TRAINING/DEPLOYING A GESTURE DETECTION MODEL

In this lab session, you'll use machine learning to build a gesture recognition system that runs on a TinyML microcontroller. This is a challenging task to solve using rule-based programming, as people don't perform gestures in the same way every time. However, machine learning can handle these variations with ease. You'll learn how to collect high-frequency data from actual sensors, use signal processing to clean up data, build a neural network classifier, and deploy your model back to a device. At the end of this tutorial, you'll fully understand how to apply machine learning in embedded devices using Edge Impulse and Arduino Nano BLE 33.

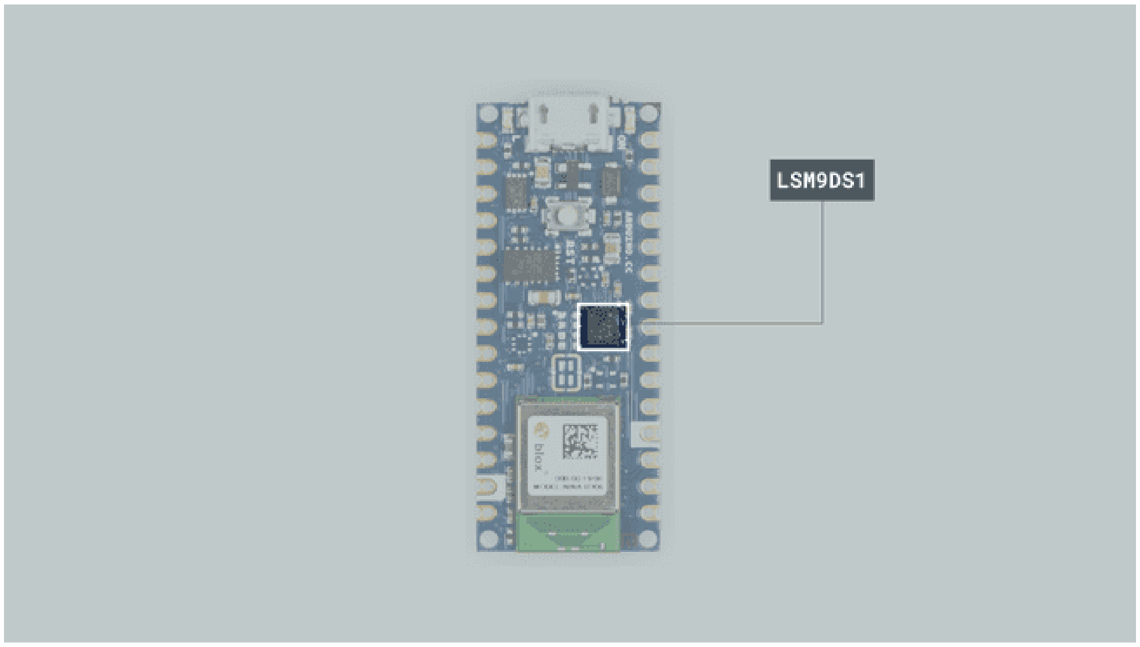
HAND GESTURE DETECTION

This example shows how to run a simple feedforward neural network to recognize three hand gestures. “up-down”, “left-right” and “push-pull”. Using Arduino Nano BLE 33 IMU sensors (i.e. accelerometer, gyroscope, and magnetometer) we will do the following:

- ✓ Collect data from the device’s IMU sensors
- ✓ Create training and testing dataset
- ✓ Through Edge Impulse execute data pre-processing and feature extraction
- ✓ Build & Train our machine-learning model
- ✓ Test our model with live data.

LSM9DS1 INERTIAL MODULE

IMU stands for: inertial measurement unit. It is an electronic device that measures and reports a body's specific force, angular rate and the orientation of the body, using a combination of accelerometers, gyroscopes, and oftentimes magnetometers. In this tutorial we will learn about the LSM9DS1 IMU module, which is included in the Arduino Nano 33 BLE Board.



The LSM9DS1 Sensor Module

The LSM9DS1 is a system-in-package featuring a 3D digital linear acceleration sensor, a 3D digital angular rate sensor, and a 3D digital magnetic sensor.

The LSM9DS1 Library

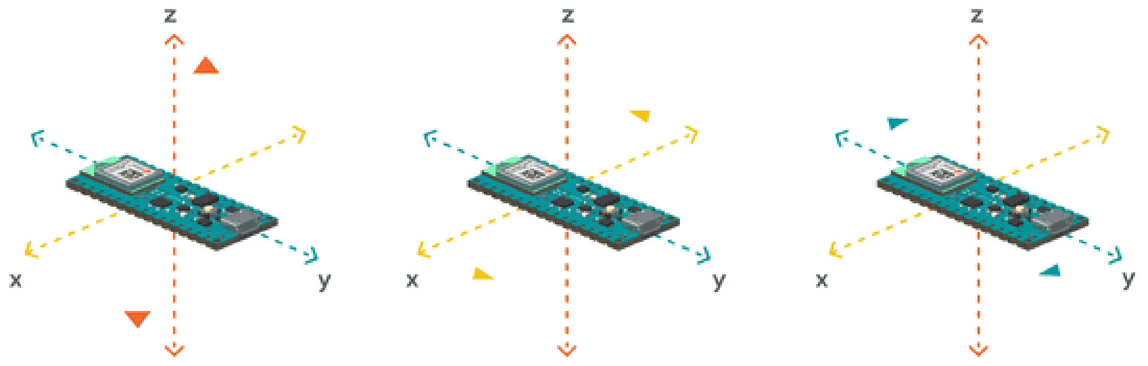
The Arduino LSM9DS1 library allows us to use the Arduino Nano 33 BLE IMU module without having to go into complicated programming. The library takes care of the sensor initialization and sets its values as follows:

- **Accelerometer** range is set at [-4, +4]g -/+0.122 mg.
- **Gyroscope** range is set at [-2000, +2000] dps +/-70 mdps.
- **Magnetometer** range is set at [-400, +400] uT +/-0.014 uT.
- **Accelerometer** output data rate is fixed at 104 Hz.
- **Gyroscope** output data rate is fixed at 104 Hz.
- **Magnetometer** output data rate is fixed at 20 Hz.

