Preparation:

- 1. Get Started with Wio Terminal
- 2. Connect Wio Terminal with Edge Impulse
- 3. Add some necessary Arduino libraries

Get Started with Wio Terminal

Step 1. You need to Install <u>Arduino Software</u>.

Download Arduino IDE

Launch the Arduino application

Double-click the Arduino IDE application you have previously downloaded.

Step 2. Add the Wio Terminal Board Library

 Open your Arduino IDE, click on File > Preferences, and copy the below URL to Additional Boards Manager URLs:

https://files.seeedstudio.com/arduino/package_seeeduino_boards_index.j
son

		Pref	ferences		
		Settings	Network		
Sketchbook location:					
/Users/ansonhe/Documents,	Arduino				Browse
Editor language:	English (English	1)	\$	(requires restar	t of Arduino)
Editor font size:	12				
Interface scale:	✓ Automatic	100 🗘 %	(requires restart of Arduino)		
Theme:	Default theme	ᅌ (red	quires restart of Arduino)		
Show verbose output during:	compilation	🗸 uploa	d		
Compiler warnings:	None ᅌ				
🗸 Display line numbers			Enable Code Folding		
🗹 Verify code after upload			Use external editor		
Check for updates on star	tup		Save when verifying or upl	oading	
Use accessibility features					
Additional Boards Manager UR	Ls: ttps://files.s	eeedstudio	o.com/arduino/package_seeeduir	no_boards_index.j	json 🔲
More preferences can be edite	d directly in the fil	e			
/Users/ansonhe/Library/Ardu	ino15/preferences	.txt			
(edit only when Arduino is not	running)				
				ОК	Cancel

2. Click on **Tools > Board > Board Manager** and Search **Wio Terminal** in the Boards Manager.

/pe Al	II I	Wio Terminal			
pe Ai					
Seeed SA	AMD Boards				
by Seeed	Studio version 1.6.0	INSTALLED			
	cluded in this package:			· · · · · · · · · · · · · · · · · · ·	
Online Hel		emto MU, Seeeduino Zero, S	Seeeduino LoRaWAN, Wio GPS Board, S	seeeduino Wio Terminal.	
More Info					
Coloct	t version ᅌ	Install		Update	Remove
Select		mstan		Opuate	Kelliove
	no_SAMD_zero				
by Seeed Boards inc Seeeduino	I Studio version 1.0.0 I cluded in this package: o_Lotus_M0, Seeeduino_	INSTALLED _Cortex_M0+, Seeeduino_Wi	io_Lite_W600.		
by Seeed Boards inc Seeeduino Online Hel	I Studio version 1.0.0 I cluded in this package: o_Lotus_M0, Seeeduino_ clp		io_Lite_W600.		
by Seeed Boards inc Seeeduino	I Studio version 1.0.0 I cluded in this package: o_Lotus_M0, Seeeduino_ clp		io_Lite_W600.		
by Seeed Boards inc Seeeduino Online Hel	I Studio version 1.0.0 I cluded in this package: o_Lotus_M0, Seeeduino_ clp		io_Lite_W600.		
by Seeed Boards inc Seeeduino Online Hel	I Studio version 1.0.0 I cluded in this package: o_Lotus_M0, Seeeduino_ clp		io_Lite_W600.		
by Seeed Boards inc Seeeduino Online Hel	I Studio version 1.0.0 I cluded in this package: o_Lotus_M0, Seeeduino_ clp		io_Lite_W600.		
by Seeed Boards inc Seeeduino Online Hel	I Studio version 1.0.0 I cluded in this package: o_Lotus_M0, Seeeduino_ clp		io_Lite_W600.		
by Seeed Boards inc Seeeduino Online Hel	I Studio version 1.0.0 I cluded in this package: o_Lotus_M0, Seeeduino_ clp		lo_Lite_W600.		

The keyword is Wio Terminal

Step 3. Select your board and port

You'll need to select the entry in the **Tools > Board** menu that corresponds to your Arduino. Selecting the **Wio Terminal**.

Arduino File Edit Sketch	Tools Help		
 ✓ ● ✓ ●	Auto Format Archive Sketch Fix Encoding & Reload Manage Libraries Serial Monitor Serial Plotter	策T ① 郑 I ① 郑 M ① 郑 L	Arduino Uno Arduino Duemilanove or Diecimila Arduino Nano Arduino Mega or Mega 2560 Arduino Mega ADK
 Turns an LED on for one second, Most Arduinos have an on-board L it is attached to digital pin 13 the correct LED pin independent If you want to know what pin the model, check the Technical Specs https://www.arduino.cc/en/Main/F modified 8 May 2014 by Scott Fitzgerald modified 2 Sep 2016 by Arturo Guadalupi modified 8 Sep 2016 by Colby Newman This example code is in the publ 	WiFi101 / WiFiNINA Firmware Updater Blynk: Check for updates Blynk: Example Builder Blynk: Run USB script Board: "Seeeduino Wio Terminal" Cache: "Enabled" CPU Speed: "120 MHz (standard)" Optimize: "Small (-Os) (standard)" Max QSPI: "50 MHz (standard)" USB Stack: "Arduino" Debug: "Off" Port: "/dev/cu.usbmodem141401" Get Board Info	* * * * * *	Arduino Leonardo Arduino Leonardo ETH Arduino Micro Arduino Esplora Arduino Ethernet Arduino Ethernet Arduino BT LilyPad Arduino USB LilyPad Arduino Arduino Pro or Pro Mini Arduino NG or older Arduino Robot Control Arduino Robot Motor Arduino Gemma
<pre>21 22 http://www.arduino.cc/en/Tutoria 23 */ 24 24 25 // the setup function runs once whe 26 void setup() { 27 // initialize digital pin LED_BUI 28 pinMode(LED_BUILTIN, OUTPUT); 29 } 30 31 // the loop function runs over and 32 void loop() { 33 digitalWrite(LED_BUILTIN, HIGH); 34 delay(1000); 35 digitalWrite(LED_BUILTIN, LOW); 36 digitalWrite(LED_BUILTIN, LOW); 37 digitalWrite(LED_BUILTIN, LOW); 38 digitalWrite(LED_BUILTIN, LOW); 39 digitalWrite(LED_BUILTIN, LOW); 31 digitalWrite(LED_BUILTIN, LOW); 32 digitalWrite(LED_BUILTIN, LOW); 33 digitalWrite(LED_BUILTIN, LOW); 34 delay(1000); 35 digitalWrite(LED_BUILTIN, LOW); 35 digitalWrite(LED_BUILTIN, LOW); 36 digitalWrite(LED_BUILTIN, LOW); 37 digitalWrite(LED_BUILTIN, LOW); 38 digitalWrite(LED_BUILTIN, LOW); 39 digitalWrite(LED_BUILTIN, LOW); 30 digitalWrite(LED_BUILTIN, LOW); 31 digitalWrite(LED_BUILTIN, LOW); 31 digitalWrite(LED_BUILTIN, LOW); 32 digitalWrite(LED_BUILTIN, LOW); 33 digitalWrite(LED_BUILTIN, LOW); 34 digitalWrite(LED_BUILTIN, LOW); 35 digitalWrite(L</pre>	LTIN as an output.		Adafruit Circuit Playground Arduino Yún Mini Arduino Industrial 101 Linino One Arduino Uno WiFi Seeed SAMD (32-bits ARM Cortex-M0+ and Cortex-M4) Boards Seeeduino Wio Terminal Seeeduino XIAO M0 Seeeduino Femto M0 Wio GPS Board Seeeduino Zero Seeeduino LoRaWAN

Choose the right board

Select the serial device of the Wio Terminal board from the **Tools -> Port** menu. This is likely to be COM3 or higher (**COM1** and **COM2** are usually reserved for hardware serial ports). To find out, you can disconnect your Wio Terminal board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.

Note

For Mac User, it will be something like /dev/cu.usbmodem141401

le Edit Sketch T	pols Help					Auto Format	жт	Blink Arduir
Blink Blink	Auto Format Archive Sketch Fix Encoding & Reload Manage Libraries Serial Monitor Serial Plotter	Ctrl+T Ctrl+Shift+I Ctrl+Shift+M Ctrl+Shift+L		Blink	link	Archive Sketch Fix Encoding & Reload Manage Libraries Serial Monitor Serial Plotter	ጉ¥ዘ ጉ¥μ ጉ¥∟	
Turns an L8		Ctri+Shift+L		3 4 1	urns an LED on for one second,	WiFi101 / WiFiNINA Firmware Update	r	
Most Arduir it is attac the correct If you want	WiFi101 / WiFiNINA Firmware Updater Blynk: Check for updates Blynk: Example Builder Blynk: Run USB script		NO, MEGA and 2 UILTIN is set	7 i 8 t	ost Arduinos have an on-board L t is attached to digital pin 13 he correct LED pin independent f vou want to know what pin the	Blynk: Check for updates Blynk: Example Builder Blynk: Run USB script		
model, chech https://www. modified 8 by Scott Fi modified 2 by Arturo (modified 8	Board: "Seeeduino ReScreen (SAMD51) Cache: "Enabled" CPU Speed: "120 MHz (standard)" Optimize: "Small (-cb) (standard)" Max QSPI: "50 MHz (standard)" USB Stack: "Arduino" Debug: "Off"	•		10 m 11 <u>k</u> 12 13 m 14 k 15 m 16 k	odel, check the Technical Specs ttps://www.arduino.cc/en/Main/E odified 8 May 2014 y Scott Fitzgerald dodified 2 Sep 2016 y Arturo Guadalupi odified 8 Sep 2016 y Colby Newman	Board: "Seeeduino Wio Terminal" Cache: "Enabled" CPU Speed: "220 MHz (standard)" Optimize: "Small (-Os) (standard)" Max QSPi: "50 MHz (standard)" USB Stack: "Arduino" Debug: "Off"	* * * * * *	
by Colby Ne	Port		Serial ports	10 1	y couby Newman	Port: "/dev/cu.usbmodem141401"		
This examp:	Get Board Info Programmer: "AVRISP mkll" Burn Bootloader	:	COM9	21	his example code is in the publ ttp://www.arduino.cc/en/Tutoria	Get Board Info Programmer: "AVRISP mkll" Burn Bootloader		/dev/cu.AnsonHesPowerbeats3-SPP-2 /dev/cu.AnsonHesPowerbeats3-Wir /dev/cu.Bluetooth-Incoming-Port /dev/cu.BoseRevolveSoundLink-SP-1

Step 4. Upload the program

Open the LED blink example sketch: File > Examples >01.Basics > Blink.

