



Electrical and Computer Engineering
Erik Jonsson School of Engineering & Computer Science
The University of Texas at Dallas

CE6302: Microprocessor and Embedded Systems

Dr. Tooraj Nikoubin

TinyML Lab 2- Audio Classification

Submitted by:

Muripa Uppaluri (mxu220008)

Chandanam Sai Nived (sxc210186)

Table of Contents

1. Project Objective	1
1.1 Introduction	1
1.2 Texas Instruments CC1352PLaunchPad	1
2. Edge Impulse	2
2.1 Sampling New Data	2
2.2 Designing an Impulse.....	2
2.3 Configuring the MFE block.....	3
2.4 Configuring the Neural Network.....	4
2.5 Classifying new data.....	5
2.6 Deploying back to device.....	5

TinyML - Audio Classification

1. Project Objective

- A brief understanding about ML algorithms
- A brief understanding about TI Launchpad (CC1352P) and Booster Sensors
- Use machine learning to build a system that can recognize when a particular sound is happening—a task known as audio classification.

1.1 Introduction

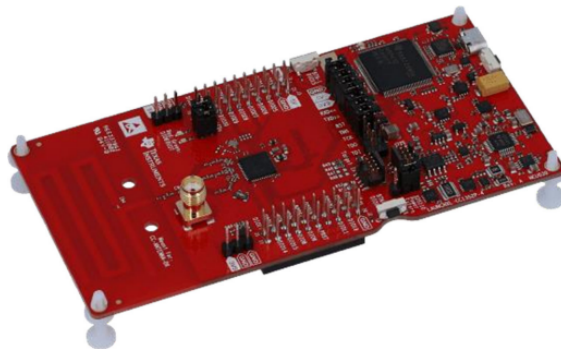
Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior. Artificial intelligence systems are used to perform complex tasks in a way that is similar to how humans solve problems.

Tiny machine learning, or TinyML, is an emerging field that is at the intersection of machine learning and embedded systems. An embedded system is a computing device that usually is small, or tiny, that operates with low power, extremely low power. So much so that some of these devices can run for days, weeks, months, sometimes even years on something like a coin cell battery. TinyML is a type of machine learning that shrinks deep learning networks to fit on tiny hardware. It brings together Artificial Intelligence and intelligent devices.

Edge computing brings computation and data storage closer to the origin of data. Majority of the edge devices that are integrated with IoT-based ecosystems are initially designed to collect sensor data and transmission of the data to neighborhood or remote cloud.

1.2 Texas Instruments CC1352P LaunchPad

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. It has Broad band antenna support for Sub-1 GHz (868 MHz / 915 MHz / 433 MHz) and 2.4 GHz frequency bands.



2. Edge Impulse

It is a cloud service for developing machine learning models in the TinyML targeted edge devices. This supports AutoML processing for edge platforms. It also supports several boards including smart phones to deploy learning models in such devices.

2.1 Sampling New Data

Machine learning works best with lots of data, so a single sample won't cut it. Now is the time to start building your own dataset. The device will capture a second of audio and transmit it to Edge Impulse:

- Noise - 5 minutes of background noise
- Faucet - 5 minutes of running faucet noise

2.2 Designing an Impulse

With the training set in place, an impulse is designed. An impulse takes the raw data, slices it up in smaller windows, uses signal processing blocks to extract features, and then uses a learning block to classify new data. Signal processing blocks always return the same values for the same input and are used to make raw data easier to process, while learning blocks learn from past experiences.

For this project we'll use the 'Spectral analysis' signal processing block. This block applies a filter, performs spectral analysis on the signal, and extracts frequency and spectral power data. Then we'll use a 'Neural Network' learning block, that takes these spectral features and learns to distinguish between the two (Noise, Faucet) classes.

The screenshot shows the Edge Impulse web interface with the following configuration:

