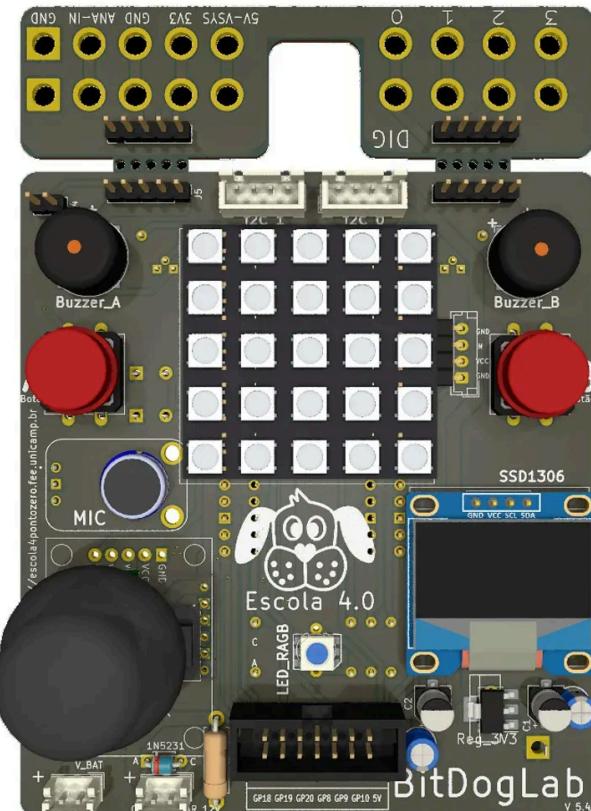




EMBARCATEH - CAMPINAS - SP



Projeto:

*Sistema para Monitoramento Inteligente de Movimentos em
Contêineres Marítimos com TinyML*

Adriana Rocha Castro de Paula

Elias Kento Tomiyama

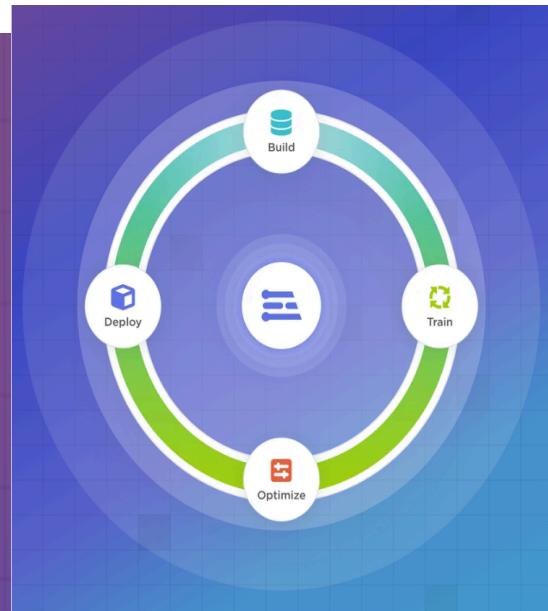
Vagner S. Vasconcelos

Setembro de 2025



AI for Any Edge Device

Sensors & Cameras



Build datasets, train models, and optimize libraries to run directly on device; from the smallest microcontrollers to gateways with the latest neural accelerators (and anything in between).



