Aprendizado de Máquina em Microcontroladores

TinyML: Exploração, Otimização e Portabilidade

Autor: Maurício Taffarel

Departamento de Engenharia Elétrica e de Computação

contact@taffarel.tech

Sumário

Introdução

Contextualização

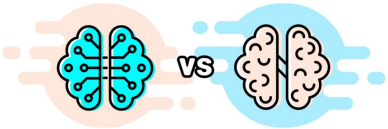

Fig. 1: Inteligência artificial e humana[1]

- Inovação Tecnológica
- Eficiência Operacional
- Solução de Problemas Complexos
- Tomada de decisões
- TinyML
- Internet das Coisas

[1] 1, *Developers vs AI: What Happens in the Future? — chanidumadalagama.medium.com*.

Desafios dos Sistemas de Machine Learning

O TinyML apresenta-se como uma solução para enfrentar vários desafios encontrados na implementação de sistemas de Machine Learning em aplicações industriais[2]:

- **Consumo elevado de energia**: O TinyML permite a execução de modelos de Machine Learning em microcontroladores com baixo consumo de energia.
- **Problemas com privacidade de dados**: O TinyML reduz a necessidade de enviar informações sensíveis para serviços na nuvem.
- **Dependência de conexão**: Com o TinyML, os modelos de Machine Learning podem ser implantados diretamente nos dispositivos.
- **Latência**: A execução local dos modelos de ML no TinyML reduz a latência, uma vez que as inferências são feitas no próprio dispositivo.

[2] 2, "A review on TinyML: State-of-the-art and prospects", 2022.

Objetivos

Objetivos gerais

Investigar e compreender os fundamentos e possibilidades da Tiny Machine Learning (TinyML).

Objetivos específicos

1. Explorar as técnicas utilizadas
2. Identificar os desafios e restrições
3. Investigar as ferramentas
4. Avaliar os modelos implementados em microcontroladores

Aprendizado de Máquina

Machine Learning como subcampo da Inteligência artificial

- Inovação Tecnológica
- Eficiência Operacional
- Solução de Problemas Complexos

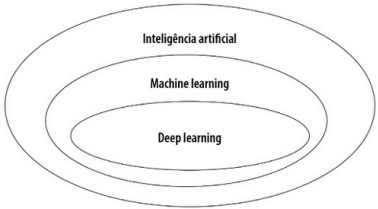

Fig. 2: Aprendizado de Máquina como um subcampo da Inteligência Artificial[4]

[4] 3, *Introdução à Inteligência Artificial: Uma abordagem não técnica*, 2019.

Tipos de aprendizado de máquina

Algorítimos de aprendizado de máquina

- **Aprendizado supervisionado**: o algoritmo aprende a partir de um conjunto de exemplos rotulados, onde os dados de entrada estão associados a um rótulo de saída correspondente.
- **Aprendizado não supervisionado**: o algoritmo aprende a partir de um conjunto de dados não rotulados, identificando padrões e estruturas sem a utilização de rótulos ou informações externas.
- **Aprendizado por reforço**: o algoritmo aprende a tomar decisões e realizar ações com base em um sistema de recompensas e punições, maximizando assim a recompensa total ao longo do tempo.

O neurônio artificial

Perceptron

Inspirado no que se sabia até o momento sobre o cérebro humano e o modelo do neurônio biológico, foram realizadas diversas tentativas de realizá-lo matematicamente. O primeiro modelo que se tem registros é o dos pesquisadores **McCulloch e Pitts** no ano de 1943[5].

Mais tarde, inspirado nas pesquisas destes pesquisadores, em 1958, o cientista **Frank Rosenblatt** desenvolveu o que o mesmo denominou como Perceptron[6]

[5] 4, "Deep learning in neural networks: An overview", 2015.

[6] 5, "The perceptron: a probabilistic model for information storage and organization in the brain", 1958.