	New	ЖN	Built-in Examples		Blink Arduino 1.8.12
	Open	жо	01.Basics		AnalogReadSerial
	Open Recent		02.Digital		BareMinimum
Blink	Sketchbook	►	03.Analog	>	Blink
1 /*	Examples	\triangleright	04.Communication		DigitalReadSerial
2 Blink	Close	жw	05.Control		Fade
3	Save	жs	06.Sensors		ReadAnalogVoltage
4 Turns an L	Save As	<mark></mark> ፚ፝፞፞፝፝፝፝	07.Display		
5 Most Arduir		A	08.Strings		
7 it is atta	i ugo ootup	<mark>ଫ</mark> װP	09.USB		

Blink Path

Now, simply click the **Upload** button in the environment. Wait a few seconds and if the upload is successful, the message "Done uploading." will appear in the status bar.



Upload the code

A few seconds after the upload finished, you should see the LED at the bottom of the Wio Terminal start to blink. If it does, congratulations! You've gotten Wio Terminal up-and-running. Please feel free to go through <u>the Wiki of Wio Terminal</u> and start building your projects!

Connect Wio Terminal with Edge Impulse

Step1: Create a new Edge Impulse project

Please Open: https://www.edgeimpulse.com/

Login first, then create a new project.

	Huiying Lai						
WEL	COME!						
2	Your profile						
+	Create new project						

Step 2: Connect the development board to your computer

Connect Wio Terminal to your computer. Entering the bootloader mode by sliding the power

switch twice quickly. For more reference, please also see <u>here</u>.



An external drive named Arduino should appear on your PC. Drag the downloaded Edge Impulse uf2 firmware files to the Arduino drive. Now, Edge Impulse is loaded on Wio Terminal!



Step3: Connect using WebUSB

Go to your Edge Impulse project, and click the Data acquisition tab, then you can see the selection "Connect using WebUSB" on the upper right. Click it.



Then, you can see a pop-tip, select the paired serial port and "Connect" as the following picture.

studio.edgeimpulse.com wants to connect to a serial port

Seeed Wio Terminal (COM12) - Paired		
	Connect	Cancel

Now, you have successfully connected the Wio Terminal with the Edge Impulse.

Record new data	
Device ⑦	
13:B5:FF:15:1D:2B	
Label	Sample length (ms.)
Label name	10000
Sensor	Frequency
Built-in light sensor	∽ 100Hz

Now, all the preparations have been done, we can start our projects!

Add some necessary Arduino libraries

Step1:Download the library

Visit the following repositories and download the entire repo to your local drive when you are doing the corresponding Practice.

Practice 1: none

Practice 2: Seeed_Arduino_LCD 、Grove 3 Axis Digital Accelerometer Practice 3: Seeed_Arduino_LCD Practice 4: Seeed_Arduino_LvGL、Grove Ultrasonic Ranger、Seeed_Arduino_LCD、 Seeed_Arduino_FreeRTOS Practice 5: Grove BME280、Seeed_Arduino_LCD

Step2: Open the Arduino IDE, click sketch -> Include Library -> Add .ZIP Library, and choose the file that you have just downloaded.



Practice 1. Gesture recognition using built-in light sensor (Rock, Vulcan)

Project Overview

In this lesson, we are going to train and deploy a simple neural network for classifying Rock,

Vulcan gestures with just a single light sensor.

Material Preparation

Hardware requirements: Wio Terminal

Connection method:



About Sensor: Built-in Light Sensor



The working principle of this project is quite trivial. Different gestures being moved above the light sensor will block a certain amount of light for certain periods of time. For example, for "Rock", we will have high values at first (nothing above the sensor) but lower values when "Rock" passes above the sensor and then high values again. For "Vulcan", we will have high-low-high-low-high-low values when each of the fingers in "Vulcan" passes above the sensor.



There is a high variance in speed and amplitude of the values from the sensor which makes a great case for using a machine learning model instead of a hand-crafted algorithm for the task.

Machine Learning Lifecycle

Please Open: https://www.edgeimpulse.com/

Data Collection

Step1: Connect Wio Terminal with Edge Impulse

Step2: Know what we are going to do

We are going to train and deploy a simple neural network for classifying rock, Vulcan gestures with just a single light sensor. So we need to select the sensor we are going to use

-- the built-in light sensor, then know what kind of data we are going to sample --gestures of Rock & Vulcan.

Select the sensor we are going to use -- Built-in light sensor

Record new data	
Device ⑦	
13:B5:FF:15:1D:2B	~
Label	Sample length (ms.)
Label name	10000
Sensor	Frequency
Built-in light sensor	100Hz ~
Built-in accelerometer	
Built-in microphone	Chart an an line
Built-in light sensor	Start sampling
External multichannel gas(Grove-multichannel gas v2)	
External temperature&humidity&pressure sensor(Grove-	BME280)
External pressure sensor(Grove-DPS310) External distance sensor(Grove-TFmini)	
External 6-axis accelerometer(Grove-BMI088)	
External ultrasonic sensor(Grove-ultrasonic sensor)	
External CO2+Temp sensor(Grove-SCD30)	

This indicates that we want to record data for 10 seconds (Sample length 10000ms), use a

built-in light sensor and set the frequency to 100Hz.

Know the data we are going to sample -- Rock & Vulcan

Rock



Vulcan



Environment

Sample the data of the Environment around you.

Step3: Sample

Record new data	
Device ⑦	~
Label Rock	Sample length (ms.)
Sensor Built-in light sensor	✓ Frequency 100Hz ✓
	Start sampling

Enter the label(label the recorded data as "Rock" here, we can later edit), click "Start Sampling", then do the gesture "Rock" shown in step 2 in a continuous motion. In about twelve seconds the device should complete sampling and upload the file back to Edge Impulse.

Attention:



1. The light sensor location.



We see a new line appear under 'Collected data' in the studio.

We will be able to preview the data collected after the sample collection is finished. Make

sure that the data is valid before proceeding to collect the next sample.

Collected data		۲ 🛛	± 0	No devices connected to the remote management API.
SAMPLE NAME	LABEL	ADDED LENGTH		
rock.2aettfpm	rock	Jul 14 2021, 18:33 10s	:	rock.2aeedpc2
rock.2aetrfiv	rock	Jul 14 2021, 18:32 10s	:	
rock.2aegkiuj	rock	Jul 14 2021, 14:41 10s	:	
rock.2aegg2jk	rock	Jul 14 2021, 14:39 10s	:	
rock.2aeedpc2	rock	Jul 14 2021, 14:03 10s	:	20 0
rock.2aeec2fd	rock	Jul 14 2021, 14:02 10s	:	0 1040 2080 3120 4160 5200 6240 7280 8320 9360
rock.2aeeaef7	rock	Jul 14 2021, 14:01 10s	:	

Do the same thing to sample "Vulcan" and "Environment".

Now, we have recorded around 1 minute of data per class:

Training data	Test data	I	Export data	
DATA COLLECT 3m 0s	ED			

Impulse Design

After you collected the samples it is time to design an "impulse".



The impulse here is the word Edge Impulse used to denote data processing – training pipeline.



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.

Time series data	Raw Data	Classification (Keras)	Dutput features
Input axes	Name	Name	3 (ENV, Vulcan, rock)
Illumination	Raw data	NN Classifier	
Window size	⑦ Input axes (1)	Input features	
100	oms. Illumination	🖌 Raw data	
Window increase	0 ms.	Output features 3 (ENV, Vulcan, rock)	
Frequency (Hz)	?		
100 C			
Zero-pad data 🗸	0		

Set as follows in this project.

These settings mean that each time an inference is performed we're going to take sensor measurements for 1000 ms. - how many measurements your device is going to take depends on the frequency. During data collection, we set the sampling frequency to 100 Hz or 100 times per 1 second. So, to sum it up, our device is going to gather 100 data samples within 1000 ms. time window and then take these values, preprocess them and feed them to the neural network to get inference results. Of course, we use the same window size during the training.

In all, set the window size to 1000 (you can click on the 1000 ms. text to enter an exact value), the window increases to 100, add the 'Raw data' and 'Classification(Keras)' blocks. Then click Save impulse.

Feature Extraction--RAW Data To configure your signal processing block, click Raw data in the menu on the left.



we use the default parameter and click "Save parameters" then click "Generate features" to start the process.

Parameters		
Scaling		
Scale axes	1	
		Save parameters

Afterwards the 'Feature explorer' will load. This is a plot of all the extracted features against all the generated windows. You can use this graph to compare your complete data set. A good rule of thumb is that if you can visually separate the data on a number of axes, then the machine learning model will be able to do so as well.



Model Training--Network: MLP

To configure our learning block, click "NN Classifier" in the menu on the left.



We create a simple neural network – MLP(Multilayer perceptron) here.

Neural Network settings	I
Training settings	
Number of training cycles ⑦	30
Learning rate ⑦	0.0005
Neural network architecture	
Input layer (100 features)	
Dense layer (64 neurons)	
Dense layer (32 neurons)	
Add an extra layer	
Output layer (3 classes)	
Start training	

Click on"Start training".

By observing the "Log", you can deduce the model performance.

```
Training output
                                                                     Cancel
Epoch 4/30
41/41 - 1s - loss: 1.8479 - accuracy: 0.7977 - val_loss: 0.9865 - val_accuracy:
0.9024
Epoch 5/30
41/41 - 1s - loss: 0.9707 - accuracy: 0.8931 - val_loss: 0.9381 - val_accuracy:
0.8811
Epoch 6/30
41/41 - 1s - loss: 0.8653 - accuracy: 0.8954 - val loss: 0.8424 - val accuracy:
0.8811
Epoch 7/30
41/41 - 1s - loss: 0.7583 - accuracy: 0.9168 - val_loss: 0.9121 - val_accuracy:
0.8689
Epoch 8/30
41/41 - 1s - loss: 0.9687 - accuracy: 0.8786 - val loss: 0.7972 - val accuracy:
0.9177
Epoch 9/30
```

The number of Training Cycles is also called an epoch. It specifies the number of times that the entire training dataset passes through the training process of the algorithm. With each epoch, the internal model parameters will update, which may help the model perform better. As the training log shows, "Epoch: 8/30" indicates the total number of training rounds is 30 while 8 rounds have been trained. Training accuracy has been achieved 0.8786.

Vodel		Model version: ③	Quantized (int8) 🔻
ast training perfo	rmance (validation set	:)	
ACCURACY		LOSS 0.82	
95.7% Confusion matrix (v	validation set)	0.02	
95.770	validation set)	VULCAN	ROCK
95.770			ROCK 0%
Confusion matrix (ENV	VULCAN	
ENV	ENV 100%	VULCAN 0%	0%

Now, the Model training is complete.

After we have our model and are satisfied with its accuracy in training, we can test it on new data in the Live classification tab.

Live classification

If the live classification gets poor performance, we need to analyze the reason and retrain our model. We will talk about it in future projects.

Model Optimization

There are various ways of optimizing machine learning models: compression, pruning and quantization. Compression is the process of reducing the size of a machine learning model. Pruning is the process of removing the weights of unimportant neurons from a machine learning model.