Introdução

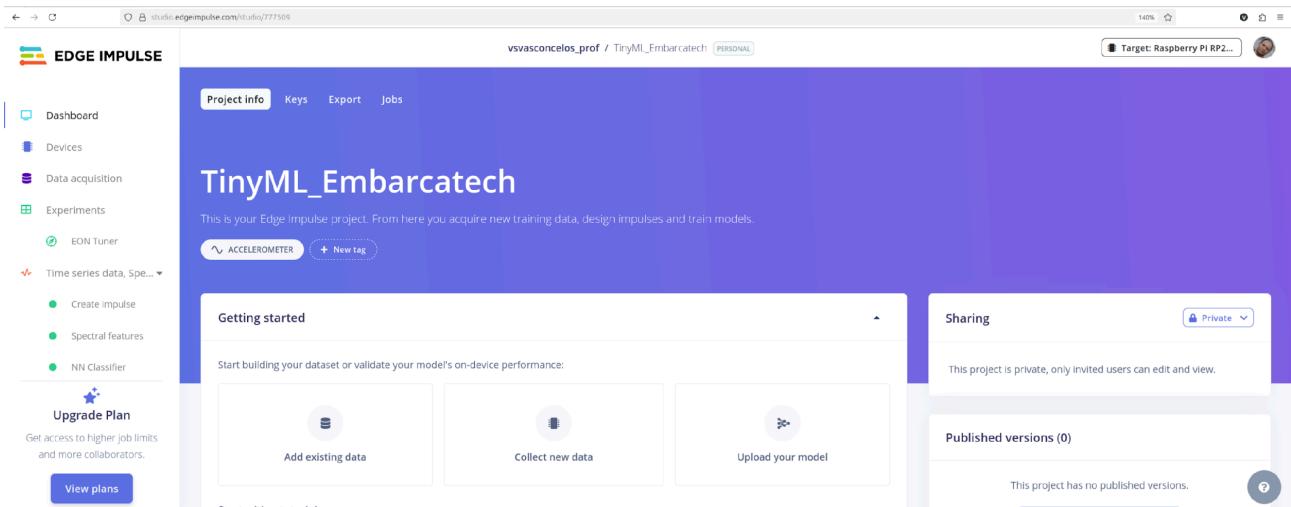
O primeiro passo é a criação de uma conta na plataforma Edge Impulse (EI) <<https://edgeimpulse.com/>>. O processo (pipeline) na plataforma inclui a coleta de dados, a extração de recursos, o treinamento de um modelo e a implantação desse modelo em um sistema incorporado. Classificaremos os dados de aceleração em 4 classes, e queremos ser capazes de determinar em qual delas um contêiner marítimo está.

Hardware necessário

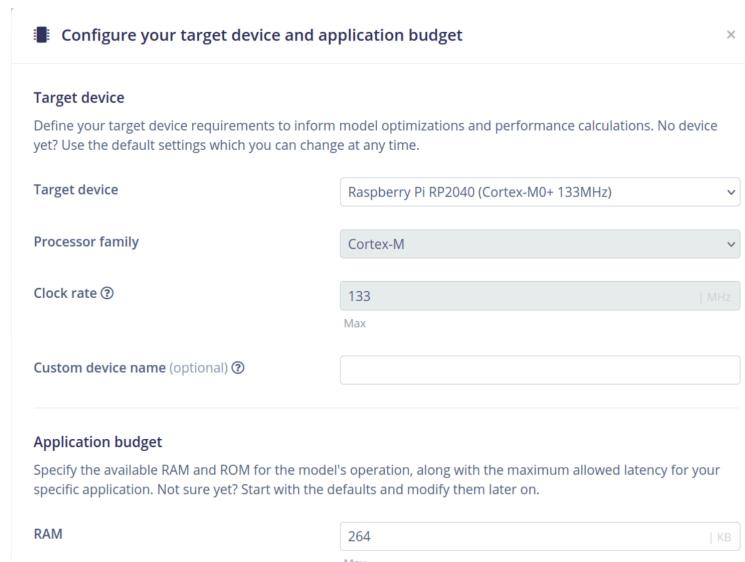
Será utilizado a BitDoglab com o acelerômetro MPU6500/MPU6050.

Conectar a BitDogLab (RP2040) ao Edge Impulse

Criamos um novo projeto no Edge Impulse com o nome TinyML_Embarcatech <<https://studio.edgeimpulse.com/studio/777509>>