Estrutura do neurônio artificial

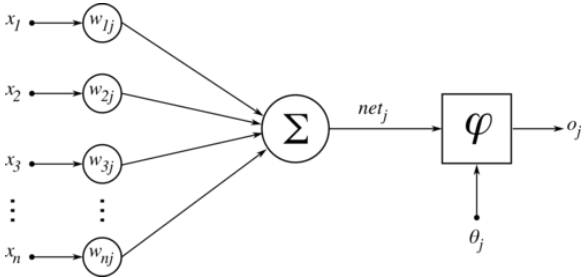

Fig. 3: Estrutura do neurônio artificial[7]

[7] Disponível em: https://commons.wikimedia.org/wiki/File:ArtificialNeuronModel.png

Equações do Perceptron

Assim, utilizando como base a Figura 3, pode-se estabelecer equações que modelam um neurônio artificial:

$$\begin{aligned} net_j(t) &= \sum_{i=1}^{n} w_{ij}(t) \cdot x_i(t) \\ o_j(t) &= \varphi\left(net_j(t)\ ,\ \theta(t)\right) \end{aligned} \tag{1}$$

Equações do Perceptron

Simplificando as equações

Pode-se simplificar a equação (1) e representa-las apenas como um somatório:

$$net'_j(t) := \theta(t) + \sum_{i=1}^{n} w_{ij}(t) \cdot x_i(t) = \omega_{0j} \cdot 1 + \sum_{i=1}^{n} w_{ij}(t) \cdot x_i(t) \quad (2)$$

Equações do Perceptron

Simplificando as equações

Pode-se simplificar a equação (1) e representa-las apenas como um somatório:

$$net'_j(t) := \theta(t) + \sum_{i=1}^{n} w_{ij}(t) \cdot x_i(t) = \omega_{0j} \cdot 1 + \sum_{i=1}^{n} w_{ij}(t) \cdot x_i(t) \quad (2)$$

$$net'_j(t) := \sum_{i=0}^{n} w_{ij}(t) \cdot x_i(t) \quad (3)$$

Equações do Perceptron

Saída do Perceptron

A saída é determinada por uma função de ativação $\phi(\cdot)$, tomando como exemplo a degrau bipolar:

$$o_j(t) = \varphi\left(net_j(t)\right) = \begin{cases} 1, & \text{se } net_j(t) \geq 0 \\ -1, & \text{se } net_j(t) < 0 \end{cases} \tag{4}$$

Equações do Perceptron

Limite da condição

O limite da condição ou *threshold* é definido quando $net'_j(t) = 0$.

$$net'_j(t) = \theta(t) + \sum_{i=1}^{n} w_{ij}(t) \cdot x_i(t) = 0 \quad (5)$$

Para $n = 2$, obtém-se:

$$x_1(t)\omega_{1j}(t) + x_2(t)\omega_{2j}(t) + \theta(t) = 0 \quad (6)$$

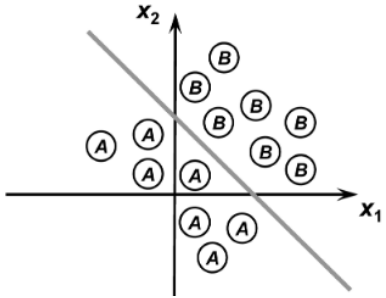

Fig. 4: Classificação com 2 entradas.[8]

[8] 6, *Introdução à Inteligência Artificial: Uma abordagem não técnica - Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas. Fundamentos Teóricos e Aspectos Práticos*, 2010

Processo de aprendizagem de uma rede neural

Equações para implementação computacional

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \eta \cdot (\mathbf{v}^k - \mathbf{o}^k) \cdot \mathbf{x}^k \tag{7}$$

$$\boldsymbol{\omega} = \begin{bmatrix} \theta \\ \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \end{bmatrix}, \ \mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{8}$$

Processo de aprendizagem de uma rede neural

Redes Neurais Artificiais

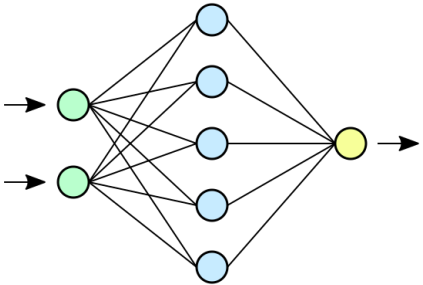

Fig. 5: Exemplo de uma rede neural composta por vários neurônios artificais[9]

[9] Disponível em: https://commons.wikimedia.org/wiki/File:Neural_network.svg

Ética para Inteligência Artificial

É preciso tomar cuidado para que os algoritmos não perpetuem preconceitos, discriminação ou desigualdades existentes na sociedade[a]

[a] 7, "Can Machine Learning be Moral?", 2022.

Devido às possibilidades de uso prejudicial das tecnologias baseadas em inteligência artificial, as empresas também adotaram a iniciativa de utilizar licenças responsáveis para[a]. Licenças Responsáveis de IA possibilitam os desenvolvedores a restringir o uso de sua tecnologia de IA, a fim de prevenir aplicações irresponsáveis e prejudiciais[b]

[a] 8, *Responsible AI License*, 2023.