Quantization is the process of converting floating point numbers to integers.

Model		Model version: (?)	Quantized (int8) 💌
Last training perform	mance (validation set)		
ACCURACY 95.7% Confusion matrix (va	lidation set)	LOSS 0.82	
	ENV	VULCAN	ROCK
ENV	100%	0%	0%
VULCAN	2.8%	92.6%	4.6%
ROCK	0%	5.4%	94.6%
F1 SCORE	0.99	0.93	0.95

This is done to save space and time. These optimisations not only make the model run faster but also help to reduce the memory consumption requirements of the system.

Model Deployment

The next step is deployment on the device.

Step 1: After clicking on the Deployment tab, choose Arduino library and download it.

Deploy your impulse

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.

Create library

Turn your impulse into optimized source code that you can run on any device.



Step 2: Now, the library can be installed to the Arduino IDE. Open the Arduino IDE, click sketch -> Include Library -> Add .ZIP Library, and choose the file that you have just downloaded.

É Arduino File Edit	Sketch Tools Help		
	Verify/Compile Upload	ЖR ЖU	Manage Libraries
	Upload Using Programmer	☆業U	Add .ZIP Library
Test.ino	Export compiled Binary	τ #S	Arduino libraries
<pre>1 void setup() { 2 // put your setup code here_</pre>	Show Sketch Folder	ЖК	Bridge
3	Include Library		Esplora
4 } 5	Add File		Ethernet

Step 3: Open Examples -> name of your project -> static buffer.

static_buffer	
<pre>/* Includes #include <gesture_recognition_inferencing.h></gesture_recognition_inferencing.h></pre>	your project library name

Step 4: Copy the following Example Code to replace the original example code:

#include <project_71231_inferencing.h> //replace with your project library name

```
ei_impulse_result_classification_t
currentClassification[EI_CLASSIFIER_LABEL_COUNT];
const char* maxConfidenceLabel;
void runClassifier()
{
   float buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = { 0 };
   uint8_t axis_num = 1;
   for (size_t ix = 0; ix < EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE; ix += axis_num) {
     uint64_t next_tick = micros() + (EI_CLASSIFIER_INTERVAL_MS * 1000);
     buffer[ix + 0] = analogRead(WIO_LIGHT);
     delayMicroseconds(next_tick - micros());
   }
   signal_t signal;</pre>
```

```
int err = numpy:: signal_from_buffer(buffer,
EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &signal);
  ei_impulse_result_t result = { 0 };
  err = run_classifier(&signal, &result, false);
  float maxValue = 0;
  for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {</pre>
    ei_impulse_result_classification_t classification_t =
result.classification[ix];
    ei printf("
                 %s: %.5f\n", classification_t.label, classification_t.value);
    float value = classification_t.value;
    if (value > maxValue) {
     maxValue = value;
      maxConfidenceLabel = classification_t.label;
    }
    currentClassification[ix] = classification_t;
  }
}
void setup(){
  Serial.begin(9600);
}
void loop(){
  runClassifier();
  Serial.println(maxConfidenceLabel);
}
```

Step 5: Upload the code.



It takes about 5 mins to upload. If the upload is successful, the message "Done uploading." will appear in the status bar.

Step 6: Open the Serial monitor. Move your hand while performing a gesture and see the probability result printed out on the Serial monitor.

Environment

💿 COM16	
Vulcan: 0.02344	
rock: 0.00000	
ENV	
ENV: 0.87109	
Vulcan: 0.12891	
rock: 0.00000	
ENV	
ENV: 0.87109	
Vulcan: 0.12891	
rock: 0.00000	
ENV	
ENV: 0.99219	
Vulcan: 0.00781	
rock: 0.00000	
ENV	
Autoscroll 🗌 Show timestamp	

Rock:



Vulcan:



Now, the model may perform well with recognizing your gestures but is poor at dealing with other people's "Rock"& "Vulcan".

You can optimize your model as follows:

When collecting samples it is important to provide diversity for the model to be able to generalize better, for example, have samples with different directions, speeds and distances from the sensor. In general, the network only can learn from data present in the dataset – so if the only samples you have are gestures being moved from left to right above the sensor, you shouldn't expect the trained model to be able to recognize gestures being moved right to left or up and down.

Collect more data samples in different light conditions.

While it was just a proof of concept demonstration, it really shows TinyML is up to something big. You probably knew it is possible to recognize gestures with a camera sensor, even if the image is down-scaled a lot. But recognizing gestures with just 1 pixel is an entirely different level!

Congratulations, we have done our first project!

Reference Edge Impulse Public project: https://studio.edgeimpulse.com/public/76967/latest

Practice 2. Motion Recognition using built-in accelerometer (Flip, Wave, Idle)

Project Overview

In this lesson, we'll take on a similar task, motion recognition, but will use a different sensor for that - a 3-axis accelerometer. This is a hard task to solve using rule-based programming, as people don't perform gestures in the exact same way every time. But machine learning can handle these variations with ease.



Material Preparation

Hardware requirements: Wio Terminal

Connection method:

Environ E	🚍 2038 MPULSE	EZA KÖJATAK ALANA ALANA MARANA Tarahan Alan Mara
A matrixes A		💼 state par la mana su ser separate constituires de variante de appendiente, en aplendies a mana protocomenta de variantes estates esta
territorio de la construición de la construici	🚸 – maaite ameri	enter O Merilla O Merilla
	 Sector Fitbulk Fitbulk Fitbulk Fitbulk Fitbulk Fitbulk Fitbulk Weinstate Weinstate Weinstate 	Network Mail Mail

About sensor: accelerometers

As you might guess from the name, accelerometers are devices that measure the acceleration of a body. It is defined as the rate of change of the velocity of an object. Accelerometers are capable of measuring acceleration either in meters per second squared (m/s2) or in G-forces (g). A single G-force on Earth is equivalent to 9.8 m/s2. There are different kinds of accelerometers. The earliest accelerometers were based on mechanical architecture.

The first accelerometer was called the Atwood machine. It was invented by an English physicist George Atwood.



The accelerometers commonly used in mobile phones are MEMS (Microelectromechanical) accelerometers.





The module used in Wio Terminal is called 3-Axis Digital Accelerometer (LIS3DHTR). Generally, the internal structure of accelerometers consists of Capacitive Plates. Some types of accelerometers use fixed capacitive plates, while some of them have the plates attached to minuscule springs that move internally depending upon the acceleration forces acting on the sensor.



1.Mass presses capacitor plate



2.Mass closes plates, changing capacitance

Machine Learning Lifecycle

Please Open: https://www.edgeimpulse.com/

Data Collection

Step1: Connect Wio Terminal with Edge Impulse

Step2: Know what we are going to do

We are going to train and deploy a simple neural network for classifying Flip, Wave, Idle motion with just an accelerometer. So we need to select the sensor we are going to use -- a built-in accelerometer, then know what kind of data we are going to sample --the motion of "Flip", "Wave", "Idle".

Select the sensor we are going to use -- Built-in accelerometer

Record new data	
Device ⑦	
13:B5:FF:15:1D:2B	~
Label	Sample length (ms.)
Label name	10000
Sensor	Frequency
Built-in accelerometer ~	62.5Hz ~
Built-in accelerometer Built-in microphone Built-in light sensor External multichannel gas(Grove-multichannel gas v2) External temperature&humidity&pressure sensor(Grov External pressure sensor(Grove-DPS310) External distance sensor(Grove-TFmini) External distance sensor(Grove-TFmini) External ultrasonic sensor(Grove-ultrasonic sensor)	e-BME280)

This indicates that we want to record data for 10 seconds (Sample length 10000ms), use the built-in accelerometer and frequency 62.5Hz.

Know the data we are going to sample -- Flip, Wave, Idle

- Idle just sitting on your desk while you're working.
- Flip turn device, similar to how you would turn a valve
- Wave waving the device from left to right.

Idle



Flip



wave



Step3: Sample

Record new data	
Device	
13:85:FF:15:1D:28	~
Label	Sample length (ms.)
Wave	10000
Sensor	Frequency
Built-in accelerometer	62.5Hz ~
Built-in accelerometer	
Built-in microphone	Start sampling
Built-in light sensor	Start sampling
External multichannel gas(Grove-multichannel gas v2)	B1453001
External temperature&humidity&pressure sensor(Grove External pressure sensor(Grove-DPS310)	E-BME280)
External pressure sensor(Grove-DFSST0) External distance sensor(Grove-TFmini)	
External 6-axis accelerometer(Grove-BMI088)	
External ultrasonic sensor(Grove-ultrasonic sensor)	
External CO2+Temp sensor(Grove-SCD30)	

Enter the label, click "Start Sampling", then do the gesture "Wave" shown in step 2 in a **continuous motion.** In about twelve seconds the device should complete sampling and upload the file back to Edge Impulse. You see a new line appear under 'Collected data' in the studio. When you click it you now see the raw data graphed out. As the accelerometer on the development board has three axes you'll notice three different lines, one for each axis.



Note

Make sure to perform variations on the motions. E.g. do both slow and fast movements and vary the orientation of the board. You'll never know how your user will use the device.

Gather data from 2 other people, except for yourself.

Now, we have recorded around 1min of data per class:

Test data	Т	Export data	
FED			

Impulse Design

With the training set in place, we can design an impulse.

Time series 🛛 🛢	Spectral Analysis	Classification (Keras)	Output features
Input axes (3)	Name	Name	3 (flip, idle, wave)
accX, accY, accZ	Spectral features	NN Classifier	
Window size ⑦	Input axes (3)	Input features	Save Impulse
2000 ms.	accX	 Spectral features 	
Window increase 🏾 🄊	accY	Output features	
80 ms.	✓ accZ	3 (flip, idle, wave)	
Frequency (Hz) 📀			
62.5 C			
Zero-pad data 🛛 📀			
⊻			
5 S			

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 tutorial, 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 three (Idle, Flip and Wave) classes.

In all, set the window size to 2000 (you can click on the 2000 ms. text to enter an exact value), the window increases to 80, and adds the 'Spectral Analysis' and 'Classification (Keras)' blocks. Then click Save impulse.

Feature Extraction--Spectral Analysis

To configure your signal processing block, click Spectral features in the menu on the left.



This will show you the raw data on top of the screen (you can select other files via the drop-down menu).



And the results of the signal processing through graphs on the right.

For the spectral features block you'll see the following graphs:

- After filter the signal after applying a low-pass filter. This will remove noise.
- Frequency domain the frequency at which signal is repeating (e.g. making one wave

movement per second will show a peak at 1 Hz).

• Spectral power - the amount of power that went into the signal at each frequency.



A good signal processing block will yield similar results for similar data. If you move the sliding window (on the raw data graph) around, the graphs should remain similar. Also, when you switch to another file with the same label, you should see similar graphs, even if the orientation of the device was different.

