

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI

JOSÉ ANDERSON DOS REIS

SAMUEL LIMA BRAZ

TONY ALBERT LIMA

SISTEMA DE MONITORAMENTO DE SAÚDE DE COLMEIAS BASEADO EM
TINYML

Itajubá

2024

JOSÉ ANDERSON DOS REIS
SAMUEL LIMA BRAZ - 2022008027
TONY ALBERT LIMA - 2022014810

SISTEMA DE MONITORAMENTO DE SAÚDE DE COLMEIAS BASEADO EM TINYML

Relatório Final da disciplina IESTI01 -
TINYML - Aprendizado de Máquina
Aplicado para Dispositivos IOT
Embarcados, lecionada na
Universidade Federal de Itajubá com o
objetivo de expor a idealização do projeto
final.

Orientador: Marcelo José Rovai
Coorientador: José Alberto Ferreira Filho

Itajubá
2024

RESUMO

Este projeto visa desenvolver um sistema eficiente e de baixo custo para monitorar a saúde e a atividade de colmeias usando o tinyML. Como prova de conceito para realizar o desenvolvimento em local controlado o projeto foi desenvolvido reconhecendo capacitores eletrolíticos. O sistema irá coletar e analisar imagens para encontrar os capacitores e contar sua quantidade, assemelhando-se à obtenção de informação sobre as abelhas que possibilita uma potencial extração de informações adicionais para avaliar a saúde das colmeias. Plataformas de computação de ponta como EdgeImpulse ou Roboflow foram usadas para criação, treinamento e implantação de modelos em um dispositivo Xiao Esp32s3.

Palavras-chave: TinyML, Aprendizado Profundo, Visão Computacional, EdgeImpulse, FOMO.

SUMÁRIO

1 INTRODUÇÃO.....	4
2 OBJETIVOS.....	5
3 DESCRIÇÃO DO PROJETO.....	6
4 DESENVOLVIMENTO.....	7
4.1 DIAGRAMA DE BLOCOS.....	7
4.2 HARDWARE.....	7
4.3 COLETA DE DADOS.....	8
4.3.1 Aquisição de Dados.....	8
4.3.2 Pré-processamento.....	9
4.3.2.1 Rotulagem.....	9
4.3.2.2 Normalização.....	10
4.3.2.3 Aumento de Dados.....	10
4.4 MODELO.....	11
4.4.1 Arquitetura da rede.....	11
4.4.2 Treinamento do modelo.....	14
4.4.2.1 Ajuste de Hiper parâmetros.....	14
4.4.2.2 Avaliação do modelo.....	15
4.4.3 Implantação.....	16
4.4.3.1 Otimização.....	16
4.4.3.2 Implementação.....	16
4.5 INFERÊNCIA.....	17
5 CONCLUSÃO.....	17
6 REFERÊNCIAS.....	19

1 INTRODUÇÃO

As abelhas desempenham um papel crucial nos ecossistemas globais, polinizando plantas que fornecem uma parte significativa do nosso abastecimento alimentar. No entanto, as populações de abelhas enfrentam inúmeras ameaças, incluindo perda de habitat, pesticidas e doenças. A detecção precoce destas ameaças é essencial para a intervenção dos apicultores e a sobrevivência das colônias. Os métodos tradicionais de monitorização das colmeias baseiam-se frequentemente em inspeções físicas, que podem ser demoradas, perturbadoras para a colmeia e potencialmente deixar passar alterações sutis na saúde. Nosso projeto propõe uma solução automatizada de baixo custo e não intrusiva usando TinyML.

Tendo em vista a dificuldade de desenvolvimento e implementação deste trabalho para a disciplina de TinyML, nós optamos por desenvolver um sistema semelhante localiza e conta capacitores em uma mesa, esse sistema possui os mesmos desafios de identificar objetos pequenos utilizando um dispositivo de baixo consumo de energia, a Xiao Esp32s3. Portanto servirá como demonstração do processo desenvolvimento e capacidade do sistema.

2 OBJETIVOS

Este projeto visa desenvolver um sistema automatizado e de baixo custo para monitoramento da saúde das colmeias usando tinyML. O objetivo principal é criar um modelo que possa contar com precisão as abelhas dentro da colmeia com base na análise de imagens. Isso foi realizado através da contagem de componentes eletrônicos do tipo capacitor eletrolítico como prova de conceito em ambiente controlado no lugar das abelhas.