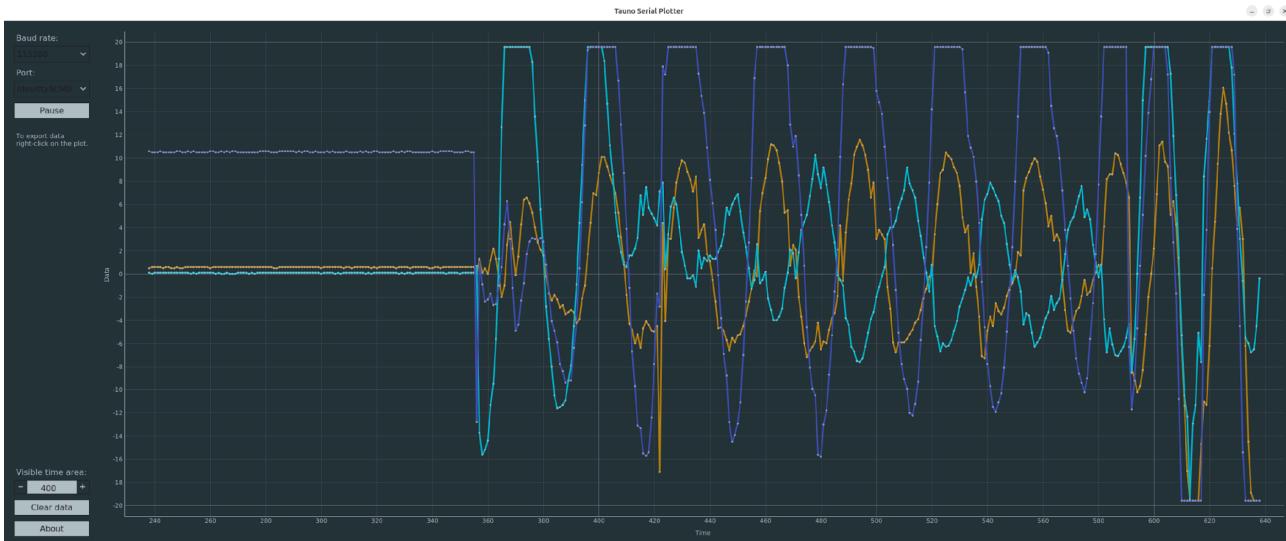


Um detalhe importante, como pode ser observado do lado direito superior da figura acima, já configuramos as características do modelo de ML para a RP2040. Os detalhes da configuração são apresentados na figura abaixo.



Para detalhes de conexão do RP2040, verifique o link <https://docs.edgeimpulse.com/hardware/boards/raspberry-pi-pico> onde é detalhado a instalação do Edge Impulse CLI <https://docs.edgeimpulse.com/tools/clis/edge-impulse-cli/installation> que é o aplicativo desktop para conexão dos dispositivos (devices) na plataforma Edge Impulse.

Nesta etapa, o firmware para a aquisição dos sinais do acelerômetro já devem estar funcionando na BitDogLab, conforme exemplo na figura abaixo (Tauno-Serial-Plotter) <https://taunoerik.wordpress.com/2021/01/11/tauno-serial-plotter/> enviando os dados de aceleração para a saída serial, isto, a BitDogLab deve estar conectada no desktop.



Para iniciar a conexão, no console do desktop , digitar o comando:
edge-impulse-data-forwarder, conforme figura abaixo:

```
vagner@Lenovo:build$ edge-impulse-data-forwarder --clean
Edge Impulse data forwarder v1.34.1
? What is your user name or e-mail address (edgeimpulse.com)? vsvasconcelos_prof
? What is your password? [hidden]
Endpoints:
  Websocket: wss://remote-mgmt.edgeimpulse.com
  API: https://studio.edgeimpulse.com
  Ingestion: https://ingestion.edgeimpulse.com

[SER] Connecting to /dev/ttyACM0
[SER] Serial is connected (E6:61:41:04:03:5D:33:2E)
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com

[SER] Detecting data frequency...
[SER] Detected data frequency: 100Hz
? 3 sensor axes detected (example values: [0.6,0.1,10.5]). What do you want to call them? Separate the names with ',': accX, accY, accZ
[WS ] Device "BitDogLab" is now connected to project "TinyML_Embarcatech". To connect to another project, run `edge-impulse-data-forwarder --clean`.
[WS ] Go to https://studio.edgeimpulse.com/studio/777509/acquisition/training to build your machine learning model!
```

Detalhe: na figura, observa-se o complemento do comando acima “-- clean”, isso só se fez necessário para refazer a conexão que já existia, na 1ª conexão não é necessário.