Parameters	
Scaling	
Scale axes	1
Filter	
Туре	low ~
Cut-off frequency	3
Order	6
Spectral power	
FFT length	128
No. of peaks	3
Peaks threshold	0.1
Power edges	0.1, 0.5, 1.0, 2.0, 5.0
	Save parameters

Click "Save parameters" then click "Generate features" to start the process.

Afterwards the 'Feature explorer' will load. This is a plot of all the extracted features against all the generated windows. You can use this graph to compare your complete data set.



Model Training--Network: MLP

With all data processed it's time to start training a neural network. To configure our learning block, click "NN Classifier" in the menu on the left.



Neural networks are a set of algorithms, modeled loosely after the human brain, that is 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 three classes.

We create a simple neural network – MLP(Multilayer perceptron) here.

Neural Network settings		:		
Training settings				
Number of training cycles ⑦	10			
Learning rate ③	0.001			
Validation set size ③	20	%		
Auto-balance dataset ③				
Neural network architecture				
Input layer (33 feature	es)			
Dense layer (20 neuro	ns)			
Dense layer (10 neurons)				
Add an extra layer				
Output layer (3 classe	5)			
Start training				

Now, the Model training is complete.
Model		Model	version: ② Quantized (int8) 👻
Last training performance (v	alidation set)		
ACCURACY 99.7% Confusion matrix (validation se)	LOSS 0.02	
	FLIP	IDLE	WAVE
FLIP	100%	0%	096
IDLE	0%	100%	0%
WAVE	0%	0.8%	99.2%
F1 SCORE	1.00	1.00	1.00
accX RMS • flip - correct • idle - correct • wave - correct • wave - incorrect • wave - incorrect • wave - correct • wave - correct	accY RMS	✓ ac	cz RMS v
On-device performance ⑦			
INFERENCING TIME 1 ms.	PEAK RAM 1.7K	USAGE	FLASH USAGE 19.3K

INFERENCING TIME: The time that a device takes to complete a prediction. PEAK RAM USAGE and FLASH USAGE: Let's look at the Wio Terminal's Flash and RAM.

Powerful MCU - Microchip ATSAMD51P19

- ARM Cortex-M4F core running at 120MHz (Boost up to 200MHz)
- 4 MB External Flash, 192 KB RAM

Every embedded machine learning model has constraints. The model we get this time meets these constraints.

After we have our model and are satisfied with its accuracy in training, we can test it on new

data in the Live classification tab.



If the live classification gets poor performance, we need to analyze the reason and retrain our model.

Model Optimization Model optimization follows Practice 1.

Model Deployment

The next step is deployment on the device.

Step 1: After clicking on the Deployment tab, choose Arduino library and download it.



Step 2: Now, the library can be installed to the Arduino IDE. Open the Arduino IDE, click sketch -> Include Library -> Add .ZIP Library, and choose the file that you have just downloaded.

É Arduino File Edit	Sketch Tools	Help			
			ЖR ЖU	Manage Libraries	
				Add .ZIP Library	
Test.ino 1 void setup() {		-		Arduino libraries	
<pre>2 // put your setup code here 3</pre>	Show Sketch Folder Include Library		¥К	Bridge Esplora	
4 } 5	Add File			Ethernet	

Step 3: Open Examples -> name of your project -> static buffer.



Step 4: Copy the following Example Code to replace the original example code:

Also, since Wio Terminal has an LCD screen, we're going to display the name of the

detected class if this class confidence value is above the threshold.

```
#include"TFT eSPI.h"
#include <project 48833 inferencing.h>//replace with your project library name
#include"LIS3DHTR.h"
TFT_eSPI tft;
LIS3DHTR<TwoWire> lis;
                           9.80665f
#define CONVERT G TO MS2
ei impulse result classification t
currentClassification[EI_CLASSIFIER_LABEL_COUNT];
const char* maxConfidenceLabel;
void runClassifier()
{
  float buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = { 0 };
  for (size_t ix = 0; ix < EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE; ix += 3) {</pre>
    uint64_t next_tick = micros() + (EI_CLASSIFIER_INTERVAL_MS * 1000);
    lis.getAcceleration(&buffer[ix], &buffer[ix + 1], &buffer[ix + 2]);
    buffer[ix + 0] *= CONVERT_G_TO_MS2;
    buffer[ix + 1] *= CONVERT_G_TO_MS2;
    buffer[ix + 2] *= CONVERT_G_TO_MS2;
    delayMicroseconds(next_tick - micros());
  }
  signal_t signal;
  int err = numpy:: signal from buffer(buffer,
EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &signal);
  ei_impulse_result_t result = { 0 };
  err = run classifier(&signal, &result, false);
  float maxValue = 0;
  for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {</pre>
    ei_impulse_result_classification_t classification_t =
result.classification[ix];
    ei_printf("
                  %s: %.5f\n", classification t.label, classification t.value);
    float value = classification t.value;
    if (value > maxValue) {
      maxValue = value;
      maxConfidenceLabel = classification t.label;
    }
    currentClassification[ix] = classification_t;
  }
}
void setup(){
 tft.begin();
  lis.begin(Wire1);
  lis.setOutputDataRate(LIS3DHTR DATARATE 100HZ);
  lis.setFullScaleRange(LIS3DHTR_RANGE_4G);
```

```
tft.setRotation(3);
tft.setTextSize(4);
}
void loop(){
  runClassifier();
  tft.drawString((String)maxConfidenceLabel, 120, 120);
}
```





It takes about 5 mins to upload. If the upload is successful, the message "Done uploading." will appear in the status bar.

Step 6: Wave the Wio Terminal and see the probability result printed out on the LCD screen.



Reference Edge Impulse Public project:

Practice 3.Keyword Spotting using built-in microphone. (Hello Wio)

Project Overview

This model uses Wio Terminal built-in microphone to collect vocal wake words and ambient sounds to train the model. This microphone also helps to wake up the device with the wake-up word ("Hello Wio").

Material Preparation

Hardware requirements: Wio Terminal

Connection method:

	229 - 420 - 4	 Hittini y se Nation y se
 Densitie Statutie Statutie<		

About sensor

Sound is a vibration that propagates (or travels) as an acoustic wave, through a

a transmission medium such as a gas, liquid or solid.



The source of sound pushes the surrounding medium molecules, they push the molecules next to them and so on and so forth. When they reach other objects it also vibrates slightly – we use that principle in the microphone. The microphone membrane gets pushed inward by the medium molecules and then back to its original position.



That generates an alternating current in the circuit, where voltage is proportional to the sound amplitude – the louder the sound, the more it pushes the membrane, thus generating higher voltage. We then read this voltage with an analogue-to-digital converter and record at equal intervals – the number of times we take measurement of sound in one second is called a sampling rate, for example, 8000 Hz sampling rate is taking measurement 8000 times per second. Sampling rate obviously matters a lot for the quality of the sound – if we sample too slow we might miss important bits and pieces. The numbers used for recording sound digitally also matter – the larger range of a number used, the more "nuances" we can preserve from the original sound. That is called audio bit depth – you might have heard terms like 8-bit sound and 16-bit sound. Well, it is exactly what it says on the tin – for 8-bit sound unsigned 8-bit integers are used, which have ranged from 0 to 255. For 16-bit sound signed 16-bit integers are used, so that's -32768 to 32767. Alright, so in the end we have a string of numbers, with larger numbers corresponding to loud parts of the sound and we can



visualize it like this - this is 1 second of gunshot sound recorded at 8000 Hz frequency in 8-bit depth (0-255).

We can't do much with this raw sound representation though – yes, we can cut and paste the parts or make it quieter or louder, but for analyzing the sound, it is, well, too raw. Here is where Fourier transform, Mel scale, spectrograms and cepstrum coefficients come in. For the purpose of this project, we'll define the Fourier transform as a mathematical transform that allows us to decompose a signal into its individual frequencies and the frequency's amplitude.



Machine Learning Lifecycle

Please Open: <u>https://www.edgeimpulse.com/</u>

Data Collection

Step1: Connect Wio Terminal with Edge Impulse

Step2: Know what we are going to do

We are going to train and deploy a simple neural network for keyword spotting with just a built-in microphone. So we need to select the sensor we are going to use -- Built-in microphone, then know what kind of data we are going to sample --keyword.

Select the sensor we are going to use -- a Built-in microphone

Record new data	
Device ⑦	
Wio Terminal	~
Label	Sample length (ms.)
Label name	5000
Sensor	Frequency
Built-in microphone	16000Hz ~
Built-in accelerometer	
Built-in microphone	Start sampling
Built-in light sensor	Start sumpling
External multichannel gas(Grove-multichannel gas v2)	
External temperature&humidity&pressure sensor(Grove-BME280)	
External pressure sensor(Grove-DPS310)	
External distance sensor(Grove-TFmini) External 6-axis accelerometer(Grove-BMI088)	
External 6-axis accelerometer(Grove-BMI088) External ultrasonic sensor(Grove-ultrasonic sensor)	
External dirasonic sensor(Grove-dirasonic sensor) External CO2+Temp sensor(Grove-SCD30)	

This indicates that we want to record data for 5 seconds (Sample length 5000ms), use the built-in microphone and frequency 16000 Hz.

Know the data we are going to sample.

We want to build a system that recognizes keywords, so our first job is to think of a great one. It can be the name of your device, the name of your pet, etc. But keep in mind that some keywords are harder to distinguish from others, and especially keywords with only one syllable might lead to false- positives(like 'Hi'). This is the reason that Apple, Google and Amazon all use at least three-syllable keywords ('Hey Siri', 'OK, Google', 'Alexa').

So we choose "Hello Wio", and say hello to our Wio Terminal.

Hello Wio



In addition to our keyword, we'll also need audio that is not our keyword. Like background noise, and humans saying other words.

Background



Acquire background sound

Record the ambient background sound

Unknown



other words

Say some common words different from the wake-up wor

This is required because a machine learning model has no idea about right and wrong, but only learns from the data we feed into it.

So we should tell the machine learning model; when you hear this, this is background, when you hear that, that is unknown words. and only when you hear "hello Wio "that is Hello Wio.

Step3: Sample

Record new data	
Device ⑦	
Wio Terminal	~
Label	Sample length (ms.)
Hello wio	5000
Sensor	Frequency
Built-in microphone	16000Hz 🗸
Built-in accelerometer Built-in microphone Built-in light sensor External multichannel gas(Grove-multichannel gas v2)	Start sampling
External temperature/kiumalidi/kp/exsure sensor(Grove-BME280) External pressure sensor(Grove-Dra7510) External distance sensor(Grove-Dra7510) External distance sensor(Grove-Trimin) External durasionic sensor(Grove-BMI088) External durasionic sensor(Grove-BMI088) External durasionic sensor(Grove-BMI088)	

Enter the label, click "Start Sampling", and start saying our keyword over and over again (with some pause in between). Because the recording needs to use SPI Flash which will operate erasing, the time it takes usually longer than we set.

