UNIVERSIDADE FEDERAL DE ITAJUBÁ – UNIFEI

Instituto de Engenharia de Sistemas e Tecnologia da Informação – IESTI
Curso de Engenharia de Computação

RECONHECIMENTO DE PLACAS DE TRÂNSITO

Rodrigo Zaparoli Silveira – 2023003016
Henry Boldori Cappellari – 2020017205

Itajubá – MG
2025

SUMÁRIO

1 INTRODUÇÃO

Este trabalho propõe o desenvolvimento de um sistema embarcado de baixo custo para **detecção da presença de placas de trânsito em imagens**, utilizando técnicas de aprendizado de máquina embarcado (TinyML). O sistema tem como objetivo indicar **se há ou não uma placa de trânsito** na imagem capturada.

Para isso, é utilizado um microcontrolador **Arduino Nano 33 BLE Sense**, que oferece suporte a modelos de aprendizado de máquina, juntamente com uma câmera **CMOS OV7675**. A proposta visa executar a inferência **em tempo real**, diretamente no microcontrolador.

O desenvolvimento de soluções como essa é relevante para sistemas de veículos inteligentes, especialmente em contextos com restrições de hardware e conectividade. Contudo, diversos desafios precisam ser superados, como a baixa qualidade da imagem, limitações de processamento e a grande variabilidade de contextos visuais em que uma placa pode ou não estar presente.

2 OBJETIVOS

Desenvolver um sistema embarcado baseado em TinyML, utilizando o Arduino Nano 33 BLE Sense, capaz de **detectar se há ou não uma placa de trânsito** presente em uma imagem capturada em tempo real pela câmera integrada.

3 DESCRIÇÃO DO PROJETO

O sistema funciona capturando imagens por meio da câmera embarcada e, após um pré-processamento, aplica um modelo de aprendizado de máquina previamente treinado para **classificar a imagem como "com placa" ou "sem placa"**.

Para simplificar a arquitetura e viabilizar a execução no microcontrolador, optou-se por não utilizar técnicas de localização como *bounding boxes*. Embora comuns em tarefas de detecção de objetos, tais técnicas exigem maior capacidade computacional e uso de memória RAM — o que não é compatível com os recursos limitados do Arduino Nano 33 BLE Sense.

Dessa forma, o sistema realiza uma **classificação binária da imagem inteira**, o que reduz significativamente a complexidade do modelo, acelera a inferência e melhora o desempenho geral da aplicação embarcada.

O conjunto de dados utilizado foi obtido a partir de repositórios públicos disponíveis no Kaggle, e adaptado para conter imagens com e sem placas de trânsito. Algumas imagens foram ajustadas para balancear a proporção entre as classes e evitar viés durante o treinamento.

4 METODOLOGIA

4.1 Hardware Utilizado

- Arduino Nano 33 BLE Sense
- Câmera CMOS OV7675

4.2 Coleta de Dados

Foi utilizado um dataset público disponível no Kaggle, complementado com imagens que não contêm placas de trânsito, com o objetivo de balancear as classes "com placa" e "sem placa". Imagens problemáticas foram removidas ou ajustadas manualmente para garantir maior qualidade e representatividade.