If you want to read more about the LSM9DS1 sensor module see here.

ACCELEROMETER

An accelerometer is an electromechanical device used to measure acceleration forces. Such forces may be static, like the continuous force of gravity or, as is the case with many mobile devices, dynamic to sense movement or vibrations.

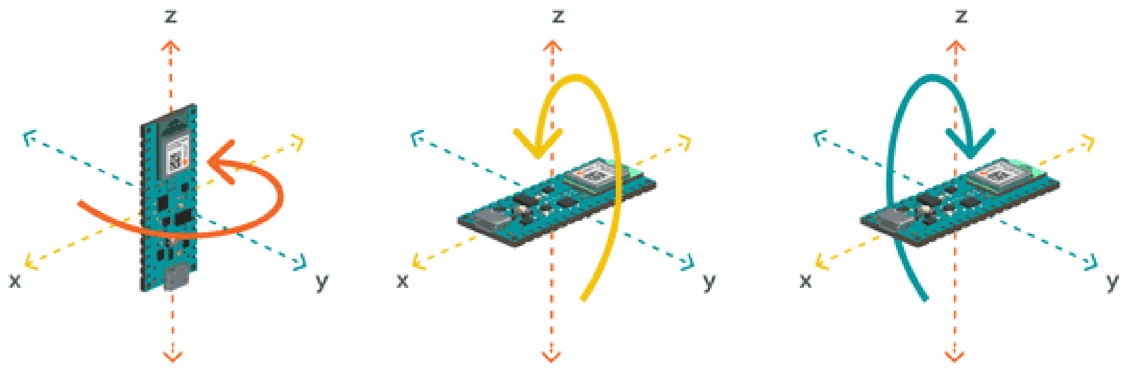


How the accelerometer works.

In this example, we will use the accelerometer as a "level" that will provide information about the position of the board. With this application we will be able to visualize what the relative position of the board is as well as the degrees, by tilting the board up, down, left or right.

GYROSCOPE

A gyroscope sensor is a device that can measure and maintain the orientation and angular velocity of an object. Gyroscopes are more advanced than accelerometers, as they can measure the tilt and lateral orientation of an object, whereas an accelerometer can only measure its linear motion.

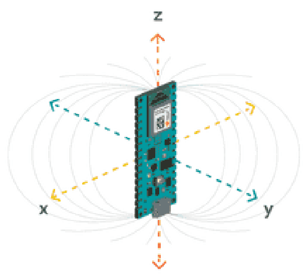


How gyroscopes work.

Gyroscope sensors are called "Angular Rate Sensors" or "Angular Velocity Sensors". Measured in degrees per second, angular velocity is the change in the rotational angle of the object per unit of time. In this example, we will use the gyroscope to indicate the direction of the force applied to the board. This will be achieved by swiftly moving the board instantly in four directions: forward, backward, to the left, and to the right. The results will be visible through the Edge Impulse Data Acquisition tab.

MAGNETOMETER

A magnetometer is a device that measures magnetism, that is, the direction, strength, or relative change of a magnetic field at a particular location. We will not use this specific sensor for our example, but you can take an in-depth look at it through the Arduino IDE and use it as it best fits in your future projects. **Arduino IDE> Example> Arduino_LSM9DS1 > Simplemagnetometer**



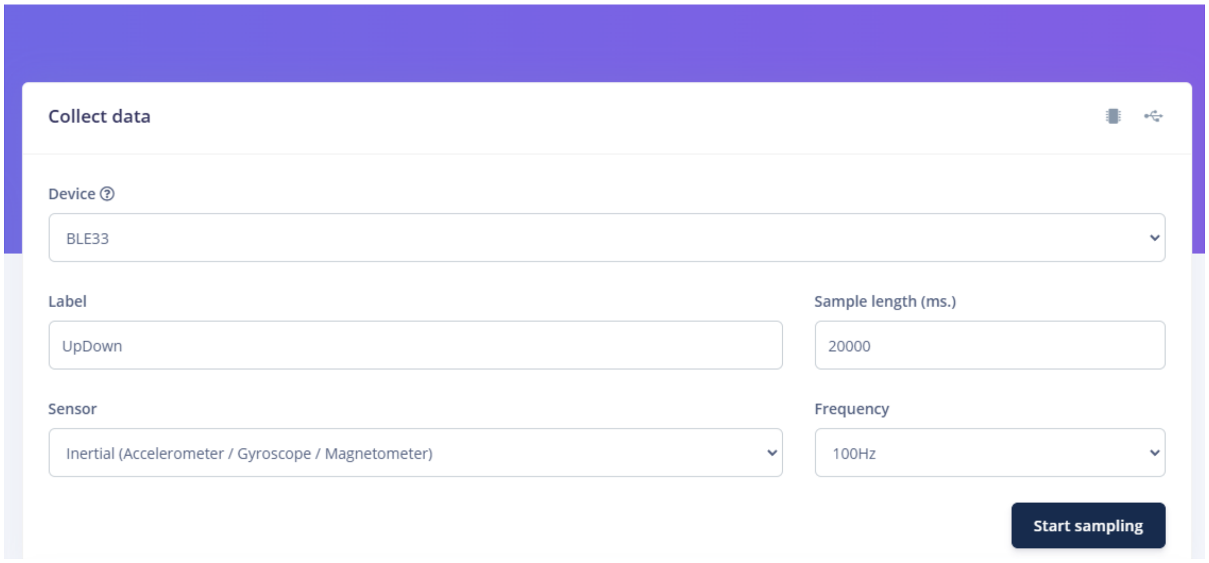
How Magnetometer works.