Após o comando, é solicitado login e senha da plataforma EI, o aplicativo detecta a frequência do sinal (100HZ neste exemplo) e se tratar de um sensor de 3 eixos, solicitando a entrada de nomes para cada um deles, entramos como accX (aceleração no eixo X), accY e accZ. Como já tínhamos definido um nome para o sensor na plataforma EI, o aplicativo já o

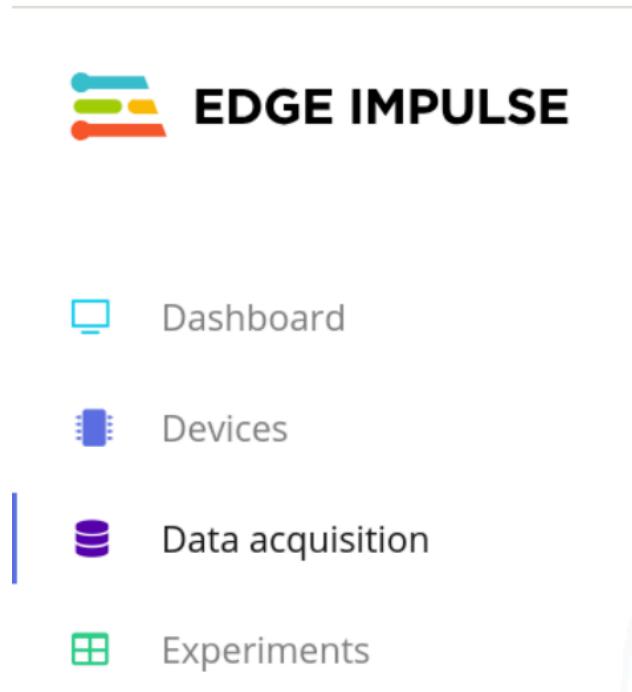
utilizou (BitDogLab), se não tivéssemos entrando já com o nome, o aplicativo solicitaria um. Na sequência, clicando no link final, mostrado na figura acima <<https://studio.edgeimpulse.com/studio/777509>>, isso nos leva diretamente para a página principal do projeto no EI.

Nela, do lado esquerdo superior, clicando em “Devices”, identificamos a BitDogLab conectada.

The screenshot shows the Edge Impulse Studio interface at the URL <https://studio.edgeimpulse.com/studio/777509/devices>. The left sidebar has a 'Devices' item selected. The main content area is titled 'Your devices' and displays a single entry: 'BitDogLab'. A note below says, 'These are devices that are connected to the Edge Impulse remote management API, or have posted data to the ingestion SDK.' The 'BitDogLab' entry includes a small icon, the device name, and a status message: 'Connected to data acquisition (Sensor with 3 axes (accX, accY, accZ))'.

Coleta de dados

Com o dispositivo conectado a plataforma, podemos iniciar a etapa de aquisição de dados, para isso, basta clicar em “Data acquisition”, do lado direito, logo abaixo de “Devices”.



Nesta tela, do lado esquerdo, temos o campo para aquisição de dados “Collect data”.

O campo “Label” deve ser preenchido (rotulação) conforme o tipo de movimento está sendo aquisitado.

Collect data

Device ?

BitDogLab

Label

Label name

Sample length (ms.)

10000

Sensor

Sensor with 3 axes (accX, accY, accZ)

Frequency

100Hz

Start sampling

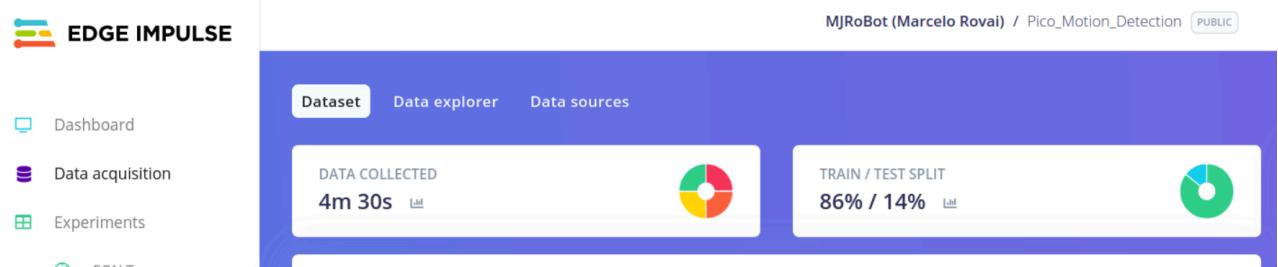
RAW DATA

Click on a sample to load...