Figura 1 — Imagem com labels identificados



Fonte: Os autores (2024).

3 DESCRIÇÃO DO PROJETO

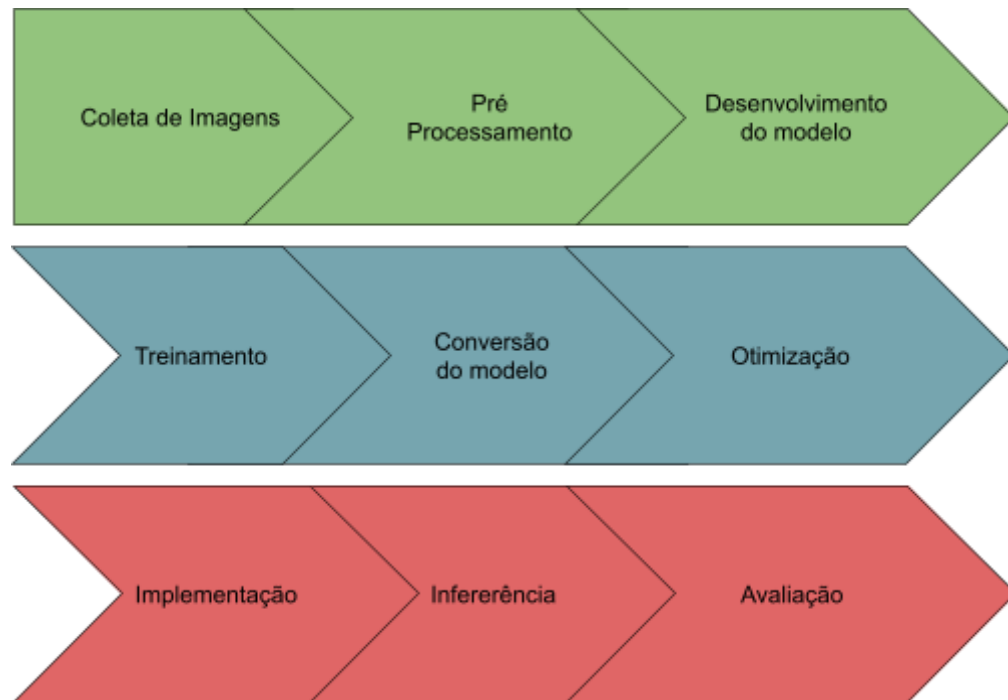
O sistema proposto consiste em uma câmera de baixa potência colocada em cima de uma mesa, capturando imagens em intervalos regulares dos capacitores espalhados nesta. Esses dados serão processados dispositivo Xiao Esp32s3 usando um modelo FOMO. Este modelo, treinado em imagens dos capacitores rotulados, é capaz de contar o número de componentes presentes em cada imagem. É composto por três componentes principais:

1. Aquisição de dados: Coletamos várias imagens de capacitores em diferentes posições da mesa, com e sem mãos aparecendo na imagem e com diferentes quantidades.
2. Desenvolvimento do modelo: As imagens capturadas foram carregadas na plataforma Roboflow para a rotulagem. Utilizamos caixas delimitadoras para identificar os capacitores. Os dados rotulados foram então usados para treinar um modelo TinyML na plataforma Edge Impulse.
3. Implantação: O modelo treinado foi implantado no dispositivo e uma aplicação criada para possibilitar a visualização dos resultados que mostra a imagem capturada sobreposta com as labels identificadas.

4 DESENVOLVIMENTO

4.1 DIAGRAMA DE BLOCOS

Figura 2 — Diagrama dos processos de desenvolvimento



Fonte: Os autores (2024).

4.2 HARDWARE

Câmera: OV2640 é módulo de câmera inserido na placa Xiao Esp32. É uma câmera colorida de 1600*1200 pixels.

Microcontrolador: O microcontrolador ESP32-S3 é um dual-core XTensa LX7 MCU, capaz de rodar até 240 MHz.

Figura 3 — Xiao



Fonte: Os autores (2024).

4.3 COLETA DE DADOS

O dataset para treinamento do modelo consistem de imagens da mesa com a identificação *bounding box* dos capacitores presentes na imagem.

4.3.1 Aquisição de Dados

A coleta de dados foi realizada instalando uma camera em acima de uma mesa para capturar as imagens de cacapitores foi realizada uma marcação na mesa que indica os limites de captura da câmera. Na Figura 4 é possível ver a disposição utilizada para capturar as imagens.