Setting Up your Project

1. Create a new project repository at **Edge Impulse**
2. Connect your **Arduino Nano BLE 33** device through your terminal
3. You would need to clean your device connection from the previous project directory and reconnect it into your new project repository by typing `edge-impulse-daemon -clean`
4. Enter your email address and password
5. Connect your device to the correct project repository
6. After your device is successfully connected to the correct project you will go to the edge impulse tab and do the following

Collecting Data

1. With your device connected, we can collect some data. In the studio, we go to the **Data acquisition** tab. This is where all your raw data is stored and where you can start sampling new data.
2. Under "**Collect Data**", select your device and set the label name "**Up-Down**".
3. Set the sample to the 2000s (it will record 20s of data), which is *optional if you want to play with the sample rate.*
4. Keep frequency to 10Hz *(optional if you want to play with the frequency rate as well)*
5. Select "**Inertial (Accelerometer/Gyroscope/Magnetometer)**" in the Sensor section
6. Record your hand movements for "*UpDown*" by moving your microcontroller up and down and after each move pausing for 2 seconds.



Collect data

Device: BLE33

Label: UpDown

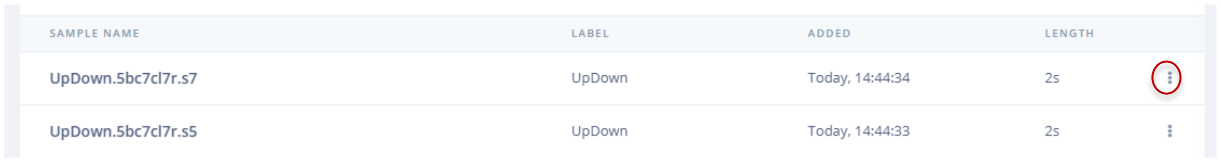
Sample length (ms.): 20000

Sensor: Inertial (Accelerometer / Gyroscope / Magnetometer)

Frequency: 100Hz

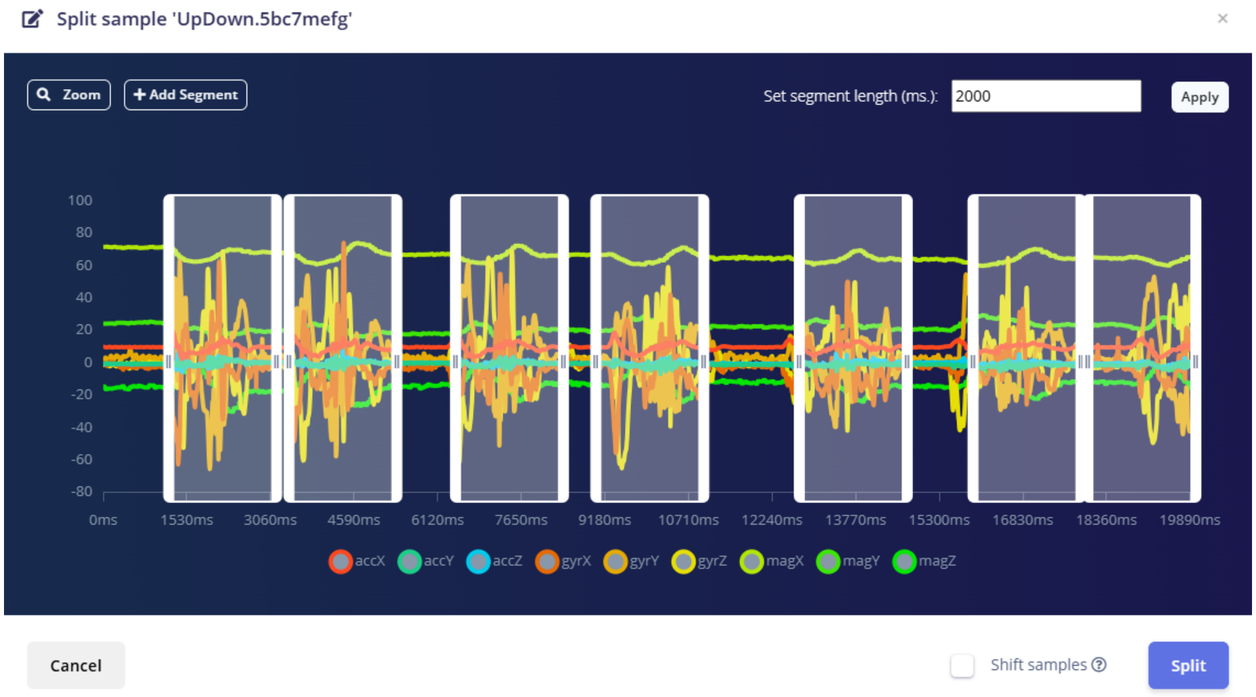
Collect data tab

7. It will automatically generate 20s of samples. Click on the "3-dot" menu and split the data in 2s. Adjust if necessary. Shown below



SAMPLE NAME	LABEL	ADDED	LENGTH
UpDown.5bc7cl7r.s7	UpDown	Today, 14:44:34	2s
UpDown.5bc7cl7r.s5	UpDown	Today, 14:44:33	2s

Dataset tab

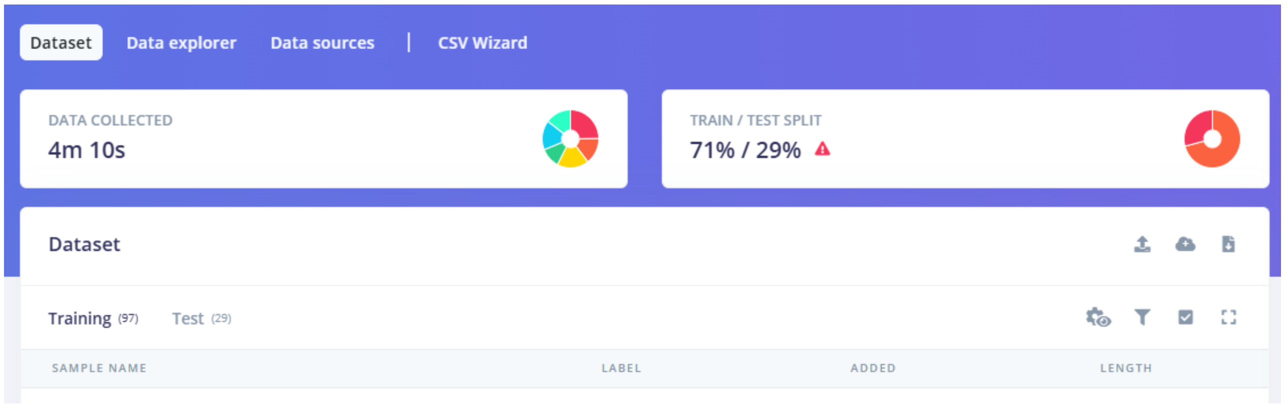


Split sample 'UpDown.5bc7mefg'