Deve-se aquisitar dados para todas as classes de movimento envolvidas, em quantidades iguais ou muito próximas para cada uma delas (dados balanceados), em tese, quanto mais dados melhor.

Para simplificar o processo e ainda verificar o impacto na utilização de dados coletados com um acelerômetro diferente, foram utilizados os dados do projeto “Pico_Motion_Detection”

<<https://studio.edgeimpulse.com/public/20571>>, de autoria de Marcelo Rovai, onde foi aquisitado 4m30s de dados, sendo utilizados 86% deles para treinamento do modelo de ML e 14% para testes.

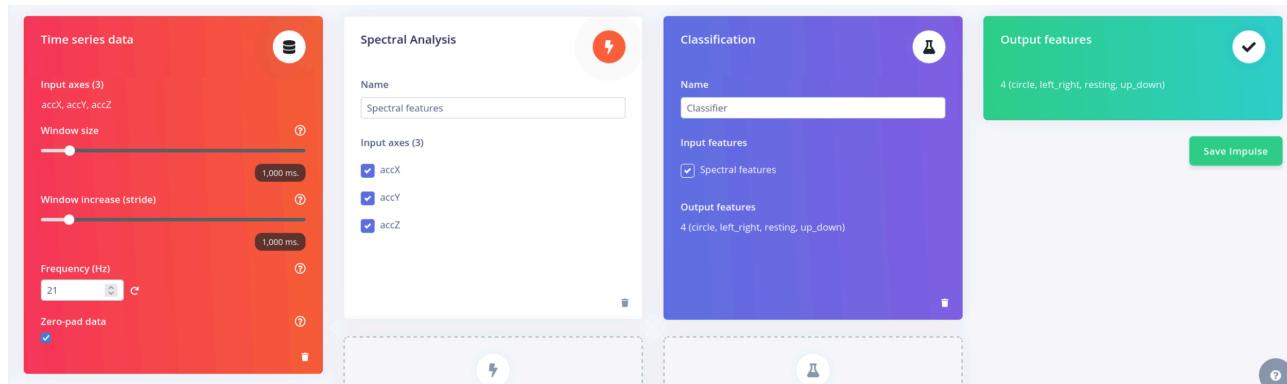


Clicando no simbolo de histograma (ao lado do 4m30s), observa-se as 4 classes rotuladas: “circle”; “left_right”; “resting”; e “up-down”. Um ponto importante a notar é o balanceamento dos dados, isso é, todas as classes possuem a mesma quantidade de dados.



Extração de recursos

Com os dados coletados, passamos para a etapa de extração de características (Feature Engineering), para isso, selecione a página de “Impulse design” e “Create impulse” no projeto. Adicione um bloco de “Time series data”, um de processamento “Spectral Analysis” e um de “Classification”, conforme figura abaixo.