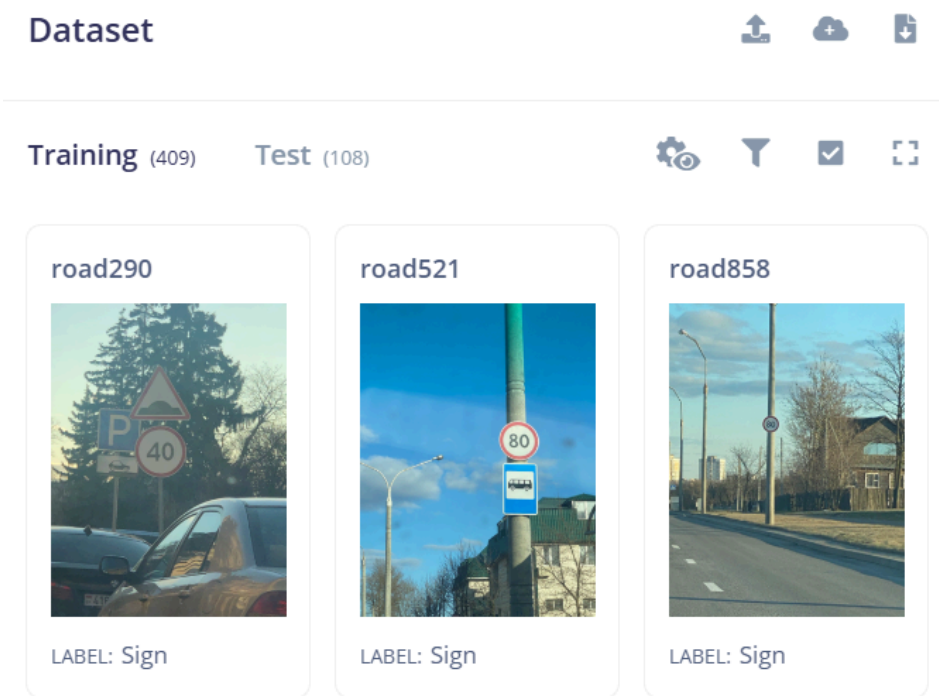

Imagem 1 – Exemplos do dataset..

4.3 Pré-processamento

As imagens foram submetidas ao seguinte pipeline:

- Redimensionamento para **96x96pixels**
- Conversão para tons de cinza (*gray-scale*)
- Normalização dos pixels para a faixa **[0,1]**

Esse processo visa padronizar os dados e reduzir a complexidade de entrada para o modelo.

Imagem 2 – Resultado do pré-processamento de uma imagem.

4.4 Arquitetura do Modelo

- Arquitetura: **Rede Neural Convolucional (CNN)**
- Frameworks: **TensorFlow**, **Edge Impulse**
- Técnica: **Transfer Learning**

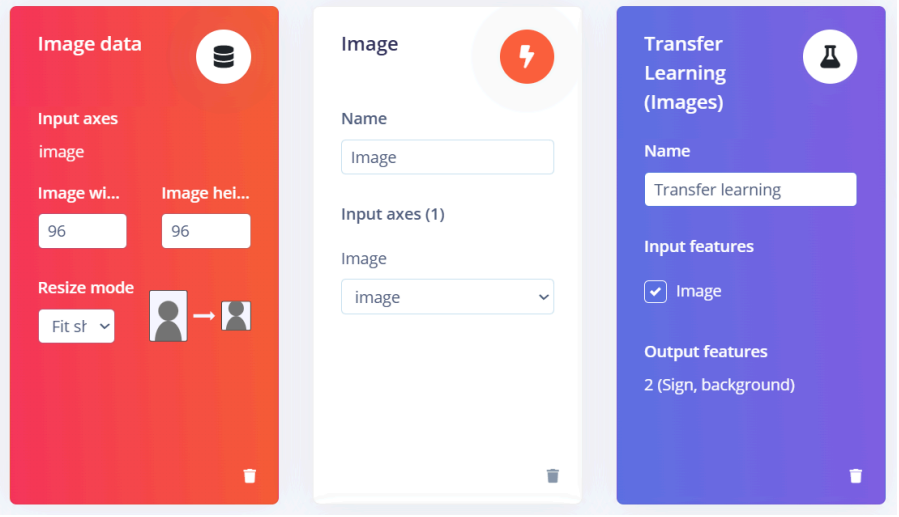

Imagem 3 – Estrutura do modelo final utilizado.

