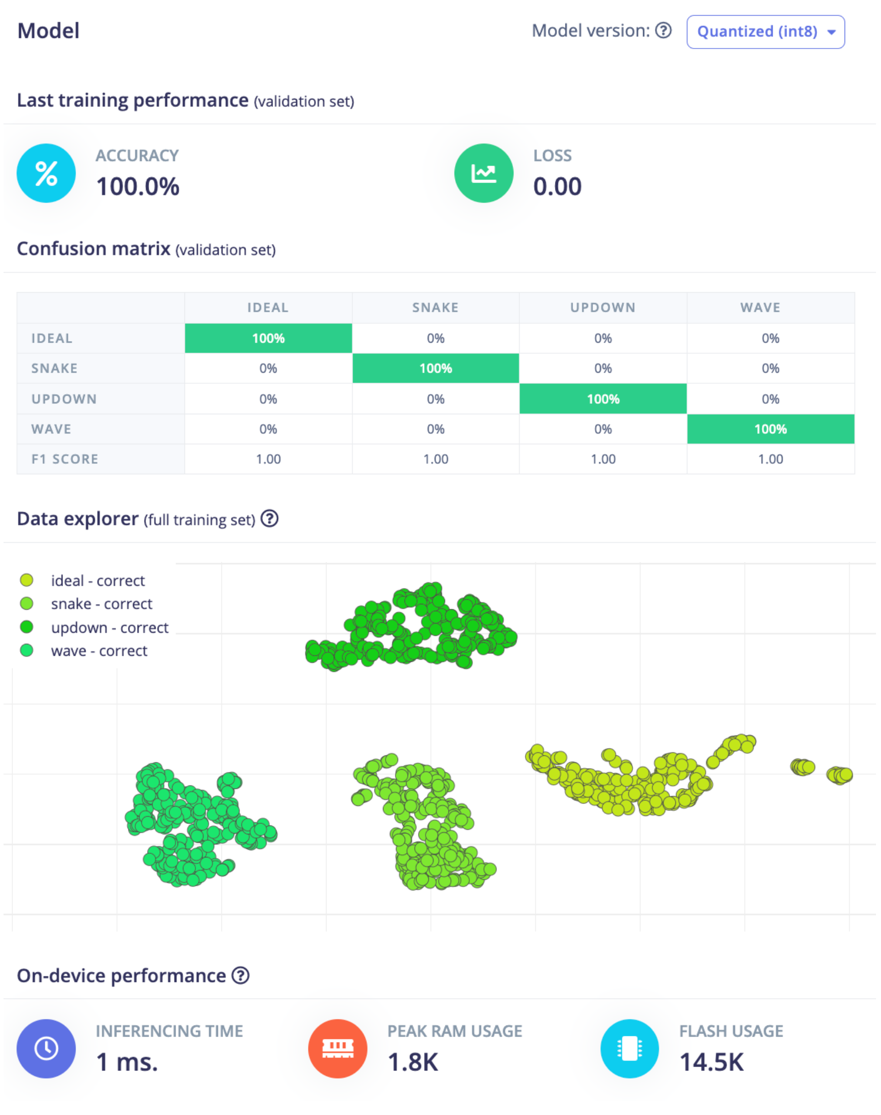
Fig: Spectral power – the amount of power that went into the signal at each frequency

<u>Configuring the Neural Network:</u>

Neural networks refer to a group of algorithms modeled after the human brain, designed to identify patterns. The network we are training for this project will take signal processing data as its input and attempt to assign it to one of the four classes.

A neural network comprises interconnected layers of neurons, each connection possessing a weight. Within the input layer, a neuron may represent the height of the first peak of the X-axis (derived from the signal processing block), whereas a neuron within the output layer might correspond to the wave class. During neural network definition, all connections are initialized randomly, resulting in random predictions made by the neural network.

During training, we utilize all the raw data to request the network to make a prediction, and subsequently make minute adjustments to the weights based on the result.



Model

Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 100.0%

LOSS 0.00

Confusion matrix (validation set)

	IDEAL	SNAKE	UPDOWN	WAVE
IDEAL	100%	0%	0%	0%
SNAKE	0%	100%	0%	0%
UPDOWN	0%	0%	100%	0%
WAVE	0%	0%	0%	100%
F1 SCORE	1.00	1.00	1.00	1.00

Data explorer (full training set)

On-device performance

INFERENCING TIME 1 ms.