[b] 9, *Behavioral use licensing for responsible AI*, 2022.

Aprendizado de Máquina em Sistema embarcado

Desafios para integração de modelos em sistemas embarcados

Sistema embarcado

Um sistema embarcado é qualquer sistema que esteja embutido ou incorporado dentro de outros sistemas de forma a desempenhar uma determinada função específica do sistema maior.

Quando se trata de integração de aprendizado de máquina nesses sistemas, o tamanho e a potência limitada dos dispositivos embarcados apresentam desafios significativos[10].

[10] 10, “An Overview of Machine Learning within Embedded and Mobile Devices–Optimizations and Applications”, 2021.

TinyML

Definição

TinyML

TinyML pode ser definido como o conjunto de técnicas que permite a implementação de algoritmos em dispositivos que não tem tantos recursos computacionais e capacidade de memória.[a] Geralmente costuma-se utilizar 1mW ou menos para dizer-se que um dispositivo está dentro do escopo do TinyML[b c].

[a] 2, "A review on TinyML: State-of-the-art and prospects", 2022.

[b] 11, *Tiny ML*, 2020.

[c] 12, *CMSIS-NN & Optimizations for Edge AI*, 2021.

TinyML
Aplicações

O TinyML pode ser usado para diversas aplicações[11]:

- Reconhecimento de imagem
- Reconhecimento de voz
- Detecção de expressões faciais
- Controle de tráfico
- Detecção de padrões

[11] 13, "A review on TinyML: State-of-the-art and prospects", 2022.

Otimização de modelos

Compactação de modelos

Compactando os modelos

Uma forma de compactar o modelo é quantizar os pesos, isto permite reduzir o tamanho do modelo sem degradar tanto a sua precisão.

Estratégias de quantização

Existem duas formas de quantização: quantização pós-treinamento e treinamento consciente de quantização. A primeira estratégia de quantização é feita após o treinamento e a outra durante o treinamento.[14, *Treinamento consciente de quantização — TensorFlow Model Optimization — web.archive.org*, 2023]

Otimização de modelos

Compactação de modelos

- Quantização de faixa dinâmica
- Quantização com amostra representativa
- Quantização inteira completa

Ferramentas utilizadas

Com objetivo de se investigar as técnicas de implementação redes neurais em sistemas embarcados foram utilizadas as seguintes ferramentas:

- **TensorFlow Lite**: Biblioteca móvel que permite a execução de modelos TensorFlow em dispositivos móveis e incorporados.[12].
- **EdgeImpulse**: Plataforma de desenvolvimento para soluções de machine learning em sistemas embarcados, ela permite o treinamento de estruturas e desenvolvimento diretamente do navegador com pouco uso de codificação[13].

[12] 15, *TensorFlow Lite — ML para dispositivos móveis e do Edge — web.archive.org*, 2023.

[13] 16, *For beginners — web.archive.org*, 2023.

Ferramentas utilizadas

TensorFlow Lite

Modelo a ser desenvolvido:

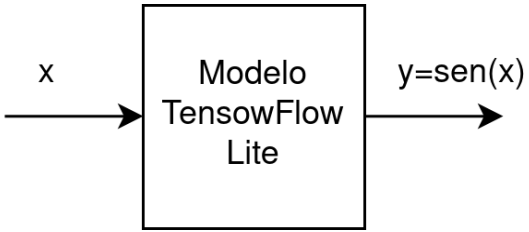

Fig. 6: Modelo a ser desenvolvido no TensorFlow Lite[14]

[14] Autoria Própria

Ferramentas utilizadas

TensorFlow Lite - Dados

Foram utilizados para compor a base de dados, um conjunto de 1000 amostras gerados através de uma distribuição uniforme entre $x = [0, 2\pi]$ e com um erro gaussiano com média 0 variância $0,1$, divididos da seguinte maneira:

1. 60% utilizados para treinamento;
2. 20% utilizados para validação;
3. 20% utilizados para teste.