Set segment length (ms.): 2000

Data split sample pop up tab (select 2000 for 2s split on the "set segment length ms tab)

8. Repeat the same procedure for "**PushPull**", "**LeftRight**".
9. Sample enough data to create a robust dataset (minimum 60 sec on each labeled data).
10. Split your dataset into at least 70%/30% range or 80%/20% (train/test split)



DATA COLLECTED 4m 10s

TRAIN / TEST SPLIT 71% / 29%

Dataset

Training (97) Test (29)

SAMPLE NAME	LABEL	ADDED	LENGTH

Data acquisition tab

EDGE IMPULSE

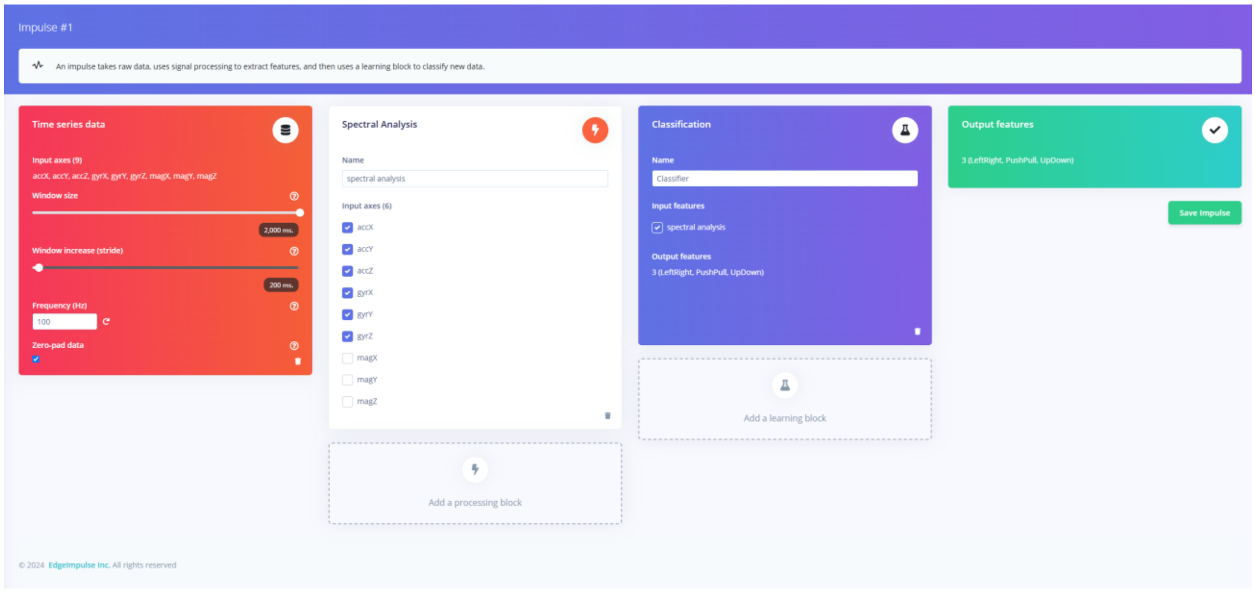
1. After successfully collecting your data go to "**create impulse**" tab and keep time series data as is
2. Select **spectral analysis** for this example.

Spectral analysis is a technique that uses frequencies to analyze the content of a signal

3. Select accelerometer and gyroscope data and uncheck magnetometer data.
4. Add "**classifier**" option as the "**learning block**" section
5. Make sure your output features are selected properly. (Uou should have 3 output features: PushPull, UpDown, LeftRight)

NOTE: Sometimes labeling your data incorrectly can mistreat it as another output feature.

6. Your "**create impulse**" tab looks similar to the figure given below then click on "**save impulse**"


Impulse #1

An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.

Time series data
Input axes (9)
accX, accY, accZ, gyrX, gyrY, gyrZ, magX, magY, magZ
Window size: 2,000 ms.
Window increase (stride): 200 ms.
Frequency (Hz): 100
Zero-pad data: ✓

Spectral Analysis
Name: spectral analysis
Input axes (6)
✓ accX
✓ accY
✓ accZ
✓ gyrX
✓ gyrY
✓ gyrZ
☐ magX
☐ magY
☐ magZ

Classification
Name: Classifier
Input features
✓ spectral analysis
Output features
3 (LeftRight, PushPull, UpDown)

Output features
3 (LeftRight, PushPull, UpDown)

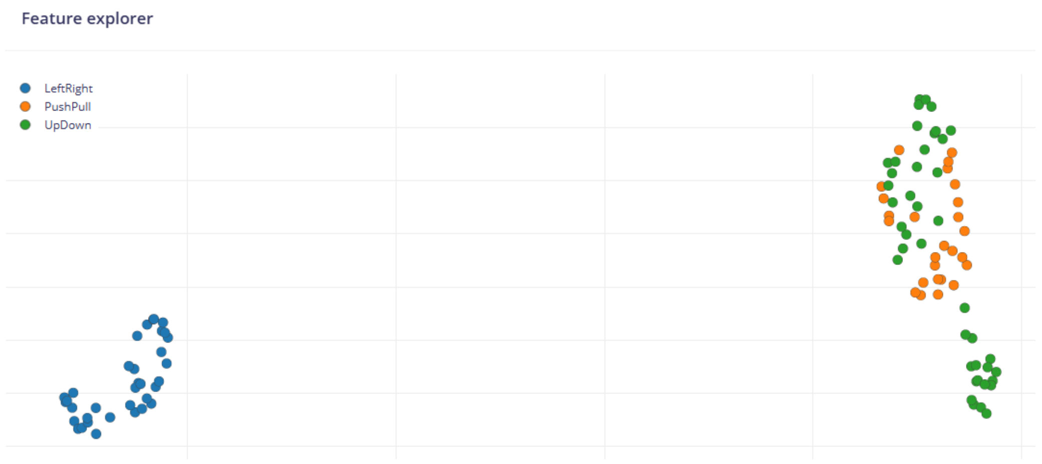
Edge Impulse "Create Impulse" Dashboard after selecting necessary blocks.