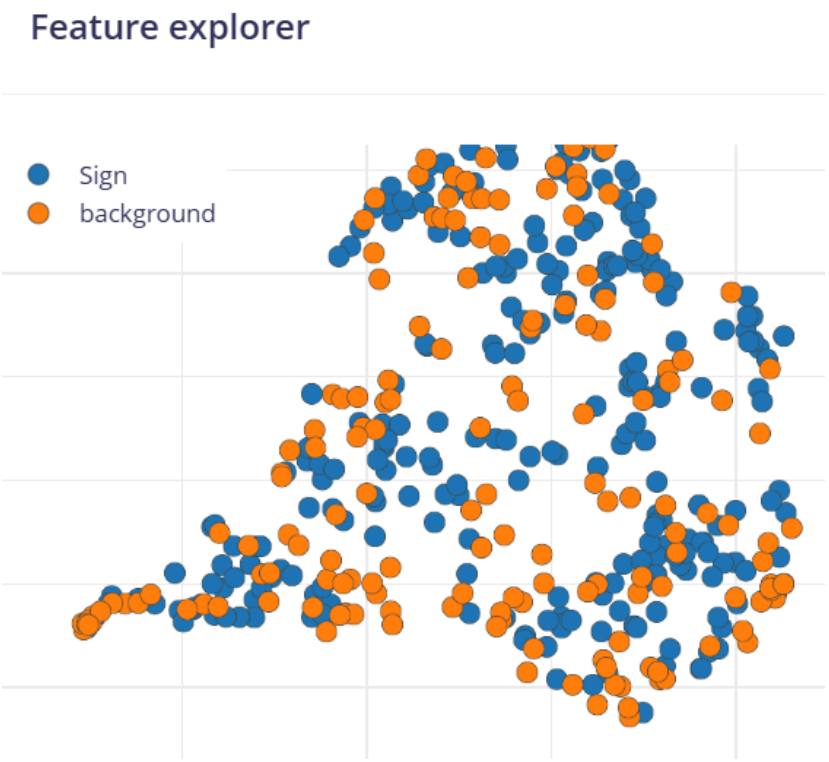

Imagem 4 – Visualização das features extraídas.

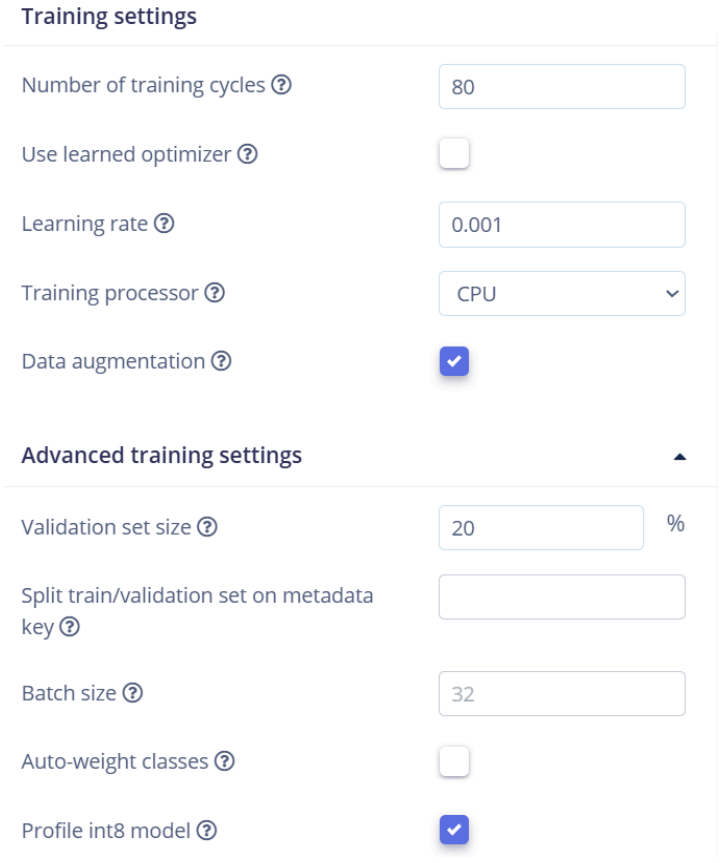

Imagem 5 - Configurações de treinamento

4.5 Resultados e precisão do modelo

Durante o processo de pré-processamento, uma das etapas fundamentais é o redimensionamento das imagens, no entanto, essa simplificação apresenta um **trade-off importante entre desempenho e precisão**. Em cenários simples, onde há apenas uma placa bem centralizada e com boa iluminação, o modelo apresenta desempenho satisfatório, sendo capaz de detectar a presença da placa com alta confiabilidade.

Em situações mais complexas, o *downscaling* pode comprometer significativamente a qualidade das informações visuais contidas na imagem. Isso inclui:

- **Presença de múltiplas placas** em uma única imagem, que acabam sendo mescladas ou diluídas em uma representação pixelada e de baixa definição;
- **Variações no ângulo de captura**, que distorcem a forma da placa e dificultam o reconhecimento por redes convolucionais simples;
- **Condições de iluminação adversas**, como sombras ou reflexos, que perdem contraste e detalhes importantes após a conversão para tons de cinza e redução de resolução.