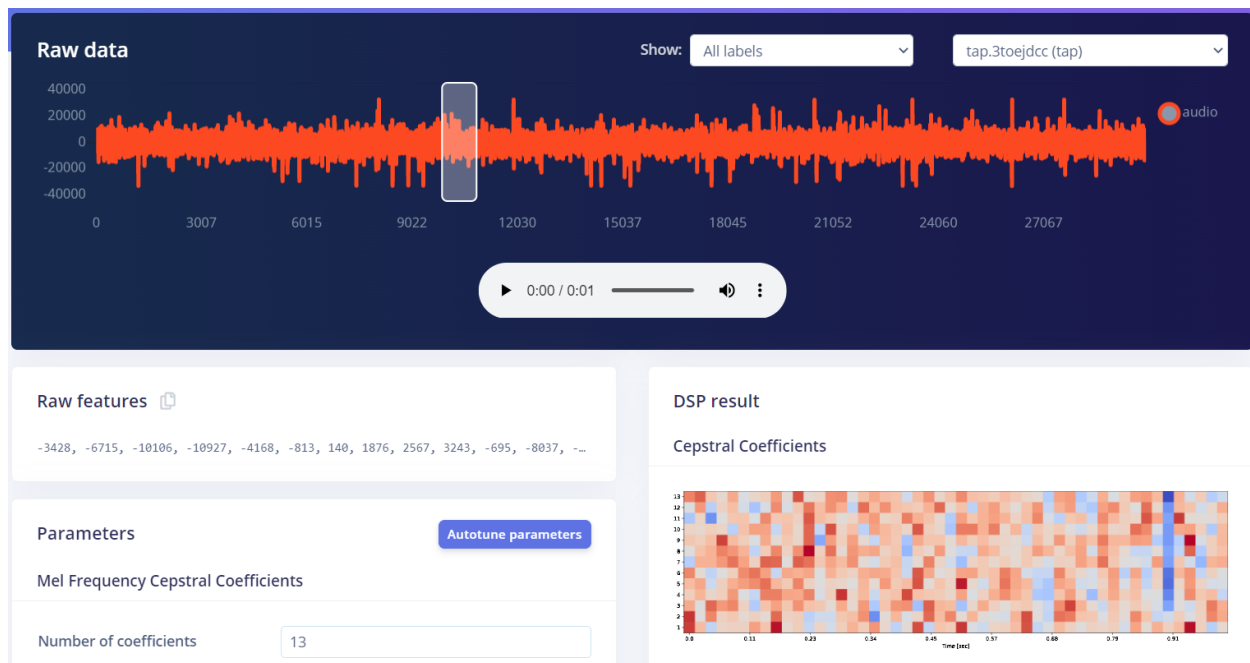
- Time series data (Red Panel):**
 - Input axes: audio
 - Window size: 1000 ms.
 - Window increase: 100 ms.
 - Frequency (Hz): 16000
 - Zero-pad data: ☒
- Audio (MFCC) (White Panel):**
 - Name: MFCC
 - Input axes (1): ☒ audio
- Classification (Purple Panel):**
 - Name: Classifier
 - Input features: ☒ MFCC
 - Output features: 2 (noise, tap)

At the bottom, there are two dashed boxes for adding more blocks:

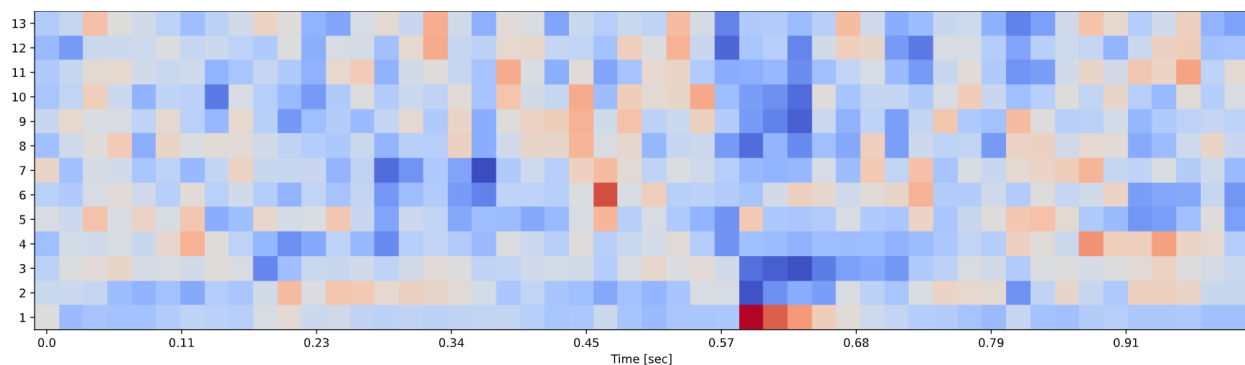
- Add a processing block** (lightning bolt icon)
- Add a learning block** (flask icon)

2.3 Configuring the MFE block

The MFE block transforms a window of audio into a table of data where each row represents a range of frequencies, and each column represents a span of time. The value contained within each cell reflects the amplitude of its associated range of frequencies during that span of time. The spectrogram shows each cell as a colored block, the intensity which varies depending on the amplitude.



The patterns visible in a spectrogram contain information about what type of sound it represents. For example, the spectrogram in this image shows a pattern typical of background noise:

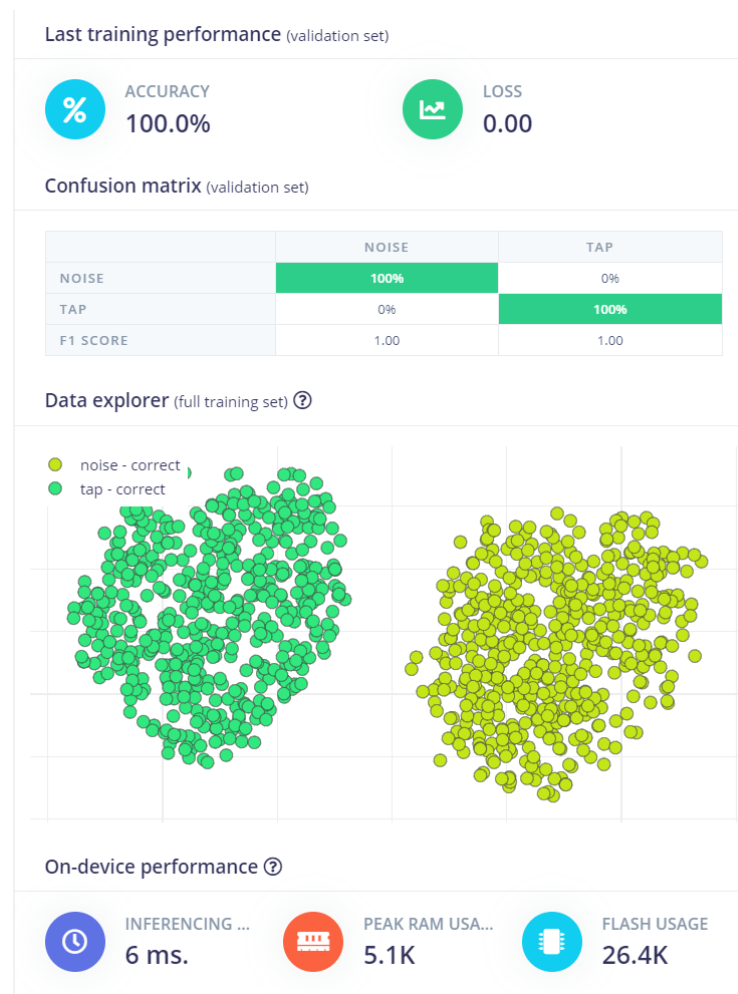


2.4 Configuring the Neural Network

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. The network that we're training here will take the signal processing data as an input, and try to map this to one of the four classes.

A neural network consists of layers of neurons, all interconnected, and each connection has a weight. One such neuron in the input layer would be the height of the first peak of the X-axis (from the signal processing block); and one such neuron in the output layer would be wave (one of the classes). When defining the neural network all these connections are initialized randomly, and thus the neural network will make random predictions.

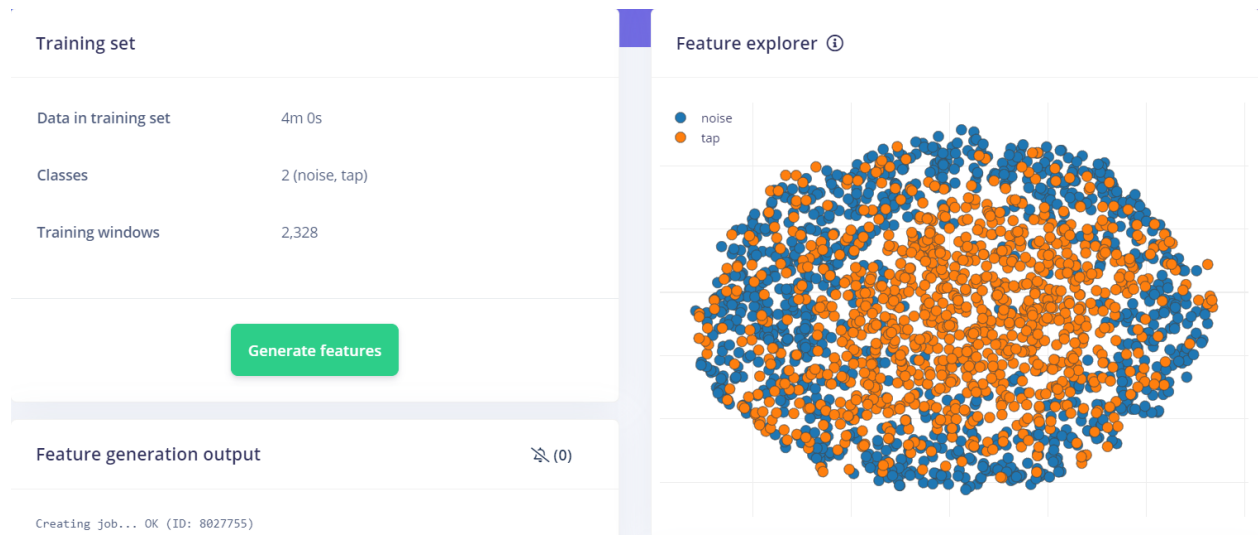
During training we then take all the raw data, ask the network to make a prediction, and then make tiny alterations to the weights depending on the outcome (this is why labeling raw data is important). This way, after a lot of iterations, the neural network learns; and will eventually become much better at predicting new data.



Full Training Set

2.5 Classifying New 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.



2.6 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.

```
Predictions (DSP: 142 ms., Classification: 8 ms., Anomaly: 0 ms.):
  noise:    0.996094
  tap:      0.000000
Predictions (DSP: 141 ms., Classification: 8 ms., Anomaly: 0 ms.):
  noise:    0.195312
  tap:      0.804688
Predictions (DSP: 141 ms., Classification: 8 ms., Anomaly: 0 ms.):
  noise:    0.000000
  tap:      0.996094
```

The output shows the prediction of the sample as the 99% probability of being tap (fauchet) by using already neural networks which are trained by classified data that are labeled earlier.