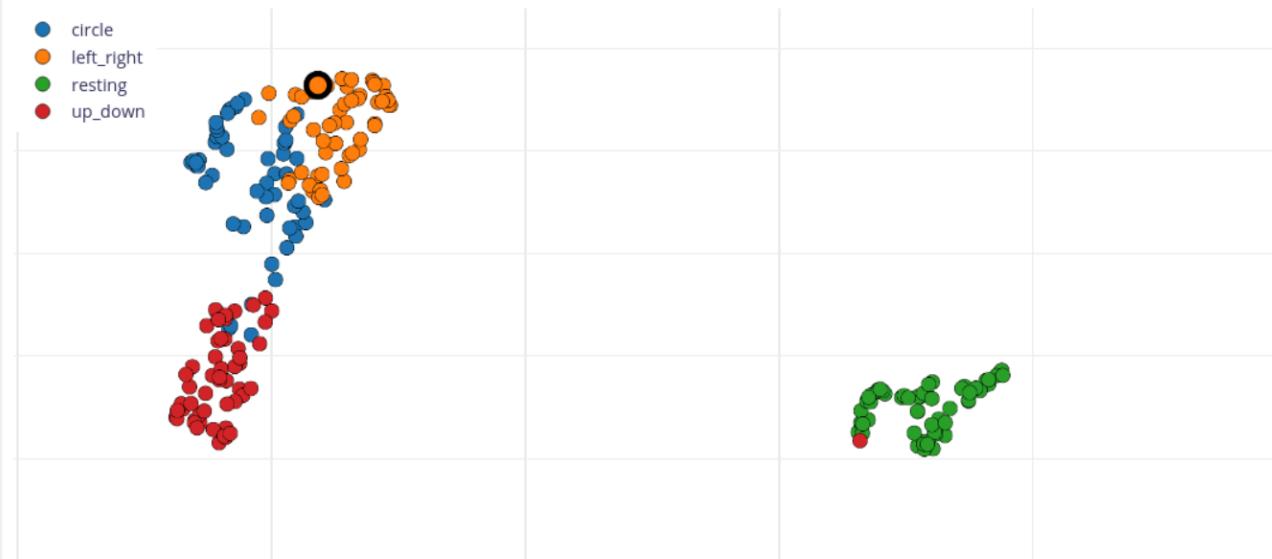


Após os ajustes, clique no botão “Salve impulse”, do lado direito.

Vá para a página “Spectral Features”, verifique os ajustes disponíveis, e realize os necessários, conforme as características do seu projeto, clique em “Generate features”.

Observe no *explorador de recursos*, atentando para que padrões você vê nas amostras. Suas classes podem ser facilmente separadas?

Feature explorer



Esse é um processo iterativo, deve-se ir ajustando os valores e observando os resultados, até se conseguir a melhor separação entre as classes.

Treinamento do modelo

Vá para a página “Classifier” em seu projeto, verifique os ajustes possíveis e faço a seleção conforme necessidades específicas do projeto.

The screenshot shows the Edge Impulse Classifier settings interface. On the left, a sidebar lists project components: Dashboard, Devices, Data acquisition, Experiments, EON Tuner, Time series data, Spectral features, Classifier (selected), Retrain model, and Live classification. At the bottom is an 'Upgrade Plan' section with a star icon. The main area is titled 'Neural Network settings' and contains sections for 'Training settings' and 'Advanced training settings'. Under 'Training settings', there are fields for 'Number of training cycles' (set to 30), 'Use learned optimizer' (unchecked), 'Learning rate' (set to 0.0005), and 'Training processor' (set to CPU). Under 'Advanced training settings', there is a dropdown menu. Below this is the 'Neural network architecture' section, which shows an input layer with 54 features and a single dense layer with 20 neurons. There are edit and delete icons for the architecture components.

Do lado direito de “Neural Network settings”, clicando nos 3 pontinhos, pode-se alterar para o modo expert, possibilitando maiores opções de ajustes na rede neural, conforme figura abaixo:

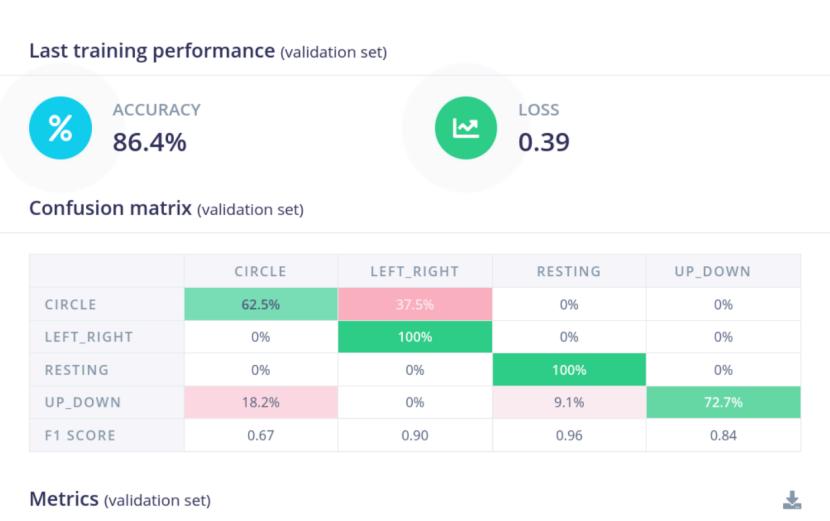
Neural network architecture

 Save

```
1 import tensorflow as tf
2 from tensorflow.keras.models import Sequential
3 from tensorflow.keras.layers import Dense, InputLayer, Dropout, Conv1D, Conv2D,
4     Flatten, Reshape, MaxPooling1D, MaxPooling2D, AveragePooling2D,
5     BatchNormalization, Permute, ReLU, Softmax
6 from tensorflow.keras.optimizers.legacy import Adam
7
8 EPOCHS = args.epochs or 30
9 LEARNING_RATE = args.learning_rate or 0.0005
10 # If True, non-deterministic functions (e.g. shuffling batches) are not used.
11 # This is False by default.
12 ENSURE_DETERMINISM = args.ensure_determinism
13 # this controls the batch size, or you can manipulate the tf.data.Dataset
14 # objects yourself
15 BATCH_SIZE = args.batch_size or 32
16 if not ENSURE_DETERMINISM:
17     train_dataset = train_dataset.shuffle(buffer_size=BATCH_SIZE*4)
18     train_dataset=train_dataset.batch(BATCH_SIZE, drop_remainder=False)
19     validation_dataset = validation_dataset.batch(BATCH_SIZE, drop_remainder=False)
20
21 # model architecture
22 model = Sequential()
23 model.add(Dense(20, activation='relu',
24                 activity_regularizer=tf.keras.regularizers.l1(0.00001)))
25 model.add(Dense(10, activation='relu',
26                 activity_regularizer=tf.keras.regularizers.l1(0.00001)))
27 model.add(Dense(classes, name='y_pred', activation='softmax'))
28
29 # this controls the learning rate
30 opt = Adam(learning_rate=LEARNING_RATE, beta_1=0.9, beta_2=0.999)
31 callbacks.append(BatchLoggerCallback(BATCH_SIZE, train_sample_count, epochs
32 =EPOCHS, ensure_determinism=ENSURE_DETERMINISM))
```

Com todos os ajustes realizados, clicar em “Save & Train”, e após alguns minutos, o modelo deverá estar treinado.

Após o treinamento, do lado direito é possível verificar as métricas do modelo.



Conforme já observado, esse é um processo iterativo, se as métricas de treinamento não atendem o caso de uso, deve-se ajustar os parâmetros do modelo e retreiná-lo. Caso continuem não atendendo, pode ser necessário mais dados ou ainda reavaliar a extração de recursos.

Teste de modelo

As métricas apresentadas anteriormente são para os dados de treinamento, agora vamos reavaliar o modelo treinado com dados que ele ainda não conhecia (dados de testes).

Na página “Model testing” do projeto, clique em “Classify All”, após alguns instantes, o conjunto de teste deverá ser classificado usando o modelo treinado.

E novamente as métricas podem ser avalidas.



ACCURACY

77.78%

Metrics for Classifier



METRIC	VALUE
Area under ROC Curve ⓘ	1.00
Weighted average Precision ⓘ	0.95
Weighted average Recall ⓘ	0.94
Weighted average F1 score ⓘ	0.94

Confusion matrix

	CIRCLE	LEFT_RIGHT	RESTING	UP_DOWN	UNCERTAIN
CIRCLE	44.4%	0%	0%	0%	55.6%
LEFT_RIGHT	0%	88.9%	0%	0%	11.1%
RESTING	0%	0%	100%	0%	0%
UP_DOWN	0%	0%	0%	77.8%	22.2%
F1 SCORE	0.62	0.94	1.00	0.88	

Caso as métricas atendam os objetivos de negócio, o próximo passo é implementar o modelo no microcontrolador.

Implementação

Na página “Deployment” do Edge Impulse, existem opções de seleção de dispositivos e quantização do modelo.

Como já estávamos configurados para o target: RP2040, a tela já veio com essa opção, conforme figura abaixo:

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.](#)



Search deployment options

DEFAULT DEPLOYMENT



Raspberry Pi RP2040

A binary containing both the Edge Impulse data acquisition client and your full impulse.

Contudo, nesta opção, o download da EI já é um arquivo compilado “ei_rp2040_firmware.uf2”, bastando carregá-lo na BitDogLab.