Afterwards, we have a file like this, clearly showing our keywords, separated by some noise. So we can see that I have three. Hello Wio.



This data is not suitable for Machine Learning yet though. We will need to cut out the parts where we say our keyword. This is important because we only want the actual keyword to be labeled as such, and not accidentally label noise, or incomplete sentences.

Tap the little three dots here. : and select the "Split sample".

Collected data		T	~	1	0
SAMPLE NAME	LABEL	ADDED	LEN	NGTH	
hello_wio.2po	hello_wio	Today, 20:	5s	,	:

we want to look at a certain window length a second here.

Split sample 'hello_wio.2poarog4'



That is what we are going to look at. And we need to make sure that the actual word is in there, not noise.



If it has a window like that, there's actually only noise in there, the model gets confused.

which is very bad for the accuracy of the model.

In addition, we can either collect this ourselves or make our life a bit easier by

using a dataset that we get online.

If we search online, we can find some data specially made for keyword spotting. And edge impulse also provides such a dataset. This is a prebuilt dataset for a keyword spotting system based on a subset of data in the <u>Google Speech Commands Dataset</u>, with added noise

from the Microsoft Scalable Noisy Speech Dataset. It contains 25 minutes of data per class, split up in 1 second windows, sampled at 16,000Hz.

Make sure to capture wide variations of the keyword: leverage our family and our colleagues to help us collect the data, make sure we cover high and low pitches and slow and fast speakers.

Make sure we have a well-balanced dataset.



Impulse Design

With the training set in place, we can design an impulse.

Time series 😑	Audio (MFCC)	Classification (Keras)	Output features
Input axes audio Window size ⑦	Name MFCC	Name NN Classifier	3 (background, hello_wio, unknown)
1000 ms. Window increase @	Input axes (1)	Input features Input features Image: MFCC	Save Impulse
500 ms. Frequency (Hz) ?		Output features 3 (background, hello_wio, unknown)	
16000 Cª Zero-pad data ⑦ ✔	Ŧ	•	
P 💼			

And the pipeline consists of the default settings for time series, data window are correct–1000ms. And the window increase here is not going to be used, because all of our data is already a second long.

Then we add a preprocessing block and we use signal processing to clean up the data before feeding it to the neural network. We have lots of processing blocks for a wide variety of typical senses. We want one specifically for audio. We have the normal spectrogram which is really great for non-voice audio. And then we have an MFE block as well which you can also use for non-voice audio. And here we are dealing with the human voice. So, we'll use the "MFCC" signal processing block.

Audio (MFCC) Extracts features from audio signals using Mel Frequency Cepstral EdgeImpulse Inc. + Coefficients, great for human voice.

Feature Extraction--MFCC

MFCC stands for Mel Frequency Cepstral Coefficients. This sounds scary, but it's basically just a way of turning raw audio—which contains a large amount of redundant information—into a simplified form. the "MFCC" block is great for dealing with human speech.

Parameters

Mel Frequency Cepstral Coefficients	
Number of coefficients	13
Frame length	0.02
Frame stride	0.02
Filter number	32
FFT length	256
Normalization window size	101
Low frequency	300
High frequency	Click to set
Pre-emphasis	
Coefficient	0.98
Shift	1
	Save parameters

So with all these default parameters set by Edge Impulse for such a project, we won't change them this time, let's go to generate features.

Feature explorer (547 sam	bles)			0
X Axis	Y Axis		Z Axis	
Visualization layer 1	✓ Visualization layer 2	~	Visualization layer 3	~
 background hello_wio unknown 	Visualization layer 3, 6.5 1.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0	Visualization layer 1	\$ 3	

This is the feature explorer.

So what we see here is all the data in my data sets are shown in three dimensions after the feature extraction step.

What I am interested in is whether my "Hello Wio", and my unknown words are nicely separated.

And we see that as a nice separation between the orange clusters, which all contain "Hello Wio" samples and the green cluster contains "unknown" words.

Feature explorer (547 samples)					3
X Axis		Y Axis		Z Axis	
Visualization layer 1	~	Visualization layer 2	~	Visualization layer 3	~
 background hello_wio unknown 	Visualization layer 3	² ² ² ² ² ² ² ² ² ²			

This is a great way to check whether our dataset contains wrong items and to validate whether our dataset is suitable for ML (it should separate nicely).

Model Training--Network:CNN

With all data processed it's time to start training a neural network

Aud	io training options				
Data	a augmentation ⑦	•			
	Add noise 🕲		None	Low	High
	Mask time bands ⑦		None	Low	High
	Mask frequency bands 🕲		None	Low	High
	Warp time axis ③	(

enable 'Data augmentation', a super-powerful feature where during training we randomly manipulate data in our training data set. So we can add artificial noise to make it more resilient to noisy environments

This is a very quick way to make our dataset work better in real life (with unpredictable sounds coming in) and prevents our neural network from overfitting (as the data samples are changed every training cycle).

We use the default hyperparemeters and neural network provided by Edge Impulse.

Number of training cycles ⑦	100					
Learning rate ②	0.005					
Neural network architecture						
Architecture presets ⁽²⁾ 1D Convolutional (Default) 2D Convolutional						
Input layer (650 features)						
Backers (22)						
Reshape layer (13 columns)						
1D conv / pool layer (8 neurons, 3 kernel size, 1 layer)						
Dropout (rate 0.25)						
1D conv / pool layer (16 neurons, 3 kernel size, 1 layer)						
Dropout (rate 0.25)						
Flatten layer						
Add an extra layer						
Output layer (3 classes)	Output layer (3 dasses)					

Model Optimization

Model Deployment

The next step is deployment on the device.

Step 1: After clicking on the Deployment tab, choose Arduino library and download it.



Step 2: Now, the library can be installed to the Arduino IDE. Open the Arduino IDE, click sketch -> Include Library -> Add .ZIP Library, and choose the file that you have just downloaded.

É Arduino File Edit	Sketch Tools Help		
	Verify/Compile Upload	業R 業U	Manage Libraries
	Upload Using Programmer	企業U	Add .ZIP Library
Test.ino	Export compiled Binary	₹₩S	Arduino libraries
<pre>1 void setup() { 2 // put your setup code here.</pre>	Show Sketch Folder	ЖК	Bridge
3	Include Library		Esplora
4 } 5	Add File		Ethernet

Step 3: Open Examples -> name of your project -> static buffer.



Step 4: Copy the following Example Code to replace the original example code:

For Wio Terminal we will rely on DMA or Direct Memory Access controller to obtain

samples from ADC (Analog to Digital Converter) and save them to the inference buffer

without the involvement of MCU.