Figura 4 — Aquisição de dados



Fonte: Os autores (2024).

A aquisição foi realizada com auxílio de um script em C++ para tirar fotos e salvar repetidamente.

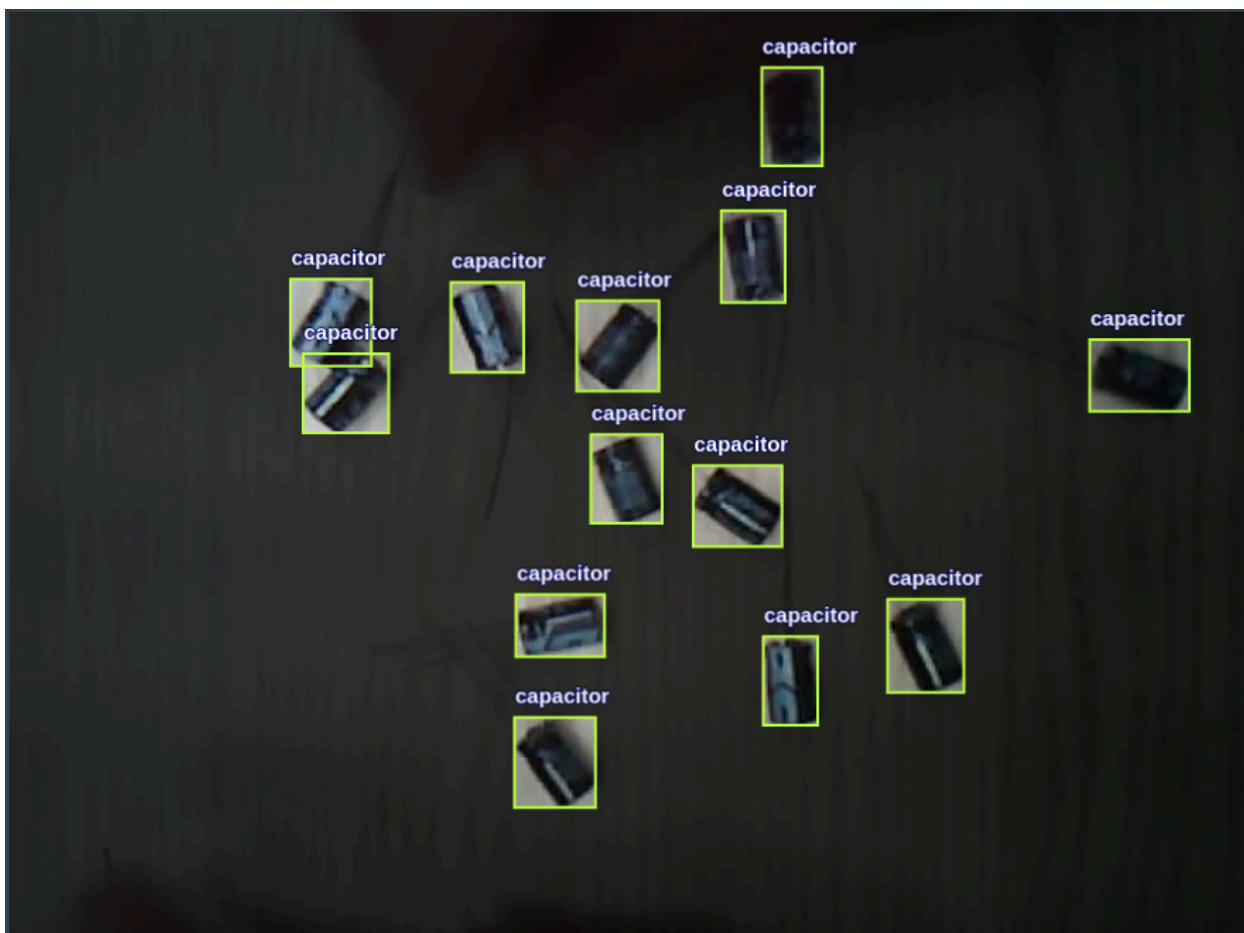
4.3.2 Pré-processamento

Os dados da imagem passaram por pré-processamento para prepará-los para o treinamento de modelos. Isso envolve a rotulagem, normalização e aumento de dados.