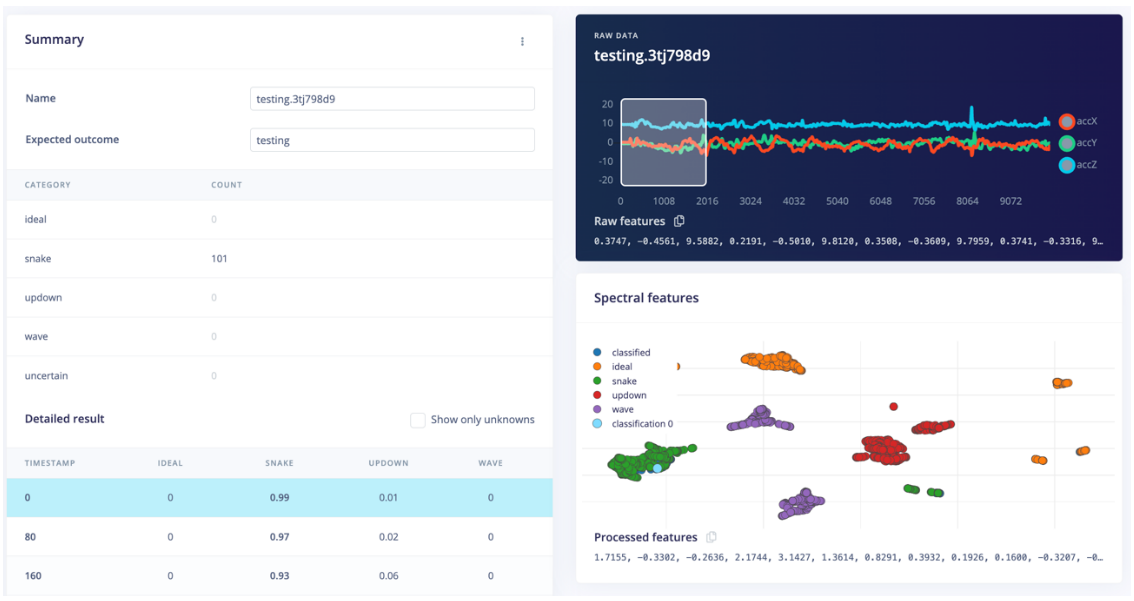
PEAK RAM USAGE 1.8K

FLASH USAGE 14.5K

Fig: Training Set

Live Classification of data:

By giving neural networks more data to learn patterns in data sets by classifying the data properly, classifying new data is fairly simple by increasing the testing data to train neural networks.



Summary

Name	testing.3tj798d9
Expected outcome	testing

CATEGORY	COUNT
ideal	0
snake	101
updown	0
wave	0
uncertain	0

Detailed result

Show only unknowns

TIMESTAMP	IDEAL	SNAKE	UPDOWN	WAVE
0	0	0.99	0.01	0
80	0	0.97	0.02	0
160	0	0.93	0.06	0

RAW DATA

testing.3tj798d9

Raw features

0.3747, -0.4561, 9.5882, 0.2191, -0.5010, 9.8120, 0.3508, -0.3609, 9.7959, 0.3741, -0.3316, 9…

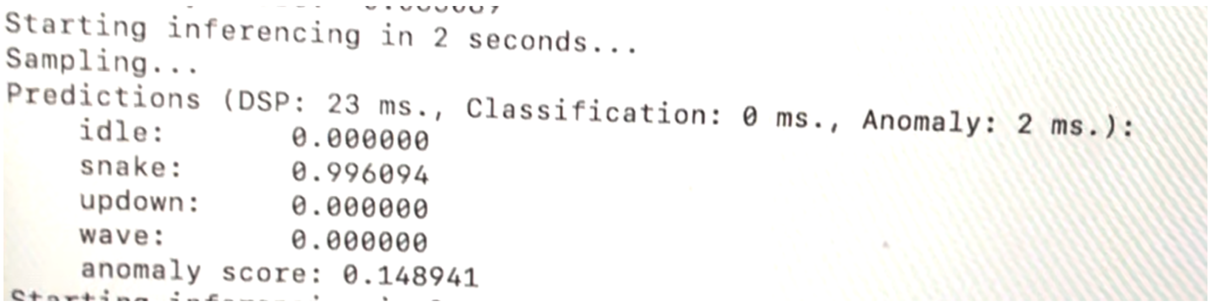
Spectral features

Processed features

1.7155, -0.3302, -0.2636, 2.1744, 3.1427, 1.3614, 0.8291, 0.3932, 0.1926, 0.1600, -0.3207, -0…

Deploying Back to Device:

With the impulse designed, trained, and verified we can deploy this model back to your device. This makes the model run without an internet connection, minimizes latency, and runs with minimum power consumption. Edge Impulse can package up the complete impulse - including the signal processing code, neural network weights, and classification code - up in a single C++ library that you can include in embedded software.



```
Starting inferencing in 2 seconds...
Sampling...
Predictions (DSP: 23 ms., Classification: 0 ms., Anomaly: 2 ms.):
    idle:       0.000000
    snake:      0.996094
    updown:     0.000000
    wave:       0.000000
    anomaly score: 0.148941
```

Fig: Sample output of Snake Detection

The output shows the prediction of the sample as the 98% probability being snake by using already neural networks which are trained by classified data that are labeled earlier.