SPECTRAL ANALYSIS

In sensor data, especially from Inertial Measurement Units (IMUs), **spectral features** refer to characteristics derived from the frequency domain of the data. IMU sensors, which measure parameters like acceleration, angular velocity, and sometimes magnetometer readings, generate time-series data. To analyze this data, converting it from the time domain to the frequency domain (using techniques like Fourier Transform) can reveal useful patterns that are not easily visible in the time series.

1. Go to **Impulse Design > Spectral Analysis**
2. You can check out each dataset and their corresponding image. Make necessary tweaks if needed.
3. Then Click on the "**save parameters**" button
4. It will prompt you to "**Generate Feature**" tab Click on "**Generate Feature**" button
5. The generate feature tab will take in your gesture datasets and generate your spectral analysis graph
6. After proper generation of data it will prompt you to "**Feature Explorer**" tab which will give you a UMAP of your dataset samples.

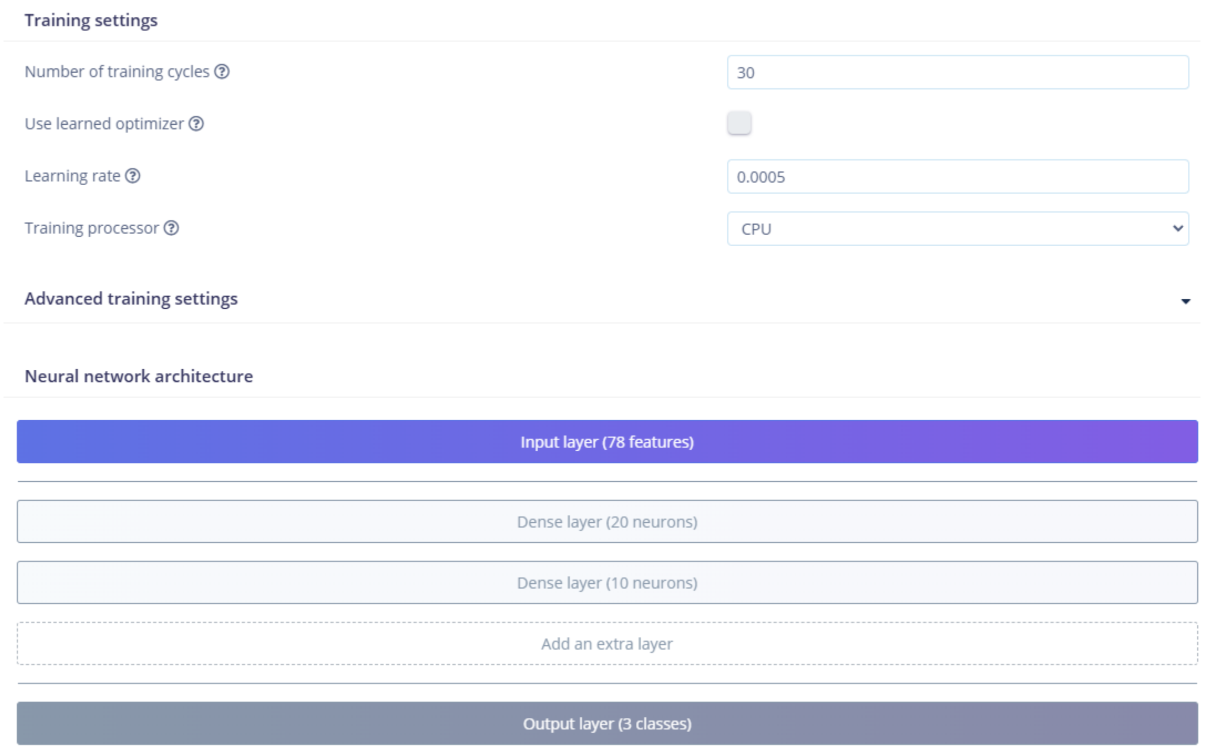
7. This will be easy to identify whether your distribution is grouped and distributed correctly.



The actual features will be generated. Using UMAP, a dimension reduction technique, the Feature explorer shows how the features are distributed on a two-dimensional plot.

CLASSIFIER

1. Go to **Classifier** tab
2. You can mark your Neural Network training either in block level or expert mode
3. For the first training work with block level, you can train the following network and adjust accuracy based on it



Training settings

Number of training cycles	30
Use learned optimizer	☐
Learning rate	0.0005
Training processor	CPU

Advanced training settings

Neural network architecture

Input layer (78 features)

Dense layer (20 neurons)