4.3.2.1 Rotulagem

Um aspecto crucial do pré-processamento desses dados é a rotulagem das imagens com caixas delimitadoras em torno dos capacitores. Isso foi realizado através do RoboFlow, onde a rotulagem manual foi realizada, posicionando as caixas, em conjunto com a funcionalidade de Auto Label fornecida pela plataforma.

Figura 5 — Rotulagem na plataforma Roboflow



Fonte: Os autores (2024).

4.3.2.2 Normalização

Para garantir a consistência e melhorar o desempenho do nosso modelo, padronizamos nossos dados. Isso envolve redimensionar todas as imagens para um tamanho uniforme e normalizar os valores de pixels. Através do Roboflow foi realizado um aplicação de auto-orientação e o redimensionamento para 640x640.

4.3.2.3 Aumento de Dados

Para aumentar a robustez do nosso modelo e evitar o sobreajuste, realizamos o aumento de dados. Isso envolve a criação de versões modificadas de nossas imagens através de técnicas como rotação, escala, inversão e corte. Isso aumenta efetivamente o tamanho do nosso conjunto de dados e fornecerá ao nosso modelo

uma variedade maior de dados para aprender. Também através do Roboflow conseguimos realizar isso utilizando as seguintes características:

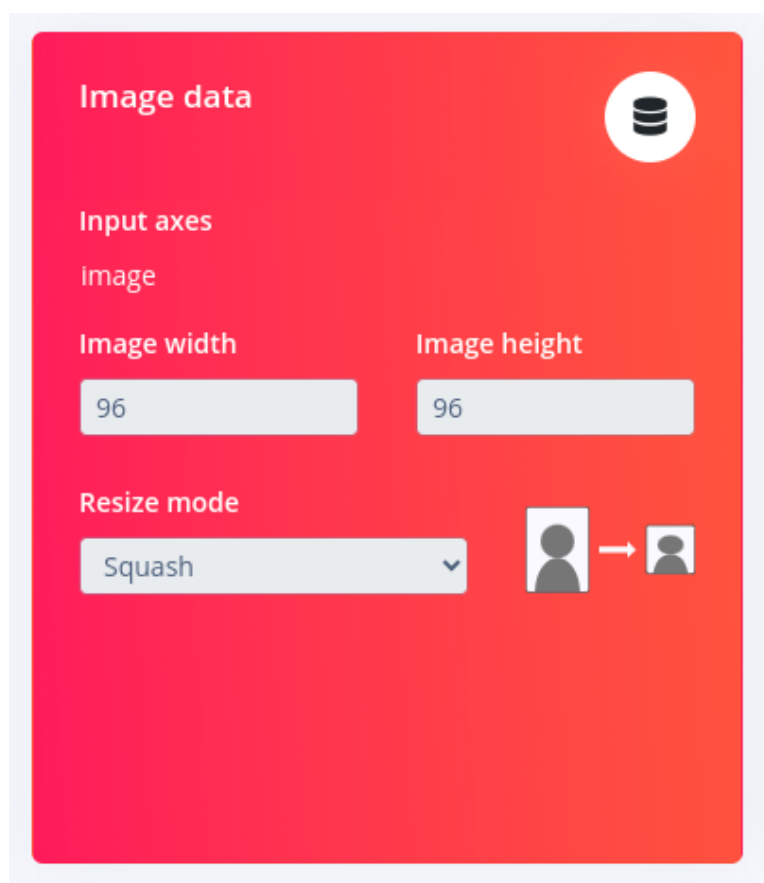
- Espelhamento vertical e horizontal;
- Variação de brilho da imagem entre -25% e +25%;
- Variação de exposição da imagem entre -10% e +10%;
- Aplicação de *blur* até 2.5px.

4.4 MODELO

4.4.1 Arquitetura da rede

A arquitetura da rede neural para este projeto foi projetada para utilizar a detecção de objetos com FOMO e consiste da camada de entrada que redimensiona a imagem para um tamanho de 96x96px.