Contudo, nosso projeto possui muitos outros requisitos de periféricos e comunicação, assim, nossa opção foi alterar para “C++ library”, conforme figura abaixo:

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.](#)



C++ library X



SELECTED DEPLOYMENT

C++ library

A portable C++ library with no external dependencies, which can be compiled with any modern C++ compiler.

Com a opção acima, o download da EI é uma pasta compactada com as seguintes pastas/arquivos:



edge-impulse-sdk



model-parameters



tflite-model



CMakeLists.txt



README.txt

E com as pastas/arquivos acima, incorporamos o modelo de ML treinado no nosso projeto, seguindo o tutorial: Run C++ library on Raspberry Pi Pico (RP2040) [https://docs.edgeimpuse.com/hardware/deployments/run-cpp-rpi-rp2040#run-c-library-on-raspberry-pi-pico-rp2040](https://docs.edgeimpulse.com/hardware/deployments/run-cpp-rpi-rp2040#run-c-library-on-raspberry-pi-pico-rp2040) .

Antes ainda de fazer download do Build, conforme figura abaixo, pode-se comparar o tempo de inferência, utilização de RAM e acurácia em função da quantização adotada.

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

Quantized (int8)	SPECTRAL FEAT	CLASSIFIER	TOTAL
Selected ✓	LATENCY 17 ms.	2 ms.	19 ms.
	RAM 1.0K	1.4K	1.4K
	FLASH -	15.5K	-
	ACCURACY		77.78%

Unoptimized (float32)	SPECTRAL FEAT	CLASSIFIER	TOTAL
Select	LATENCY 17 ms.	10 ms.	27 ms.
	RAM 1.0K	1.6K	1.6K
	FLASH -	16.0K	-
	ACCURACY		77.78%

Estimate for Raspberry Pi RP2040 (Cortex-M0+ 133MHz) - [Change target](#)

Outra característica interessante é a possibilidade de baixar diretamente o modelo para um celular, e fazer a inferência diretamente nele, bastando escanear o código abaixo:

Run this model

Scan QR code or launch in browser to test your prototype



Launch in browser

Com as seleções realizadas, clicar em “Build” que o download do modelo deve acontecer automaticamente.

Inferência

Inferência é o processo em que um modelo treinado usa o conhecimento que adquiriu para analisar novos dados e gerar resultados, como previsões ou decisões, sem ter visto esses dados antes.

Com base no tutorial “Run C++ library on Raspberry Pi Pico (RP2040)” e com as devidas adequações a estrutura do nosso projeto, o modelo de ML foi incorporado na BitDogLab, permitindo a inferência na borda (tunyML), conforme pode ver verificado nas capturas da saída serial:

<pre>39 Coletou 42 amostras, exe 40 41 Timing: DSP 41 ms, infer 42 43 Predictions: 44 45 circle: 0.00000 46 47 left_right: 0.00000 48 49 resting: 0.99609 50 51 up_down: 0.00000 52 53 resting 54</pre>	<pre>837 Coletou 42 amostras, e 838 839 Timing: DSP 37 ms, inf 840 841 Predictions: 842 843 circle: 0.77344 844 845 left_right: 0.06641 846 847 resting: 0.15234 848 849 up_down: 0.00781 850 851 circle</pre>
<pre>889 Coletou 42 amostras, e 890 891 Timing: DSP 37 ms, infer 892 893 Predictions: 894 895 circle: 0.02344 896 897 left_right: 0.97656 898 899 resting: 0.00000 900 901 up_down: 0.00000 902 903 left_right 904</pre>	<pre>1105 Coletou 42 amostras, e 1106 1107 Timing: DSP 37 ms, inf 1108 1109 Predictions: 1110 1111 circle: 0.30859 1112 1113 left_right: 0.69141 1114 1115 resting: 0.00000 1116 1117 up_down: 0.00000 1118 1119 Resultado incerto</pre>

Conforme capturas acima, exceto para a classe up_down, todas as outras 3 foram identificadas.

Foi implementado ainda, um limiar de confiança de 70%, isto é, se nenhuma classe conseguir ao menos esse percentual de confiança, a saída será “Resultado incerto”.