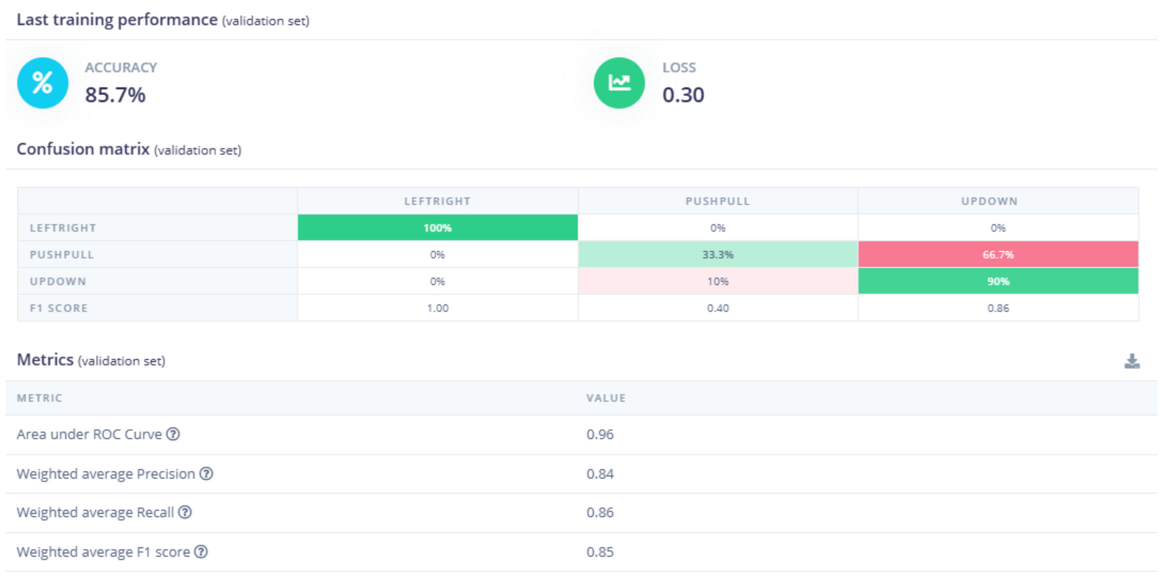
Dense layer (10 neurons)

Add an extra layer

Output layer (3 classes)

Neural Network Setting Block

4. Click on the **Save & Train** button
5. In the training output you are able to see your loss, accuracy and val_accuracy in the "**training output**" tab over each epoch.
6. After training you can check your confusion matrix in the "**Model**" tab



Last training performance (validation set)

ACCURACY 85.7%

LOSS 0.30

Confusion matrix (validation set)

	LEFTRIGHT	PUSHPULL	UPDOWN
LEFTRIGHT	100%	0%	0%
PUSHPULL	0%	33.3%	66.7%
UPDOWN	0%	10%	90%
F1 SCORE	1.00	0.40	0.86

Metrics (validation set)

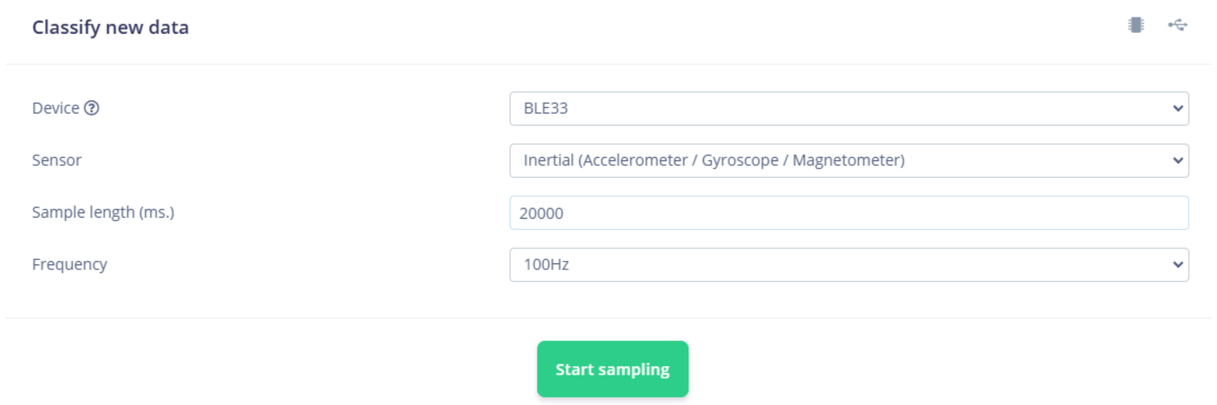
METRIC	VALUE
Area under ROC Curve	0.96
Weighted average Precision	0.84
Weighted average Recall	0.86
Weighted average F1 score	0.85

Model evaluation tab

7. From the model tab it is seen that the "PushPull" feature mismatched as "UpDown" gesture. Thus, adding more separate and clean data between those two gestures can improve the validation accuracy.

Live Classification

1. Go to **Live classification** tab
2. In the **classify new data** section upload real-time gesture data to test your model
3. You can adjust your sample length to 10000ms, 20000ms etc.
4. Click on the "**Start sampling**" button