That will allow us to collect the sound samples and perform inference at the same time.

```
#include"TFT eSPI.h"
#include <project_67469_inferencing.h>
enum {ADC_BUF_LEN = 1600};
typedef struct {
 uint16_t btctrl;
 uint16_t btcnt;
 uint32_t srcaddr;
 uint32_t dstaddr;
 uint32_t descaddr;
}dmacdescriptor;
typedef struct {
  signed short *buffers[2];
 unsigned char buf_select;
 unsigned char buf ready;
 unsigned int buf count;
 unsigned int n_samples;
}inference_t;
volatile uint8 t recording = 0;
uint16_t adc_buf_0[ADC_BUF_LEN];
uint16 t adc buf 1[ADC BUF LEN];
volatile dmacdescriptor wrb[DMAC_CH_NUM] __attribute__ ((aligned (16)));
dmacdescriptor descriptor_section[DMAC_CH_NUM] __attribute__ ((aligned (16)));
dmacdescriptor descriptor __attribute__ ((aligned (16)));
static inference t inference;
class FilterBuHp1{
 public:
 FilterBuHp1(){
   v[0] = 0.0;
 }
 private:
 float v[2];
 public:
 float step(float x)
  {
   v[0] = v[1];
   v[1] = (9.621952458291035404e-1f * x) + (0.92439049165820696974f * v[0]);
   return (v[1] - v[0]);
    }
};
FilterBuHp1 filter;
static void audio_rec_callback(uint16_t *buf, uint32_t buf_len) {
 if (recording) {
    for (uint32_t i = 0; i < buf_len; i++) {</pre>
      inference.buffers[inference.buf_select][inference.buf_count++] =
filter.step(((int16_t)buf[i] - 1024) * 16);
      if (inference.buf_count >= inference.n_samples) {
        inference.buf select ^= 1;
        inference.buf_count = 0;
        inference.buf_ready = 1;
```

```
}
    }
 }
}
void DMAC_1_Handler() {
  static uint8_t count = 0;
  if (DMAC->Channel[1].CHINTFLAG.bit.SUSP) {
    DMAC->Channel[1].CHCTRLB.reg = DMAC_CHCTRLB_CMD_RESUME;
    DMAC->Channel[1].CHINTFLAG.bit.SUSP = 1;
    if (count) {
      audio_rec_callback(adc_buf_0, ADC_BUF_LEN);
    }else {
      audio_rec_callback(adc_buf_1, ADC_BUF_LEN);
    }
    count = (count + 1) % 2;
  }
}
void config_dma_adc() {
  DMAC->BASEADDR.reg = (uint32_t)descriptor_section;
  DMAC->WRBADDR.reg = (uint32_t)wrb;
  DMAC->CTRL.reg = DMAC_CTRL_DMAENABLE | DMAC_CTRL_LVLEN(0xf);
  DMAC->Channel[1].CHCTRLA.reg = DMAC CHCTRLA TRIGSRC(TC5 DMAC ID OVF) |
                                 DMAC_CHCTRLA_TRIGACT_BURST;
  descriptor.descaddr = (uint32_t)&descriptor_section[1];
  descriptor.srcaddr = (uint32_t)&ADC1->RESULT.reg;
  descriptor.dstaddr = (uint32_t)adc_buf_0 + sizeof(uint16_t) * ADC_BUF_LEN;
  descriptor.btcnt = ADC_BUF_LEN;
  descriptor.btctrl = DMAC_BTCTRL_BEATSIZE_HWORD |
                      DMAC_BTCTRL_DSTINC |
                      DMAC_BTCTRL_VALID
                      DMAC_BTCTRL_BLOCKACT_SUSPEND;
  memcpy(&descriptor_section[0], &descriptor, sizeof(descriptor));
  descriptor.descaddr = (uint32_t)&descriptor_section[0];
  descriptor.srcaddr = (uint32_t)&ADC1->RESULT.reg;
  descriptor.dstaddr = (uint32_t)adc_buf_1 + sizeof(uint16_t) * ADC_BUF_LEN;
  descriptor.btcnt = ADC_BUF_LEN;
  descriptor.btctrl = DMAC_BTCTRL_BEATSIZE_HWORD |
                      DMAC_BTCTRL_DSTINC |
                      DMAC_BTCTRL_VALID
                      DMAC BTCTRL BLOCKACT SUSPEND;
  memcpy(&descriptor_section[1], &descriptor, sizeof(descriptor));
  NVIC_SetPriority(DMAC_1_IRQn, 0);
  NVIC_EnableIRQ(DMAC_1_IRQn);
  DMAC->Channel[1].CHINTENSET.reg = DMAC_CHINTENSET_SUSP;
```

```
ADC1->INPUTCTRL.bit.MUXPOS = ADC INPUTCTRL MUXPOS AIN12 Val;
  while (ADC1->SYNCBUSY.bit.INPUTCTRL);
  ADC1->SAMPCTRL.bit.SAMPLEN = 0x00;
  while (ADC1->SYNCBUSY.bit.SAMPCTRL);
  ADC1->CTRLA.reg = ADC_CTRLA_PRESCALER_DIV128;
  ADC1->CTRLB.reg = ADC_CTRLB_RESSEL_12BIT |
                    ADC_CTRLB_FREERUN;
  while (ADC1->SYNCBUSY.bit.CTRLB);
  ADC1->CTRLA.bit.ENABLE = 1;
  while (ADC1->SYNCBUSY.bit.ENABLE);
  ADC1->SWTRIG.bit.START = 1;
  while (ADC1->SYNCBUSY.bit.SWTRIG);
  DMAC->Channel[1].CHCTRLA.bit.ENABLE = 1;
  GCLK->PCHCTRL[TC5 GCLK ID].reg = GCLK PCHCTRL CHEN |
                                   GCLK PCHCTRL GEN GCLK1;
  TC5->COUNT16.WAVE.reg = TC WAVE WAVEGEN MFRQ;
  TC5->COUNT16.CC[0].reg = 3000 - 1;
  while (TC5->COUNT16.SYNCBUSY.bit.CC0);
  TC5->COUNT16.CTRLA.bit.ENABLE = 1;
  while (TC5->COUNT16.SYNCBUSY.bit.ENABLE);
}
static bool microphone_inference_record(void) {
  bool ret = true;
  while (inference.buf_ready == 0) {
    delay(1);
  }
  inference.buf_ready = 0;
  return ret;
}
static int microphone_audio_signal_get_data(size_t offset,
                                               size_t length,
                                               float *out_ptr) {
  numpy::int16_to_float(&inference.buffers[inference.buf_select ^ 1][offset],
out_ptr, length);
  return 0;
}
TFT eSPI tft;
ei impulse result classification t
currentClassification[EI_CLASSIFIER_LABEL_COUNT];
const char* maxConfidenceLabel;
void runClassifier()
{
  bool m = microphone_inference_record();
  if (!m) {
    return;
  }
```

```
signal t signal;
  signal.total_length = EI_CLASSIFIER_SLICE_SIZE;
  signal.get_data = &microphone_audio_signal_get_data;
  ei_impulse_result_t result = { 0 };
  EI_IMPULSE_ERROR r = run_classifier_continuous(&signal, &result, false);
  if (r != EI IMPULSE OK) {
   return;
  }
  float maxValue = 0;
  for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {</pre>
    ei_impulse_result_classification_t classification_t =
result.classification[ix];
                  %s: %.5f\n", classification t.label, classification t.value);
    ei printf("
    float value = classification t.value;
    if (value > maxValue) {
      maxValue = value;
      maxConfidenceLabel = classification t.label;
    }
    currentClassification[ix] = classification t;
  }
}
void setup(){
  tft.begin();
  run_classifier_init();
  inference.buffers[0] = (int16_t *)malloc(EI_CLASSIFIER_SLICE_SIZE *
sizeof(int16_t));
  if (inference.buffers[0] == NULL) {
    return;
  }
  inference.buffers[1] = (int16_t *)malloc(EI_CLASSIFIER_SLICE_SIZE *
sizeof(int16_t));
  if (inference.buffers[1] == NULL) {
    free(inference.buffers[0]);
    return;
  }
  inference.buf_select = 0;
  inference.buf count = 0;
  inference.n samples = EI CLASSIFIER SLICE SIZE;
  inference.buf_ready = 0;
  config dma adc();
  recording = 1;
  tft.setRotation(3);
  tft.setTextSize(4);
```

}

```
void loop(){
  runClassifier();
  if (maxConfidenceLabel == "hello_wio") {
    tft.drawString((String)"Hello Wio", 50, 110);
    delay(3000);
  } else {
    tft.fillScreen(0x0);
    }
}
Step 5: Upload the code.
```



It takes about 5 mins to upload. If the upload is successful, the message "Done uploading." will appear in the status bar.

Step 6: Say "Hello Wio" to the Wio Terminal to see whether it has been woken up.



Reference Edge Impulse Public project: https://studio.edgeimpulse.com/public/77128/latest

Practice 4.People counting using Ultrasonic sensor

Project Overview

In this project, we will create a people counting system by using Wio Terminal, an ordinary Ultrasonic ranger and a special Deep Learning sauce to top it off and actually make it work.

Material Preparation

Hardware requirements: Wio Terminal

Connection method:

Attach Wio terminal and Ultrasonic sensor with screws to wooden or 3D printed frame,

example below:



To put the frame on the wall, 3M velcro strips were used.

Additional options include using foam tape, screws or nails.

About sensor

First, let's understand the data we can get from the Ultrasonic sensor and how we can utilize it for determining the direction of objects.

This Grove - Ultrasonic ranger is a non-contact distance measurement module that works at 40KHz. When we provide a pulse trigger signal with more than 10uS through the signal pin, the Grove_Ultrasonic_Ranger will issue 8 cycles of 40kHz cycle level and detect the echo. The pulse width of the echo signal is proportional to the measured distance. Here is the formula: Distance =echo signal high time * Sound speed (340M/S)/2.



Now, use this Grove - Ultrasonic ranger. We can immediately see that for walking in, we get relatively high values(corresponding to distance from the object) first, which then decrease. And for walking out, we get completely opposite signal.



Theoretically, we could write an algorithm ourselves by hand, that can determine the direction. Unfortunately, real-life situations are complicated – we have people, that walk fast(shorter curve length) and slow (longer curve length), we have thinner people and people who are... not so thin and so on. So our hand-written algorithm needs to take all of these into account, which will inevitably make it complicated and convoluted. We have a task involving inference signal processing and lots of noisy data with significant variations... And the solution is — Deep Learning.

Warning

Do not hot-plug Grove-Ultrasonic-Ranger, otherwise, it will damage the sensor. The the measured area must be no less than 0.5 square meters and smooth.

Machine Learning Lifecycle Please Open: <u>https://www.edgeimpulse.com/</u> **Data Collection** Step1: <u>Connect Wio Terminal with Edge Impulse</u> Step2: Know what we are going to do We will train and deploy a simple neural network that can distinguish between people entering or exiting a room using only ultrasonic rangers. So we need to select the sensor we are going to use – Grove-Ultrasonic-Ranger, then know what kind of data we are going to sample --people in and people out.

Select the sense	or we are going	l to use Grov	e-Ultrasonic-Range	r.

Record new data	
Device 🕲	
33:68:FF:19:11:3C	~
Label	Sample length (ms.)
Label name	5000
Sensor	Frequency
External ultrasonic sensor(Grove-ultrasonic sensor)	21Hz 🗸
Built-in accelerometer Built-in microphone	
Built-in light sensor	Start sampling
External multichannel gas(Grove-multichannel gas v2)	
External temperature&humidity&pressure sensor(Grove-BME280)	
External pressure sensor(Grove-DPS310) External distance sensor(Grove-TFmini)	
External 6-axis accelerometer(Grove-BMI088)	
External ultrasonic sensor(Grove-ultrasonic sensor)	
External CO2+Temp sensor(Grove-SCD30)	

This indicates that we want to record data for 5 seconds (Sample length 5000ms), use a

built-in microphone and frequency 21Hz.

Know the data we are going to sample -- people in and people out

Walking in





None(walking near the device, not getting closer or further away from it)



Step3: Sample

Record new data	
Device ⑦	
33:68:FF:19:11:3C	~
Label	Sample length (ms.)
in	5000
Sensor	Frequency
External ultrasonic sensor(Grove-ultrasonic sensor)	✓ 21Hz ✓
	Start sampling

Enter the label, click "Start Sampling",For this lesson, we recorded 1 minute 30 seconds of data for every class, each time recording 5000 ms samples and then cropping them to get 1500 ms samples – remember that variety is very important in the dataset, so make sure you have samples where you (or other people) walk fast, slow, run, etc.

Walking in



Walking out



None



For none category apart from samples that have nobody in front of the device, it is a good idea to include samples that have a person just standing close to the device and walking beside it, to avoid any movement being falsely classified as in or out.

Impulse Design

When we are done with data collection, create your impulse – set window length to 1500 ms and windows size increase to 500 ms.

Time series 🔳	Raw Data 🕟	Neural A Network	Output features
Axes	Name	(Keras)	3 (in, none, out)
us	Raw data	Name	
Window size 🕐	Input axes	NN Classifier	Save Impul
	US.	Input features	save unput
1500 ms.	-	💌 Raw data	
Window Increase (2)		Output features	



Model Training--Network: CNN

The best results were achieved by tweaking network architecture a bit to obtain 92% accuracy, for that, you will need to switch to "expert" mode and change MaxPool1D strides to 1 and pool size to 4.



How good is 92% accuracy and what can be done to improve it?

92% is fairly good as proof of concept or prototype, but horrible as a production model. For production, the mileage may vary – if your application is critical and somehow used in automated control and decision making, you don't really want to have anything below 98 – 99 per cent and even that might be low, think about something like a face recognition system for payment or authentication. Are there ways to improve the accuracy of this system? The ultrasonic sensor is a cheap and ubiquitous sensor, but it is relatively slow and not very precise. We can get better data by using Grove TF Mini LiDAR Module.



• Get more data and possibly place the sensor lower, at normal human waist level to make sure it can detect shorter than normal height people and children.

• Two are better than one – having two sensors taking measurements at slightly different places will not add too much data (we only have 31 data points in each sample) but might increase the accuracy. To explore more interesting ideas, a built-in light sensor can be used if Wio Terminal is appropriately located.

Once the model is trained we can perform live classification with data from the device – here we found that a window size increase of 500 ms actually doesn't work that well and produces more false positives, so at the next step when deploying to the device, it is better to increase the value to 750 ms.

Model Optimization

Model Deployment

The next step is deployment on the device.

Step 1: After clicking on the Deployment tab, choose Arduino library and download it.