Mesmo com essas limitações, o modelo se mostrou funcional para o propósito geral do projeto, cumprindo sua proposta de detecção (presença ou ausência de placa) com desempenho aceitável na maioria dos testes realizados. Para aplicações mais exigentes, seria necessário considerar resoluções maiores e arquiteturas mais robustas, o que exigiria hardware com maior capacidade.

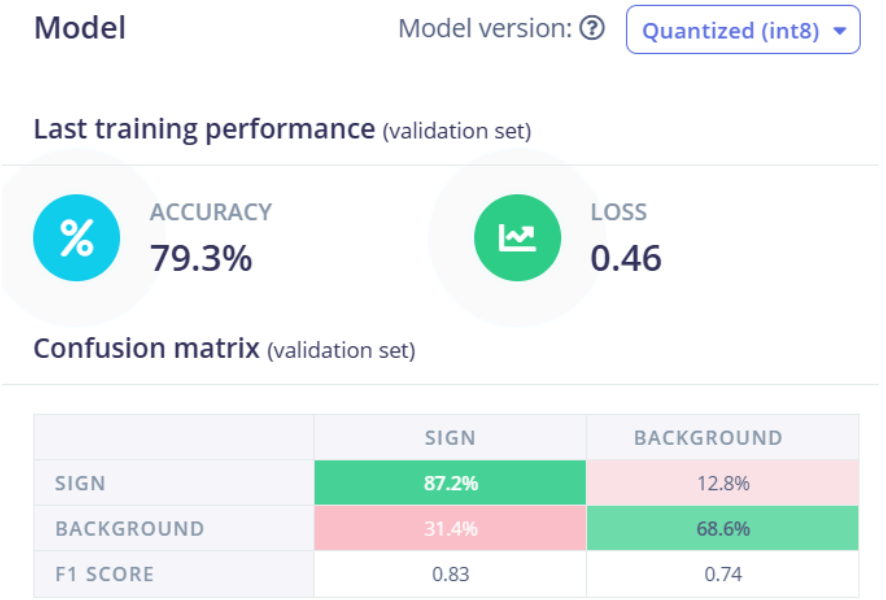

Imagem 6 - Precisão do modelo.

5 Inferência

A fase de inferência do modelo foi realizada por meio de uma simulação, utilizando a câmera da placa para capturar imagens da tela de um computador, onde eram exibidas diferentes placas de trânsito. Essa abordagem cria condições ruins para o modelo, além disso o enquadramento mostrou-se um fator importante durante a inferência. Ainda assim, o modelo demonstrou ser capaz de identificar corretamente a presença de placas na maioria das imagens exibidas.