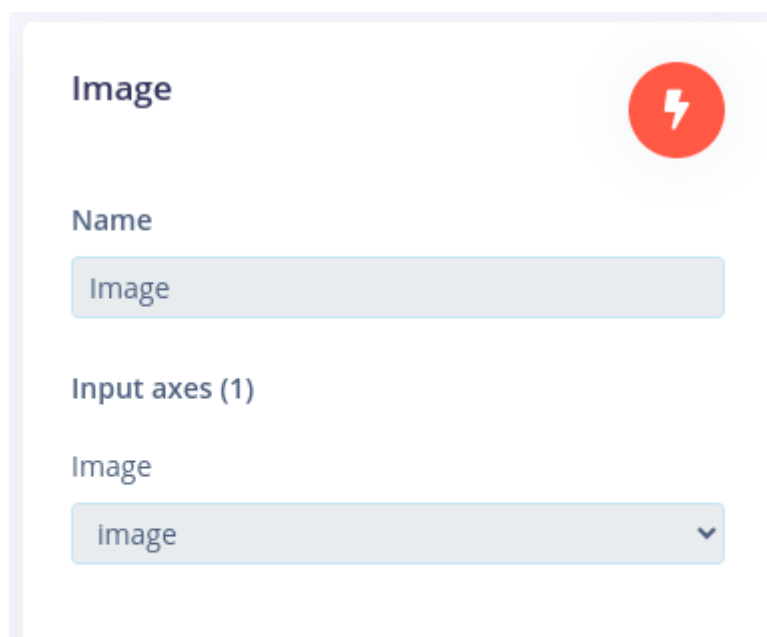
Figura 6 — Entrada da rede



Fonte: Os autores (2024).

Em seguida temos uma camada de filtros e convolução é utilizada.

Figura 7 — filtros



The screenshot shows a configuration panel for an 'Image' filter. At the top, there's a red lightning bolt icon. Below it, the 'Name' field is a text input containing 'Image'. Under the 'Input axes (1)' section, there's a dropdown menu labeled 'Image' with 'image' selected and a downward arrow.

Fonte: Os autores (2024).

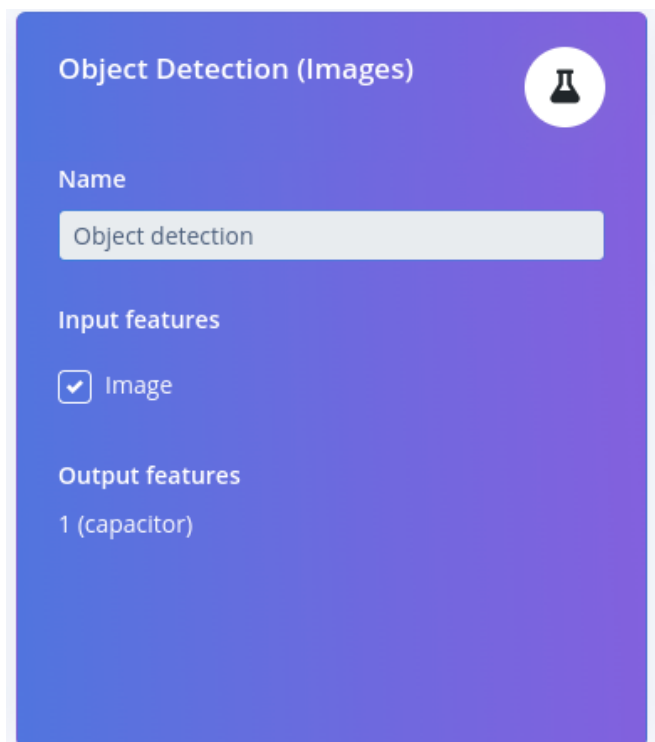
E por fim o modelo de detecção de objeto FOMO entregue pela plataforma Edge Impulse que utiliza a *MobileNetV2* com algumas modificações como mostrado na Figura 8.

Figura 8 — MobileNet

```
mobile_net_v2 = MobileNetV2(input_shape=input_shape,
                             weights=weights,
                             alpha=alpha,
                             include_top=True)
#! Default batch norm is configured for huge networks, let's speed it up
for layer in mobile_net_v2.layers:
    if type(layer) == BatchNormalization:
        layer.momentum = 0.9
#! Cut MobileNet where it hits 1/4th input resolution; i.e. (HW/4, HW/4 C)
cut_point = mobile_net_v2.get_layer('block_3_expand_relu')
#! Now attach a small additional head on the MobileNet
model = Conv2D(filters=32, kernel_size=1, strides=1,
                activation='relu', name='head_1')(cut_point.output)
model = Conv2D(filters=32, kernel_size=1, strides=1,
                activation='relu', name='head_2')(model)
logits = Conv2D(filters=num_classes, kernel_size=1, strides=1,
                activation=None, name='logits')(model)
return Model(inputs=mobile_net_v2.input, outputs=logits)
```

Fonte: Os autores (2024).