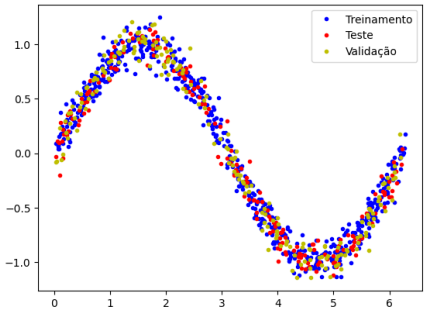

Fig. 7: Dados para treinamento e validação[15]

[15] Autoria Própria

Ferramentas utilizadas

TensorFlow Lite - Modelo Proposto

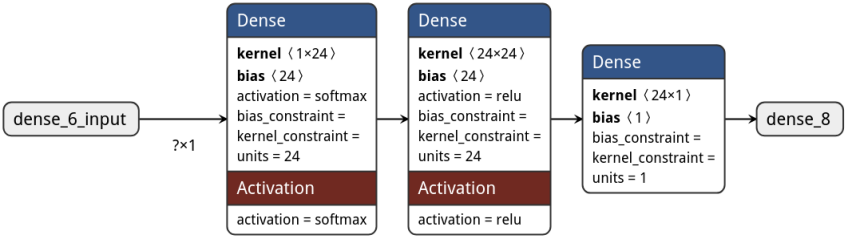

Fig. 8: Estrutura do modelo proposto[16]

[16] Autoria Própria

Ferramentas utilizadas

TensorFlow Lite - Conversão do modelo

```
# Convert Keras model to a tflite model
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
```

```
from tensorflow.lite.python.util import convert_bytes_to_c_source

source_text, header_text =
    convert_bytes_to_c_source(tflite_model, "sine_model")
with open('sine_model.h', 'w') as file:
    file.write(header_text)
with open('sine_model.cc', 'w') as file:
    file.write(source_text)
```

Ferramentas utilizadas

TensorFlow Lite - Otimização

Foi feito a quantização dinâmica e representativa.

```
1 import tensorflow as tf
2 converter = tf.lite.TFLiteConverter.from_saved_model(
      saved_model_dir)
3 converter.optimizations = [tf.lite.Optimize.DEFAULT]
4 tflite_quant_model = converter.convert()
```

Código 1: Quantização de faixa dinâmica

Ferramentas utilizadas

TensorFlow Lite - Firmware

```
#include <EloquentTinyML.h> #include "sine_model.cc"
Eloquent::TinyML::TfLite<NUMBER_OF_INPUTS, NUMBER_OF_OUTPUTS,
    TENSOR_ARENA_SIZE> ml;

void main() {
  ml.begin(sine_model);
  for(int i=0; i<=NUMBER_OF_POINTS_INFERING; i++) {
    float x = 2*PI*i/NUMBER_OF_POINTS_INFERING;
    float input[1] = { x };
    float predicted = ml.predict(input);
    printf("%.8f,%.8f", x, predicted);
  }
}
```

Código 2: Firmware para inferências no microcontrolador ESP32s

Ferramentas utilizadas

Edge Impulse

Modelo a ser desenvolvido:

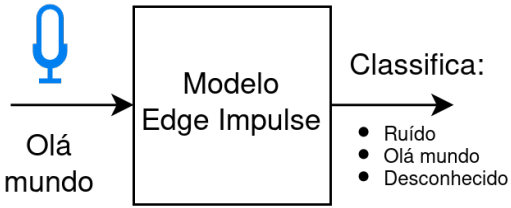

Fig. 9: Modelo a ser desenvolvido no Edge Impulse

[16] Autoria Própria

Ferramentas utilizadas

Edge Impulse - Dados

Foi levantado um conjunto de 50*min* de áudio para classificação de 3 tipos:

1. Rótulo: **ola-mundo** (1000 amostras de 1*s* de ”Olá mundo”)
2. Rótulo: **desconhecido** (1000 amostras de 1*s* de palavras desconhecidas)
3. Rótulo: **ruido** (1000 amostras de 1*s* de ruídos)

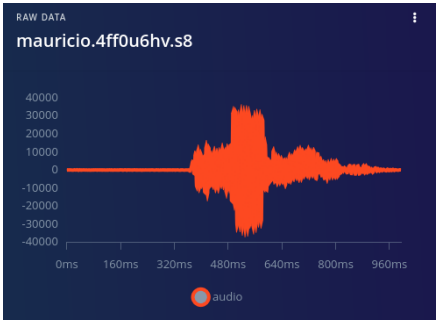

Fig. 10: Exemplo de áudio obtido[17]

[17] Autoria Própria

Ferramentas utilizadas

Edge Impulse - Extração de características

A extração de características utilizado para este modelo foi o *Mel-filterbank energy features* (MFE). Este sistema avalia o espectro do sinal e realiza uma conversão das frequências para a escala Mel.