Step 2: Now, the library can be installed to the Arduino IDE. Open the Arduino IDE, click sketch -> Include Library -> Add .ZIP Library, and choose the file that you have just

downloaded.

É Arduino File Edit	Sketch Tools Help	
	Verify/Compile #R Upload #U	Manage Libraries
	Upload Using Programmer 企業U	Add .ZIP Library
Test.ino	Export compiled Binary \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\	Arduino libraries
<pre>1 void setup() { 2 // put your setup code here</pre>	Show Sketch Folder #K	Bridge
3	Include Library	Esplora
4 } 5	Add File	Ethernet

Step 3: Open Examples -> name of your project -> static buffer.



Step 4: Copy the following Example Code to replace the original example code:

This time we will be using continuous inference examples to make sure we are not missing

any important data.

```
#include <people_counter_inferencing.h>
#include <Seeed_Arduino_FreeRTOS.h>
#include "Ultrasonic.h"
#include "TFT_eSPI.h"
#include <lvgl.h>
#define ERROR_LED_LIGHTUP_STATE HIGH
#define LVGL_TICK_PERIOD 10
/* Private variables ------ */
static bool debug_nn = false; // Set this to true to see e.g. features generated
from the raw signal
static uint32_t run_inference_every_ms = 500;
static float buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = {0};
static float inference_buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE];
float distance;
uint8_t axis_num = 1;
int16_t peopleCount = 0;
uint16_t peopleIn = 0;
uint16_t peopleOut = 0;
```

```
lv_obj_t *LogOutput;
lv_obj_t *peopleInLabel;
lv_obj_t *peopleOutLabel;
lv_obj_t *peopleNumLabel;
const char *prev_prediction = "none";
TaskHandle_t Handle_aTask;
TaskHandle_t Handle_bTask;
TaskHandle_t Handle_cTask;
Ultrasonic ultrasonic(0);
TFT_eSPI tft;
static lv disp buf t disp buf;
static lv color t buf[LV HOR RES MAX * 10];
/**
* @brief
              Arduino setup function
*/
void setup()
{
    pinMode(WIO_KEY_A, INPUT_PULLUP);
    pinMode(WIO_KEY_B, INPUT_PULLUP);
    pinMode(WIO_KEY_C, INPUT_PULLUP);
    lv_init();
    tft.begin();
    tft.setRotation(3);
    // put your setup code here, to run once:
    Serial.begin(115200);
    lv_disp_buf_init(&disp_buf, buf, NULL, LV_HOR_RES_MAX * 10);
    lv_disp_drv_t disp_drv;
    lv_disp_drv_init(&disp_drv);
    disp_drv.hor_res = 320;
    disp_drv.ver_res = 240;
    disp_drv.flush_cb = my_disp_flush;
    disp drv.buffer = &disp buf;
    lv_disp_drv_register(&disp_drv);
    lv_buttons();
    // Enter configuration mode
    if (EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME != axis_num) {
        ei_printf("ERR: EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME should be equal to
(%d) (the (%d) sensor axes)\n", axis_num, axis_num);
        return;
    }
```
```
vSetErrorLed(LED_BUILTIN, ERROR_LED_LIGHTUP_STATE);
   // Create the threads that will be managed by the rtos
    // Sets the stack size and priority of each task
    // Also initializes a handler pointer to each task, which are important to
communicate with and retrieve info from tasks
    xTaskCreate(lv_tick_task, "LVGL Tick", 128, NULL, tskIDLE_PRIORITY + 1,
&Handle_aTask);
    xTaskCreate(run inference background,"Inference", 512, NULL,
tskIDLE_PRIORITY + 1, &Handle_bTask);
    xTaskCreate(read_data, "Data collection", 256, NULL, tskIDLE_PRIORITY + 2,
&Handle_cTask);
   // Start the RTOS, this function will never return and will schedule the
tasks.
   vTaskStartScheduler();
}
/**
* @brief
              Printf function uses vsnprintf and output using Arduino Serial
* @param[in] format
                       Variable argument list
*/
void update_screen()
{
 peopleCount = peopleIn - peopleOut;
 lv_label_set_text_fmt(peopleInLabel, "%d", peopleIn);
 lv_label_set_text_fmt(peopleOutLabel, "%d", peopleOut);
 lv_label_set_text_fmt(peopleNumLabel, "%d", peopleCount);
 lv_task_handler();
}
static void lv_tick_task(void* pvParameters) {
   while(1){
     lv_tick_inc(LVGL_TICK_PERIOD);
     delay(LVGL_TICK_PERIOD);
    }
}
static void DisplayPrintf(const char* format, ...)
{
   va_list arg;
    va start(arg, format);
    String str{StringVFormat(format, arg)};
    va_end(arg);
    Log("%s\n", str.c_str());
    lv_label_set_text(LogOutput, str.c_str());
   lv_task_handler();
}
```

```
void my_disp_flush(lv_disp_drv_t *disp, const lv_area_t *area, lv_color_t
*color p)
{
 uint16_t c;
 tft.startWrite(); /* Start new TFT transaction */
 tft.setAddrWindow(area->x1, area->y1, (area->x2 - area->x1 + 1), (area->y2 -
area->y1 + 1)); /* set the working window */
 for (int y = area->y1; y <= area->y2; y++) {
    for (int x = area->x1; x <= area->x2; x++) {
     c = color_p->full;
     tft.writeColor(c, 1);
     color p++;
   }
 }
 tft.endWrite(); /* terminate TFT transaction */
 lv_disp_flush_ready(disp); /* tell lvgl that flushing is done */
}
void lv buttons(void)
{
   lv_obj_t *peopleInDisplay = lv_btn_create(lv_scr_act(), NULL); /*Add a
button the current screen*/
                                                                       /*Set
   lv_obj_set_pos(peopleInDisplay, 20, 60);
its position*/
   lv_obj_set_size(peopleInDisplay, 120, 50);
                                                                        /*Set
its size*/
    peopleInLabel = lv_label_create(peopleInDisplay, NULL);
                                                                    /*Add a
label to the button*/
                                                             /*Set the labels
    lv_label_set_text(peopleInLabel, "0");
text*/
   lv_obj_t *peopleOutDisplay = lv_btn_create(lv_scr_act(), NULL);
                                                                      /*Add a
button the current screen*/
    lv_obj_set_pos(peopleOutDisplay, 180, 60);
                                                                          /*Set
its position*/
   lv_obj_set_size(peopleOutDisplay, 120, 50);
                                                                         /*Set
its size*/
    peopleOutLabel = lv label create(peopleOutDisplay, NULL);
                                                                      /*Add a
label to the button*/
                                                              /*Set the labels
    lv label set text(peopleOutLabel, "0");
text*/
   lv_obj_t *peopleNumDisplay = lv_btn_create(lv_scr_act(), NULL);
                                                                      /*Add a
button the current screen*/
                                                                          /*Set
    lv obj set pos(peopleNumDisplay, 90, 160);
its position*/
    lv_obj_set_size(peopleNumDisplay, 140, 70);
                                                                         /*Set
its size*/
    peopleNumLabel = lv label create(peopleNumDisplay, NULL);
                                                                       /*Add a
label to the button*/
                                                              /*Set the labels
    lv_label_set_text(peopleNumLabel, "0");
```

```
text*/
    LogOutput = lv_label_create(lv_scr_act(), NULL);
    lv_label_set_long_mode(LogOutput, LV_LABEL_LONG_BREAK);
                                                               /*Break the long
lines*/
    lv_label_set_recolor(LogOutput, true);
                                                                /*Enable
re-coloring by commands in the text*/
    lv_label_set_align(LogOutput, LV_LABEL_ALIGN_LEFT); /*Center aligned
lines*/
    lv_obj_set_width(LogOutput, 320);
    lv_obj_align(LogOutput, NULL, LV_ALIGN_IN_TOP_LEFT, 20, 10);
}
#define DLM "\r\n"
static String StringVFormat(const char* format, va_list arg)
{
    const int len = vsnprintf(nullptr, 0, format, arg);
    char str[len + 1];
    vsnprintf(str, sizeof(str), format, arg);
   return String{str};
}
static void Abort(const char* format, ...)
{
   va_list arg;
   va_start(arg, format);
   String str{ StringVFormat(format, arg) };
   va_end(arg);
   Serial.printf("ABORT: %s" DLM, str.c_str());
   while (true) {}
}
static void Log(const char* format, ...)
{
   va_list arg;
   va_start(arg, format);
   String str{StringVFormat(format, arg)};
   va_end(arg);
   Serial.print(str);
}
/**
 * @brief
               Run inferencing in the background.
*/
static void run inference background(void* pvParameters)
{
    // wait until we have a full buffer
    delay((EI_CLASSIFIER_INTERVAL_MS * EI_CLASSIFIER_RAW_SAMPLE_COUNT) + 100);
    // This is a structure that smoothens the output result
    // With the default settings 70% of readings should be the same before
classifying.
```

```
ei classifier smooth t smooth;
    ei_classifier_smooth_init(&smooth, 3 /* no. of readings */, 2 /* min.
readings the same */, 0.6 /* min. confidence */, 0.3 /* max anomaly */);
   while (1) {
        // copy the buffer
        memcpy(inference_buffer, buffer, EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE *
sizeof(float));
        // Turn the raw buffer in a signal which we can the classify
        signal t signal;
        int err = numpy::signal_from_buffer(inference_buffer,
EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &signal);
        if (err != 0) {
            Log("Failed to create signal from buffer (%d)\n", err);
            return;
        }
       // Run the classifier
        ei impulse result t result = \{0\};
        err = run_classifier(&signal, &result, debug_nn);
        if (err != EI_IMPULSE_OK) {
            Log("ERR: Failed to run classifier (%d)\n", err);
            return;
        }
        // print the predictions
        Log("Predictions ");
        Log("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
            result.timing.dsp, result.timing.classification,
result.timing.anomaly);
        Log(": ");
        // ei_classifier_smooth_update yields the predicted label
        const char *prediction = ei_classifier_smooth_update(&smooth, &result);
        Log("%s ", prediction);
       if (prediction != prev_prediction)
        {
        if (prediction == "out") {peopleOut++; DisplayPrintf("#ff00ff Person
left#");}
        if (prediction == "in") {peopleIn++; DisplayPrintf("#0000ff Person
entered#");}
        prev prediction = prediction;
        update_screen();
        }
        // print the cumulative results
        Log(" [ ");
        for (size t ix = 0; ix < smooth.count size; ix++) {</pre>
            Log("%u", smooth.count[ix]);
            if (ix != smooth.count_size + 1) {
                Log(", ");
            }
            else {
              Log(" ");
```

```
}
        }
        Log("]\n");
        delay(run_inference_every_ms);
    }
    ei_classifier_smooth_free(&smooth);
}
/**
* @brief
              Get data and run inferencing
* @param[in] debug Get debug info if true
*/
static void read data(void* pvParameters)
{
    while (1) {
        // Determine the next tick (and then sleep later)
        uint64_t next_tick = micros() + (EI_CLASSIFIER_INTERVAL_MS * 1000);
        // roll the buffer -axis_num points so we can overwrite the last one
        numpy::roll(buffer, EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, -axis_num);
        distance = ultrasonic.MeasureInCentimeters();
        if (distance > 200.0) { distance = -1;}
        buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE - 1] = distance;
        // and wait for next tick
        uint64_t time_to_wait = next_tick - micros();
        delay((int)floor((float)time_to_wait / 1000.0f));
        delayMicroseconds(time_to_wait % 1000);
    }
}
void loop()
{
 //nothing, all the work is done in two threads
}
```

If you remember, in the first Practice, for the inference, we would collect all the data points in the sample, perform the inference and then go back to sampling – that means that when feeding the data to the neural network we would pause the data collection and lose some of the data.