Figura 9 — Object Detection



The image shows a configuration window titled "Object Detection (Images)" with a flask icon in the top right corner. The window has a blue-to-purple gradient background. It contains three sections: "Name" with a text input field containing "Object detection"; "Input features" with a checked checkbox labeled "Image"; and "Output features" with a list containing "1 (capacitor)".

Object Detection (Images)

Name

Object detection

Input features

☒ Image

Output features

1 (capacitor)

Fonte: Os autores (2024).

4.4.2 Treinamento do modelo

4.4.2.1 Ajuste de Hiper parâmetros

Os dados foram separados em 13% para teste e 87% para treinamento, dos quais 20% foram destinados a validação na etapa de treinamento.

Utilizando a própria plataforma do Edge Impulse conseguimos as seguintes métricas de treinamento para os dados de validação:

- F1 Score: 82.1%;
- Precisão: 98%;
- Recall: 71%.

Estes resultados foram obtidos com as configurações de hiperparâmetros mostradas na Figura 10.

Figura 10 — Treinamento

```
EPOCHS = 30
LEARNING_RATE = args.learning_rate or 0.001
BATCH_SIZE = 32

model = train(num_classes=classes,
              learning_rate=LEARNING_RATE,
              num_epochs=EPOCHS,
              alpha=0.1,
              object_weight=100,
              train_dataset=train_dataset,
              validation_dataset=validation_dataset,
              best_model_path=BEST_MODEL_PATH,
              input_shape=MODEL_INPUT_SHAPE,
              batch_size=BATCH_SIZE,
              use_velo=False,
              ensure_determinism=ensure_determinism)

disable_per_channel_quantization = False
```

Fonte: Os autores (2024).

Podemos ver também a matriz de confusão do modelo na Figura 11 que mostra que o modelo tem um comportamento bem conservador sem apresentar nenhum falso positivo.

Figura 11 — Matriz de confusão

Confusion matrix (validation set)

	BACKGROUND	CAPACITOR
BACKGROUND	100.0%	0.0%
CAPACITOR	29.4%	70.6%
F1 SCORE	1.00	0.82

Fonte: Os autores (2024).

4.4.2.2 Avaliação do modelo

Para avaliar o modelo realizamos o teste com os dados separados inicialmente e pudemos observar as seguintes métricas:

- F1 Score: 83%;
- Precisão: 97%;
- Recall: 73%;
- Acurácia: 75.58%.

Estas mostram que não houve um *overffiting* nos dados de treinamento pois são bem próximas das métricas anteriores, isso confirma que o modelo está bom e pode ser aplicado.