Os seguintes parâmetros para este bloco foram utilizados:

1. Largura do frame: $0,025s$
2. Passo do frame: $0,01s$
3. Número de filtros do banco: 41
4. Número de pontos da FFT: 512
5. Normalização de ruído: $-100dB$

[17] Autoria Própria

Ferramentas utilizadas

Edge Impulse - Modelo Proposto

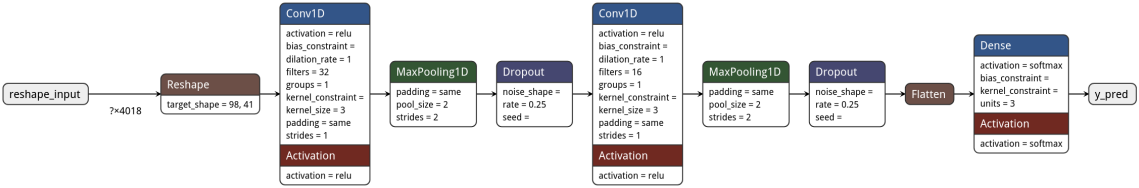

Fig. 11: Arquitetura do modelo para o Edge Impulse[18]

[18] Autoria Própria

Ferramentas utilizadas

Edge Impulse - Conversão do modelo

Conversão e Otimização

A conversão do modelo e otimização são feitos pós treinamento para a plataforma e é possível obter o modelo quantizado para 8 bits inteiro e para 32 bits com ponto flutuante.

Ferramentas utilizadas

Edge Impulse - Gravando no microcontrolador

Após o treinamento, foi utilizado os binários gerados pela plataforma para gravar o firwmare.

```
$./flash_linux.sh
Flashing board...

Done in 0.001 seconds
Write 367080 bytes to flash (90 pages)
[==============================] 100% (90/90 pages)
Done in 14.077 seconds
New upload port: /dev/ttyACM0 (serial)
```

Código 3: Gravando o firmware no Arduino Nano 33 BLE

TensorFlow Lite

Resultados do Treinamento

Os resultados do treinamento utilizando o TensorFlow:

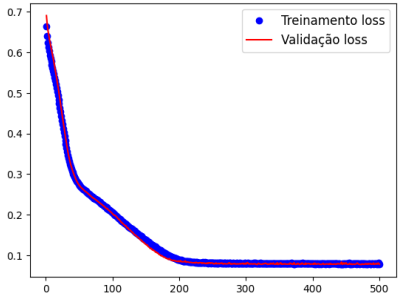

Fig. 12: Treinamento e Validação

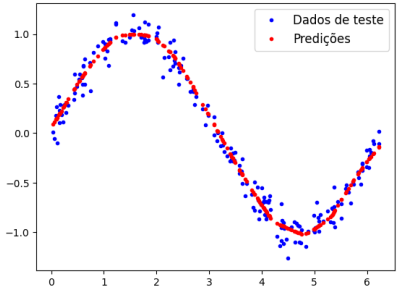

Fig. 13: Novas predições

TensorFlow Lite

Resultados da Inferência no Microcontrolador ESP32s

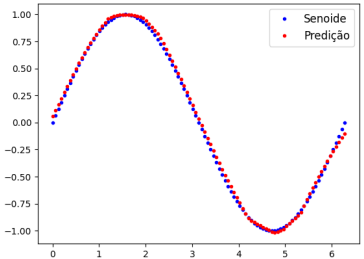

Fig. 14: Quantização de faixa dinâmica

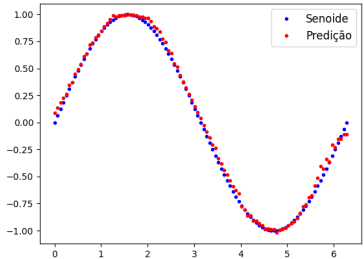

Fig. 15: Quantização com amostras representativas

Comparação das inferências

Optimização com faixa dinâmica e com amostras representativas

Table 1: Desempenho para diferentes estratégias de otimização no TensorFlow Lite

Otimização	Tamanho (kB)	Desvio padrão	Erro máximo	Latência média (μs)
DEFAULT	4764	0,0267	0,1014	136
REPR_100	3624	0,0320	0,1324	791

Edge Impulse

Extração de características

Utilizando o algoritmo **t-SNE**, pode-se ver na figura 16 diferentes regiões que separam os dados.