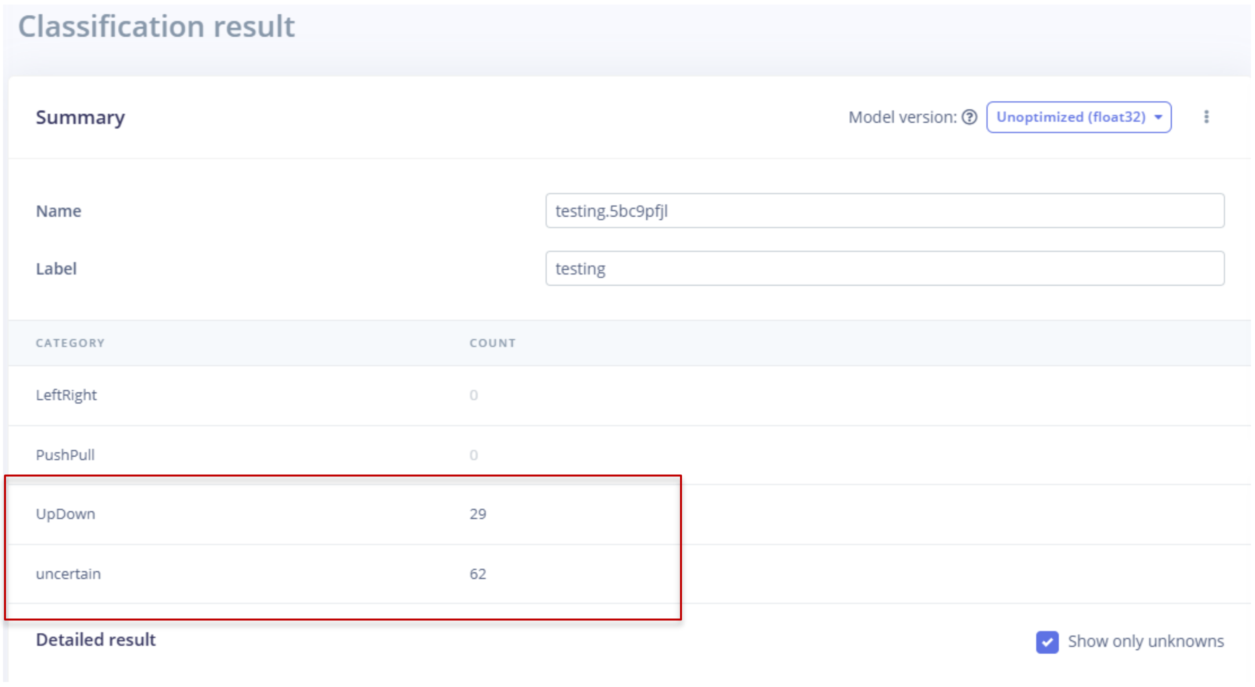


Classify new data

Device	BLE33
Sensor	Inertial (Accelerometer / Gyroscope / Magnetometer)
Sample length (ms.)	20000
Frequency	100Hz

Classify new data tab

5. It should start collecting live data from the IMU sensor of your Nano BLE 33 board.
6. The classification result tab will generate your testing data and your trained NN model will try to classify it based on the three output features was trained on.



Classification result

Summary — Model version: Unoptimized (float32)

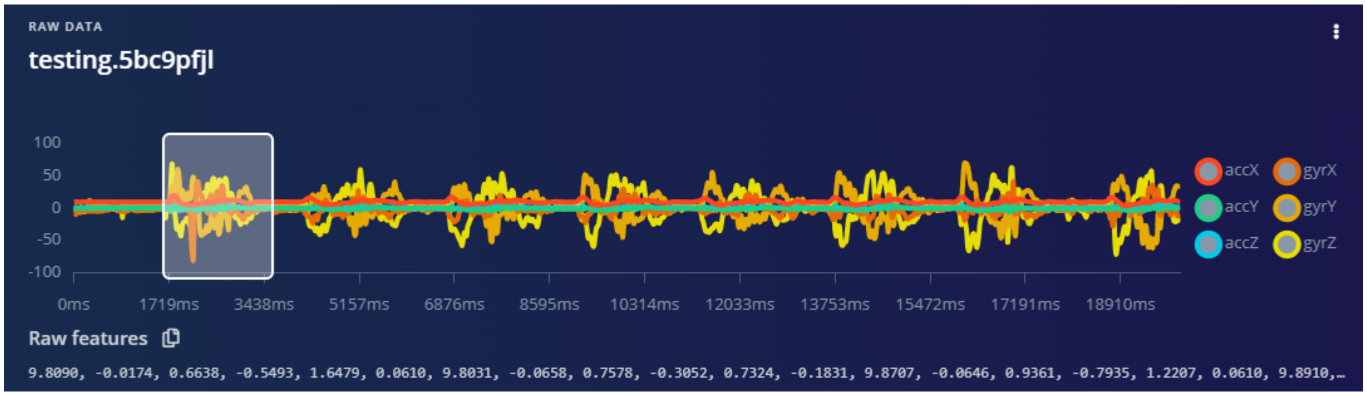
Name	testing.5bc9pfjl
Label	testing

CATEGORY	COUNT
LeftRight	0
PushPull	0
UpDown	29
uncertain	62

Detailed result — Show only unknowns ✓

Classification Result Tab

7. My gesture for 20s was a mixture of **Up-Down** and straight thus it detected "UpDown" gesture with 29 confidence and was uncertain with other gestures.
8. Below is my live gesture data collected for 20s



```
9.8090, -0.0174, 0.6638, -0.5493, 1.6479, 0.0610, 9.8031, -0.0658, 0.7578, -0.3052, 0.7324, -0.1831, 9.8707, -0.0646, 0.9361, -0.7935, 1.2207, 0.0610, 9.8910,…
```