That is not optimal and we can use either DMA (Direct Memory Access), threading or multiprocessing to fix this issue.



Fig: Showing DMA Mode of Data Transfer

Wio Terminal MCU (Cortex M4F core) only has one core, so multiprocessing is not an option – so in this case, we will use FreeRTOS and threads. What is going to happen is that during the inference process, FreeRTOS will pause inference for a brief moment, collect the data sample and then go back to inference



This way the actual inference will take a little longer, but the difference is negligible for this particular use case. We perform inference every 500 ms, so every 500 ms slice of the time window will be performed inference on for 3 times. Then we take the result that has the highest confidence across 3 inferences – for example, we have the highest confidence for "out" label 2 times and for "none" label one time, thus the result should be classified as "out". To simplify the testing we will add the lines that turn on Wio Terminal screen when a person is entering the room and turn it off when a person exits.

Step 5: Upload the code.



It takes about 5 mins to upload. If the upload is successful, the message "Done uploading." will appear in the status bar.

Step 6:



Reference

Edge Impulse Public project

https://studio.edgeimpulse.com/public/18808/latest

Practice 5. Anomaly detection using Grove BME280

Project Overview

In this project, we will use data from BME280, perform anomaly detection on-device.

Anomalies. Or specifically anomaly detection for predictive maintenance.

Some workshops will have requirements for specific range of temperature, humidity and air pressure because the abnormal environments will have adverse effects on their products. Similarly, greenhouse planting, the breeding hatchery has requirements for these three indices, a good environment helps its planting and hatching.

In these situations, we really just want our model to be able to interpret all the data as "normal" and "abnormal". It doesn't matter what are the exact characteristics of "abnormal" – they can be wildly different, the important thing is, if the "abnormal" class is detected, some measures need to be implemented. What I described now is the premise behind using Machine Learning for predictive maintenance. We monitor the state of the device or a place, be it an air conditioner, water pump or other machinery with sensors and based on the profile of known "normal" operation, try to detect when something goes SLIGHTLY wrong before it goes SERIOUSLY wrong.



Material Preparation

Hardware requirements: Wio Terminal

Connection method:



About sensor

Grove BME280 provides a precise measurement of not only barometric pressure and temperature, but also the humidity in the environment. The air pressure can be measured in a range from 300 hPa to 1100hPa with ± 1.0 hPa accuracy, while the sensor works perfectly for temperatures between - 40°C and 85°C with an accuracy of $\pm 1^{\circ}$ C. As for the humidity, you can get a humidity value with an error of less than 3%.

Owing to its high accuracy in measuring the pressure, and the pressure changes with altitude, we can calculate the altitude with ±1 meter accuracy, which makes it a precise altimeter as well.

Machine Learning Lifecycle

Please Open: https://www.edgeimpulse.com/

Data Collection

Step1: Connect Wio Terminal with Edge Impulse

Step2: Know what we are going to do

We will train and deploy a simple neural network that is able to interpret all the data as "normal" and "abnormal" using BME280. So we need to select the sensor we are going to use – Grove-BME280, then know what kind of data we are going to sample --data of normal state.

Select the sensor we are going to use -- Grove-BME280.

~
ple length (ms.)
000
uency
2.5Hz ~
Charles and a line
Start sampling
0)

This indicates that we want to record data for 20 seconds (Sample length 20000ms), use

Grove-BME280 and frequency 62.5Hz.

Know the data we are going to sample

The workshops have requirements for specific range of temperature, humidity and air

pressure because the abnormal environments will have adverse effects on their products.

We want to sample data that is in its normal state.

Step3: Sample

Record new data	
Device ③	
33:68:FF:19:11:3C	~
Label	Sample length (ms.)
normal	20000
Sensor	Frequency
External temperature&humidity&pressure sensor((62.5Hz ~
	Start sampling

Enter the label, click "Start Sampling".

Now, we have recorded around 2 minutes of data:

Training data	Test data	Export data
DATA COLLECT	TED	0

Impulse Design

When we are done with data collection, create our impulse – set window length to 1000 ms and windows size increase to 1000 ms.

Time series data	Spectral Analysis	Anomaly Detection (K-	Output features
Input axes (3)	Name	Name	1 (Anomaly score)
Temp, Pressure, Humidity	Spectral features	Anomaly detection	
Window size ⑦	Input axes (3)	Input features	Save Impulse
1000 ms.	✓ Temp	Spectral features	
Window increase 🕜	✓ Pressure	Output features	
	Humidity	1 (Anomaly score)	
Frequency (Hz) ⑦			
62.5 C	î		
Zero-pad data ⑦ ✔			
ت بر			

Feature Extraction--Spectral Analysis

The only significant tweak I made was changing the filter from low to high, which made the features more prominent.

Parameters		
Scaling		
Scale axes	1	
Filter		
Туре	high	~
Cut-off frequency	3	
Order	6	
Spectral power		
FFT length	128	
No. of peaks	3	
Peaks threshold	0.1	
Power edges	0.1, 0.5, 1.0, 2.0, 5.0	

Model Training--Network: Anomaly detection





We trainin a network that creates 10 clusters around data that we have seen before and compares incoming data against these clusters. If the distance from a cluster is too large the sample has flagged the sample as an anomaly.

After trial and error, I found that a very low cluster count works the best for anomaly detection, but this is very case-specific and depends on your data.

Model Optimization

Model Deployment

The next step is deployment on the device.

Step 1: After clicking on the Deployment tab, choose Arduino library and download it.



Step 2: Now, the library can be installed to the Arduino IDE. Open the Arduino IDE, click sketch -> Include Library -> Add .ZIP Library, and choose the file that you have just downloaded.

É Arduino File Edit	Sketch Tools Help		
	Verify/Compile Upload	ЖR ЖU	Manage Libraries
	Upload Using Programmer	企業U	Add .ZIP Library
Test.ino	Export compiled Binary	τ₩S	Arduino libraries
<pre>1 void setup() { 2 // put your setup code here.</pre>	Show Sketch Folder	ЖК	Bridge
3	Include Library		Esplora
4 } 5	Add File		Ethernet

Step 3: Open Examples -> name of your project -> static buffer.



Step 4: Copy the following Example Code to replace the original example code:

```
#define ANOMALY_THRESHOLD 30
#include "Seeed_BME280.h"
#include <Wire.h>
```

```
#include <Anomaly_detection_BME280_inferencing.h>
#include "TFT_eSPI.h"
TFT_eSPI tft;
BME280 bme280;
static bool debug_nn = false; // Set this to true to see e.g. features generated
from the raw signal
void setup()
{
 Serial.begin(115200);
 tft.begin();
 tft.setRotation(3);
 if(!bme280.init()){
   Serial.println("Failed to initialize IMU!");
   while (1);
 }
 else {
    ei_printf("IMU initialized\r\n");
 }
 if (EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME != 3) {
        ei_printf("ERR: EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME should be equal to 3
(the 3 sensor axes)\n");
        return;
 }
}
/**
* @brief
              Printf function uses vsnprintf and output using Arduino Serial
*
* @param[in] format
                         Variable argument list
*/
void ei_printf(const char *format, ...) {
   static char print_buf[1024] = { 0 };
  va list args;
   va_start(args, format);
   int r = vsnprintf(print_buf, sizeof(print_buf), format, args);
  va_end(args);
   if (r > 0) {
      Serial.write(print_buf);
   }
}
void loop()
{
```

```
float buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = { 0 };
 for (size_t ix = 0; ix < EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE; ix += 3) {</pre>
        // Determine the next tick (and then sleep later)
        uint64_t next_tick = micros() + (EI_CLASSIFIER_INTERVAL_MS * 1000);
        buffer[ix + 0] = bme280.getTemperature();
        buffer[ix + 1] = bme280.getPressure()/100;
        buffer[ix + 2] = bme280.getHumidity();
        delayMicroseconds(next_tick - micros());
    }
    // Turn the raw buffer in a signal which we can the classify
    signal t signal;
    int err = numpy::signal_from_buffer(buffer,
EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &signal);
    if (err != 0) {
        ei printf("Failed to create signal from buffer (%d)\n", err);
        return;
    }
    // Run the classifier
    ei_impulse_result_t result = { 0 };
    err = run_classifier(&signal, &result, debug_nn);
    if (err != EI_IMPULSE_OK) {
        ei_printf("ERR: Failed to run classifier (%d)\n", err);
        return;
    }
    // print the predictions
    ei_printf("Predictions ");
    ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
        result.timing.dsp, result.timing.classification, result.timing.anomaly);
    ei printf(": \n");
    for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {</pre>
        ei_printf("
                     %s: %.5f\n", result.classification[ix].label,
result.classification[ix].value);
    }
#if EI_CLASSIFIER_HAS_ANOMALY == 1
                  anomaly score: %.3f\n", result.anomaly);
    ei printf("
 if (result.anomaly > ANOMALY THRESHOLD)
  {
   tft.fillScreen(TFT RED);
   tft.setFreeFont(&FreeSansBoldOblique12pt7b);
   tft.drawString("Anomaly detected", 40, 110);
   delay(1000);
   tft.fillScreen(TFT_WHITE);
 }
```

```
#endif
Serial.print("Temp: ");
Serial.print(bme280.getTemperature());
Serial.println("C");//The unit for Celsius because original arduino don't
support speical symbols
//get and print atmospheric pressure data
Serial.print("Pressure: ");
Serial.print(bme280.getPressure());
Serial.print(bme280.getPressure());
Serial.println("Pa");
//get and print humidity data
Serial.print(bme280.getHumidity());
Serial.print(bme280.getHumidity());
Serial.println("%");
}
```

Step 5: Upload the code.



If the upload is successful, the message "Done uploading." will appear in the status bar.

Step 6: Try to simulate an abnormal situation and see whether the Wio Terminal alarms.



Reference

Edge Impulse Public project: https://studio.edgeimpulse.com/public/76507/latest