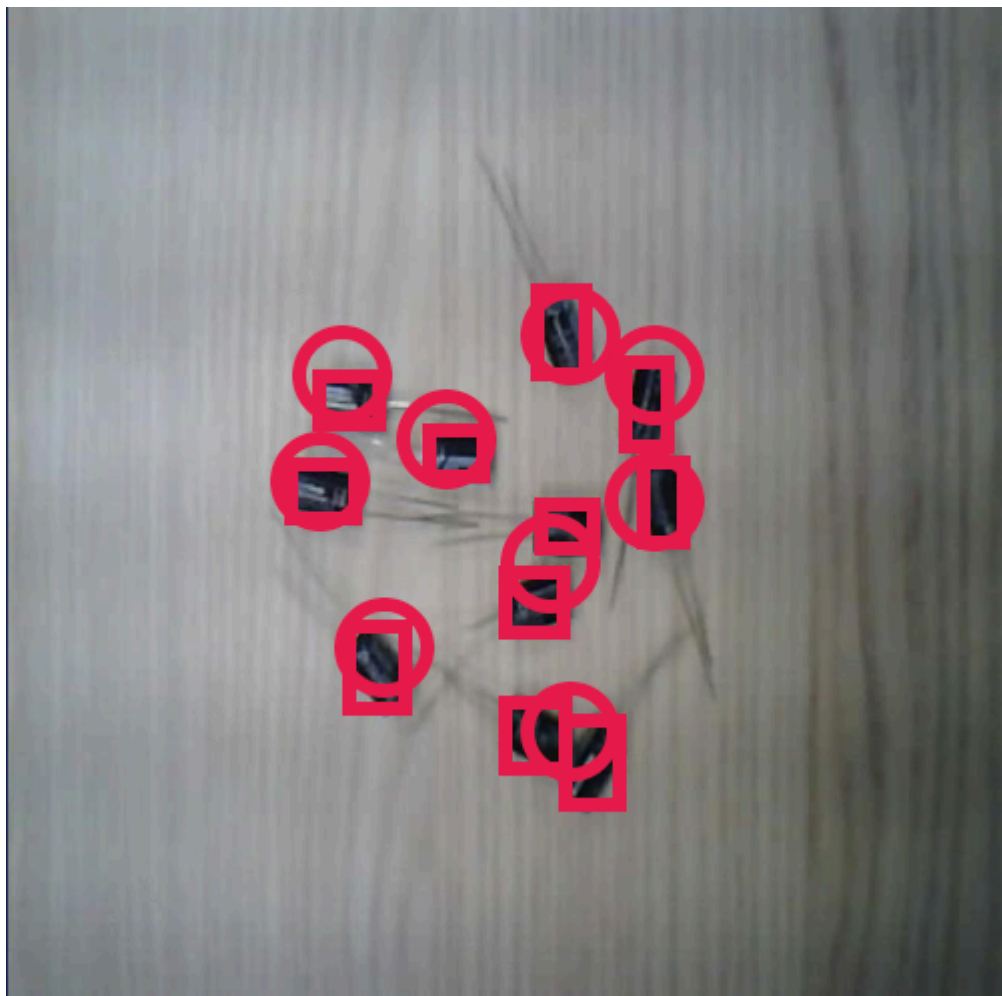
Observando algumas imagens classificadas podemos ver que o modelo identifica corretamente a maioria dos capacitores, apresentando falhas principalmente próximo as bordas da imagem e mostrando um único centroide para mais de um capacitor próximo, mas essas são características do modelo FOMO escolhido e já eram esperados.

Figura 12 — Exemplo com dados de teste



Fonte: Os autores (2024).

Figura 13 — Exemplo com dado de teste



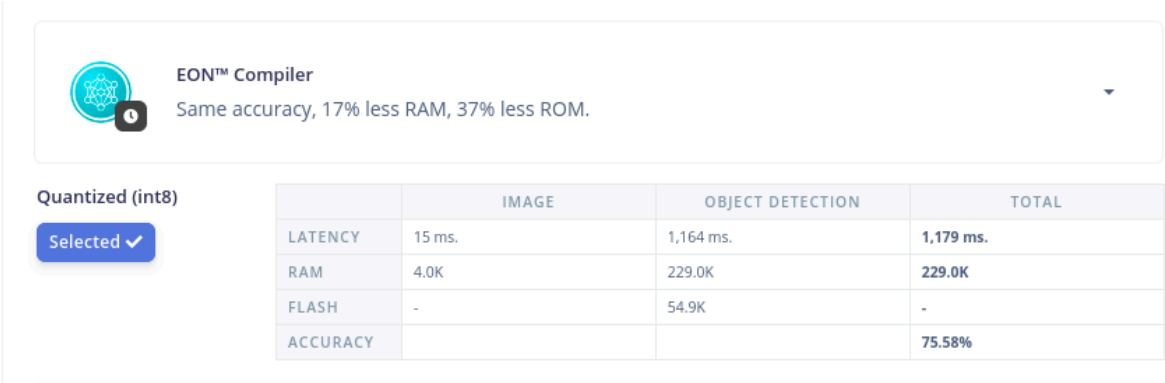
Fonte: Os autores (2024).

4.4.3 Implantação

4.4.3.1 Otimização

Para otimizar o modelo e possibilitar a execução mais eficiente no Hardware limitado característico de projetos TinyML utilizamos uma quantização para inteiros de 8 bits e o compilador EON™ que reduz a quantidade de RAM e ROM necessário para execução.