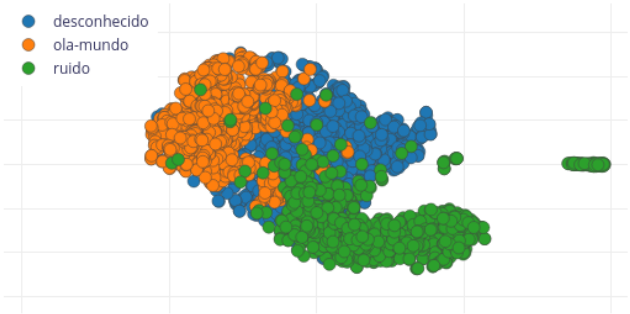

Fig. 16: Exploração dos dados - Fonte: Edge Impulse

Edge Impulse

Resultados do Treinamento

	DESCONHECIDO	OLA-MUNDO	RUIDO
DESCONHECIDO	95.1%	2.4%	2.4%
OLA-MUNDO	0.6%	99.4%	0%
RUIDO	1.9%	0%	98.1%
F1 SCORE	0.96	0.98	0.98

Fig. 17: Matriz de confusão - Fonte: Edge Impulse

Edge Impulse

Exploração dos resultados com os dados de treinamento

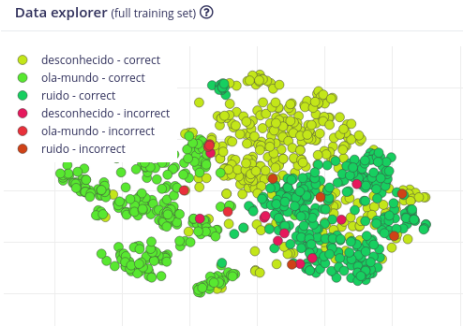

Fig. 18: Exploração de dados de treinamento - Fonte: Edge Impulse

Edge Impulse

Estimativa de desempenho

Fig. 19: Estimativa de Performance no Arduino Nano 33 BLE Sense - Fonte: Edge Impulse

Edge Impulse

Resultados com dados de teste com *data augmentation*

Na figura 20 é apresentado os resultados de como o modelo responde a detecção de eventos no tempo utilizando como sinal de entrada, dados gerados a partir dos dados de teste com dados de ruído como ruído de fundo.

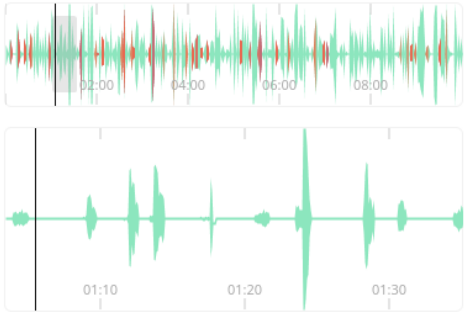

Fig. 20: Simulando ambiente real - Fonte: Edge Impulse

Edge Impulse

Visão geral do desempenho do modelo

LABEL	FAR	FRR	TRUE POSITIVES	FALSE POSITIVES	TRUE NEGATIVES	FALSE NEGATIVES
ola-mundo	1.1%	62.3%	23	2	179	38
desconhecido	10.6%	74.6%	15	20	169	44

Fig. 21: Visão geral do desempenho - Fonte: Edge Impulse

Edge Impulse

Inferências no Microcontrolador Arduino Nano 33 BLE Sense Rev 2

```
Predictions (DSP: 74 ms, Class: 39 ms, Anomaly: 0 ms):
    desconhecido: 0.15234
    ola-mundo: 0.01562
    ruido: 0.82812        # <- Prediction: ruido
Predictions (DSP: 74 ms, Class: 39 ms, Anomaly: 0 ms):
    desconhecido: 0.00391
    ola-mundo: 0.99609    # <- Prediction: ola-mundo
    ruido: 0.00000
Predictions (DSP: 74 ms, Class: 39 ms, Anomaly: 0 ms):
    desconhecido: 0.99609 # <- Prediction: desconhecido
    ola-mundo: 0.00000
    ruido: 0.00391
```

Código 4: Inferências no microcontrolador Arduino Nano 33 BLE Sense Rev 2

Conclusão

Desta forma, pode-se concluir que o trabalho atingiu os seus objetivos.

- Explorar conceitos relacionados ao TinyML
- Realizar conversões otimizações de modelos
- Explorar técnicas associados ao TinyML
- Implementar modelos com exemplos de ponta a ponta
- Avaliar e comparar os modelos

Desta forma, espera-se que este trabalho sirva como fundamento para investigações futuras, onde os conceitos e ferramentas apresentados possam ser utilizados como alicerce para a pesquisa de aspectos mais específicos no campo de TinyML.

Agradecimentos