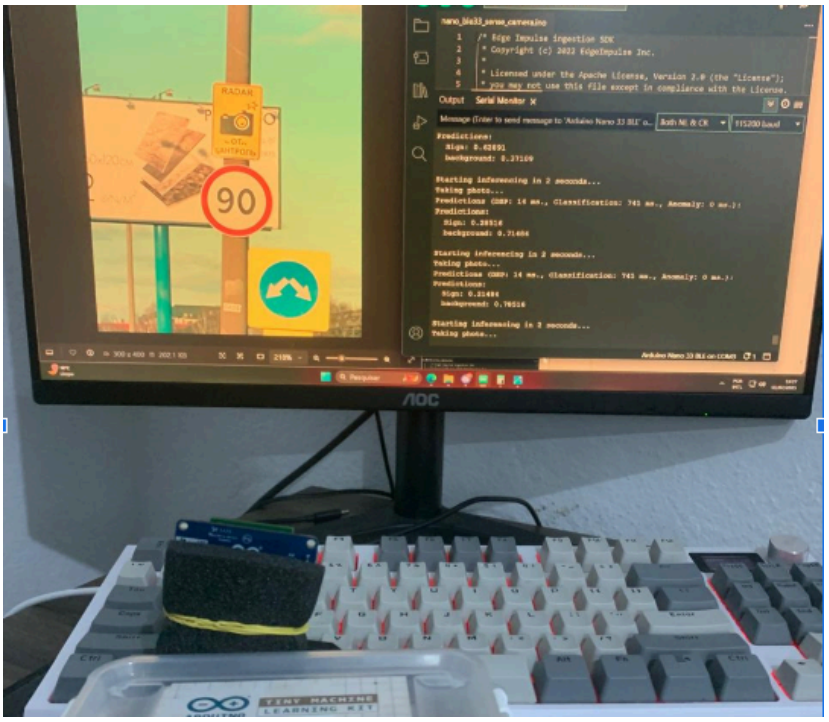

Imagem 7 - Método para inferência.

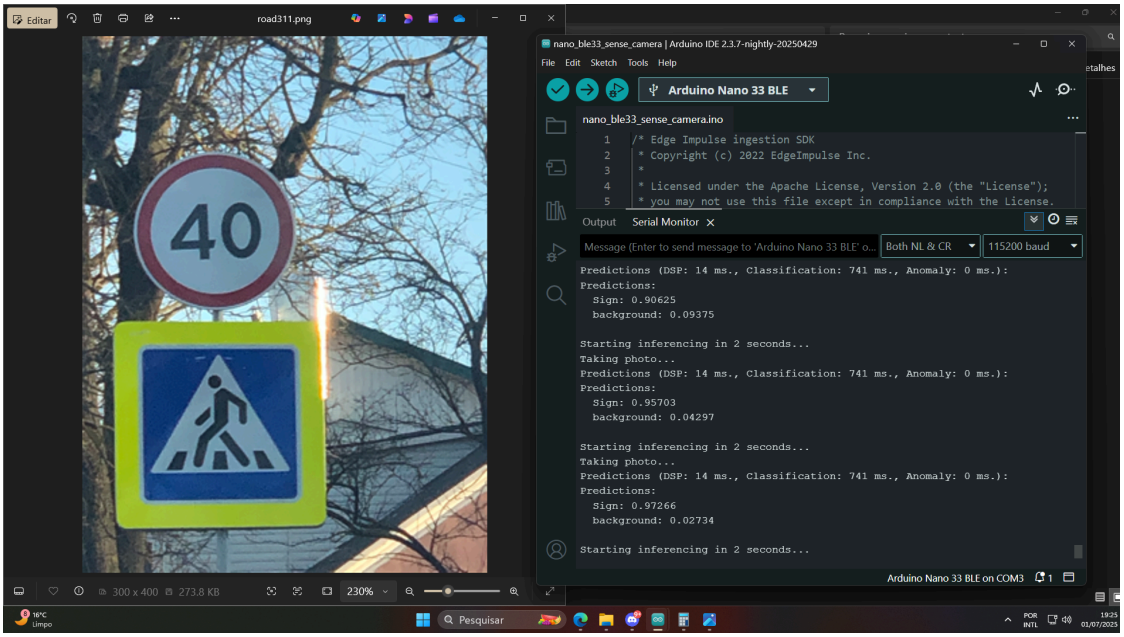

Imagem 8 - Inferência na IDE Arduino.

6 REFERÊNCIAS

KAGGLE. *Traffic Signs Image Classification 97% [CNN]*. Disponível em: https://www.kaggle.com/. Acesso em: 30 jun. 2025.

GITHUB. *aarcosg/traffic-sign-detection*. Disponível em: https://github.com/aarcosg/traffic-sign-detection. Acesso em: 30 jun. 2025.

GITHUB. *jean2612/Dataset-Placas-Transito*. Disponível em: https://github.com/jean2612/Dataset-Placas-Transito. Acesso em: 30 jun. 2025

EDGE IMPULSE**.** *Building a Speed Sign Detector with Edge Impulse* [vídeo]. YouTube, 6 meses atrás. Disponível em: https://www.youtube.com/watch?v=F0xlOGt6RJY&t=1242s. Acesso em: 1 jul. 2025.

KAGGLE**.** *Road Sign Detection* [recurso eletrônico]. Kaggle, 2020. Disponível em: https://www.kaggle.com/datasets/andrewmvd/road-sign-detection. Acesso em: 1 jul. 2025.