Figura 13 — Teste



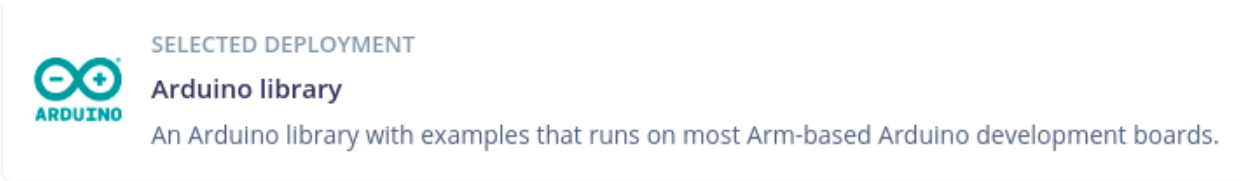
Fonte: Os autores (2024).

Como é possível ver na Figura 13 o tempo estimado para realizar a inferencia será de aproximadamente 1,2ms.

4.4.3.2 Implementação

A Implementação foi realizada através plataforma Arduino IDE, com o modelo sendo exportado como uma biblioteca arduino direto da plataforma Edge Impulse.

Figura 14 — export



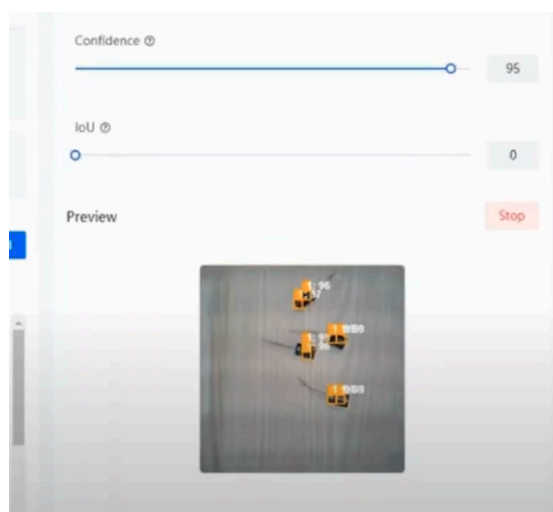
Fonte: Os autores (2024).

4.5 INFERÊNCIA

Para auxiliar na inferência utilizamos também a plataforma SenseCraft da Seed-Studio que permite uma visualização melhor da classificação realizada

diretamente no dispositivo através da web sobrepondo as imagens capturadas com os labels identificados.

Figura 15 — Inferência



Fonte: Os autores (2024).

5 CONCLUSÃO

O desenvolvimento de um sistema automatizado de monitoramento de colmeias usando TinyML mostrou-se promissor, especialmente ao considerar a implementação de um projeto semelhante para a identificação e contagem de capacitores em uma mesa. Este projeto demonstrou a viabilidade e eficiência de utilizar dispositivos de baixo consumo de energia, como o Xiao ESP32-S3, em aplicações de visão computacional.

Através do uso de técnicas de coleta e pré-processamento de dados, treinamento de modelos de machine learning, e otimização para implantação em hardware limitado, conseguimos criar um modelo capaz de realizar inferências rápidas e precisas. Os resultados obtidos, com métricas de precisão e recall satisfatórias, indicam que o sistema pode ser adaptado para aplicações mais complexas, como o monitoramento da saúde das colmeias.

A adoção de ferramentas e plataformas como Roboflow e Edge Impulse facilitou o processo de desenvolvimento, desde a rotulagem dos dados até a implantação do modelo, demonstrando que projetos de TinyML podem ser acessíveis e eficazes mesmo para desenvolvedores com recursos limitados.

Este projeto não apenas cumpriu seus objetivos como prova de conceito, mas também abriu caminho para futuras pesquisas e desenvolvimentos na área de monitoramento ambiental e agrícola, destacando a importância de soluções tecnológicas inovadoras para enfrentar desafios globais.

6 REFERÊNCIAS

ROVAI, Marcelo J. TinyML Made Easy: Object Detection with XIAO ESP32S3 Sense. Disponível em:

<https://www.hackster.io/mjrobot/tinyml-made-easy-object-detection-with-xiao-esp32s3-sense-6be28d#toc-model-design--training--and-test-13>. Acesso em: 4 jul. 2024.

ROVAI, Marcelo J. SciTinyML Scientific Use of Machine Learning on Low Power Devices. Disponível em:

https://tinyml.seas.harvard.edu/assets/slides/4D/seminars/22.03.11_Marcelo_Rovai.pdf. Acesso em: 2 mai. 2024.

Linas Kondrackis, Jacob Solawetz. (Jun 14, 2024). How to Detect Small Objects: A Guide. Roboflow Blog: <https://blog.roboflow.com/detect-small-objects/>