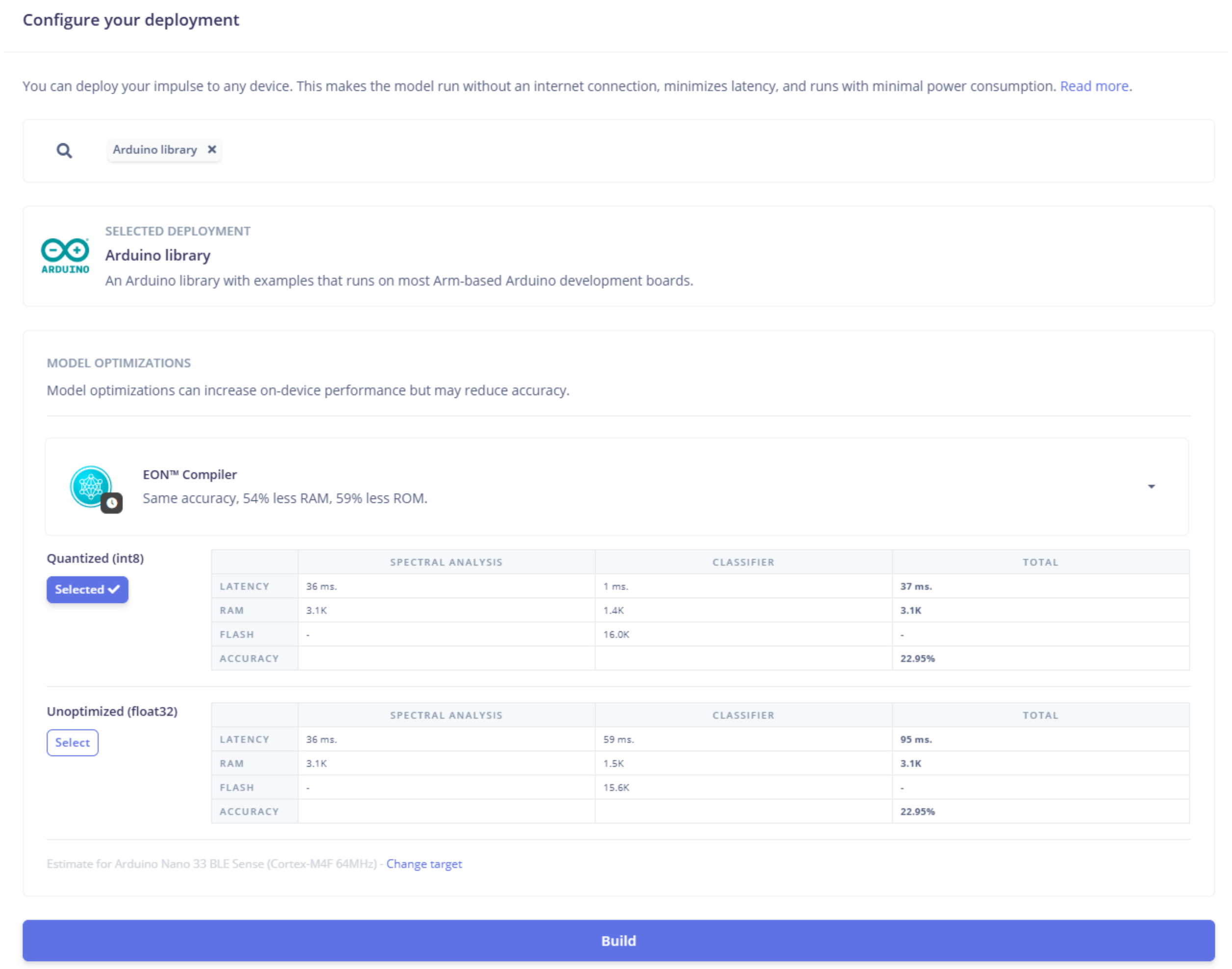
Raw data and features collected from live demonstration

Optional: Extracting Your Quantized MODEL and Testing it on Arduino IDE

For this section you must train your model with accelerometer data or any single IMU sensor.

- Go to Create Impulse> Spectral Analysis and select accelerometer datas for x,y,z axis
- Same impulse

- Train and save your classifier
- Go to Deployment section and in the "**configure your deployment**" tab select "**Arduino library**"
- Select "quantized (int8) option then click on **Build.** This should build your classifier model into an Arduino library



Configure your deployment

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Read more.

Arduino library ✕

SELECTED DEPLOYMENT
Arduino library
An Arduino library with examples that runs on most Arm-based Arduino development boards.

MODEL OPTIMIZATIONS
Model optimizations can increase on-device performance but may reduce accuracy.

EON™ Compiler
Same accuracy, 54% less RAM, 59% less ROM.

Quantized (int8) — Selected ✓

	SPECTRAL ANALYSIS	CLASSIFIER	TOTAL
LATENCY	36 ms.	1 ms.	37 ms.
RAM	3.1K	1.4K	3.1K
FLASH	-	16.0K	-
ACCURACY			22.95%

Unoptimized (float32) — Select

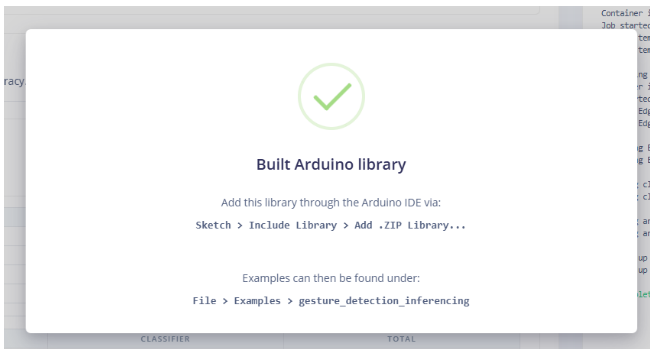
	SPECTRAL ANALYSIS	CLASSIFIER	TOTAL
LATENCY	36 ms.	59 ms.	95 ms.
RAM	3.1K	1.5K	3.1K
FLASH	-	15.6K	-
ACCURACY			22.95%

Estimate for Arduino Nano 33 BLE Sense (Cortex-M4F 64MHz) - Change target

Build

Configure your deployment tab

- A .zip folder should be prepared and downloaded to your computer.



Built Arduino library

Add this library through the Arduino IDE via:

```
Sketch > Include Library > Add .ZIP Library...
```

Examples can then be found under:

```
File > Examples > gesture_detection_inferencing
```

Build successfully

- Go to Arduino IDE
- Select **Sketch > include library > add .ZIP folder**. *(if you have existing edge impulse gesture library make sure you have deleted it)*
- Go to **File> Example > gesture_detection_inferencing>your device name> nano_ble33_sense_accelerometer_continuous**
- Make sure the correct board is selected and your device is connected to the COM port
- Select Upload (it will take some time)
- After successful compilation, go to Serial Monitor, and you should be able to see your model output.

REPORT

1. Document your model architecture and its result by using your own collected data from the Nano BLE 33 IMU sensor and generating three output features: “UpDown,” “LeftRight,” and “PushPull.” Try to achieve accuracy over 82% and document your classifier results.
2. Document the feature explorer section and include the generated graph after testing.
3. Execute **live demonstration** and include your confidence point from the “**live classification/Classification Result Tab**”
4. Document the “**on device**” section after training to see your inference time, memory utilization, and peak performance.

HELPFUL LINKS TO GET YOU STARTED

1. 10 Gesture-Based Control Using Arduino Nano 33 BLE Sense | TinyML | Edge Impulse- Roles Academy

Happy Inference!