# Sistema de Detecção e Classificação de Resíduos Sólidos com Computação em Borda Usando Deep Learning

Ana Lídia Lisboa de Andrade*‡, Carlos Henrique Oliveira de Almeida*‡, Douglas de Araújo Rodrigues†‡, Lucas Oliveira Santos†‡, Pedro Pedrosa Rebouças Filho†‡ e Josias Guimarães Batista*‡
*Programa de Pós-graduação em Engenharia de Telecomunicações (PPGET),
Instituto Federal do Ceará, Fortaleza, Ceará, Brasil
†Programa de Pós-Graduação em Engenharia de Teleinformática (PPGETI)
Universidade Federal do Ceará, Fortaleza, Ceará, Brazil
‡Laboratório de Processamento de Imagens, Sinais e Computação Aplicada (LAPISCO), Fortaleza, Brasil
Email: analidialisboadeand@gmail.com, {henrique.almeida, douglas, lucas.santos, pedropedrosa}@lapisco.ifce.edu.br, josiasbatista@ifce.edu.br

***Abstract*—This paper presents an innovative intelligent system for the automatic detection and classification of municipal solid waste, aiming to improve the efficiency and reliability of selective collection processes. Using artificial intelligence and advanced deep learning techniques, such as the YOLOv8n model, in conjunction with TensorFlow Lite, the system is capable of identifying and classifying waste into four main categories: metal, paper/cardboard, glass, and plastic. The trained model was incorporated into a mobile application developed in Flutter, aiming to optimize selective collection and increase the accuracy of waste separation, promoting proper disposal. Users can easily interact with the application by sending or capturing images and receiving instant feedback on the type of waste detected, accompanied by standardized color codes to assist in the correct separation of materials. After comparative tests with different versions of the network, we chose to use YOLOv8n, with an average accuracy (mAP) of 95.08% and an *F1-score* of 93.12%. Developed for mobile devices, the system is an accessible tool that contributes to environmental sustainability, encouraging responsible disposal practices and facilitating use in edge computing scenarios.**

***Resumo*—Este artigo apresenta um sistema inteligente inovador para a detecção e classificação automática de resíduos sólidos urbanos, com o objetivo de melhorar a eficiência e a confiabilidade dos processos de coleta seletiva. Utilizando inteligência artificial e técnicas avançadas de deep learning, como o modelo YOLOv8n em conjunto com TensorFlow Lite, o sistema é capaz de identificar e classificar resíduos em quatro categorias principais: metal, papel/papelão, vidro e plástico. O modelo treinado foi incorporado a um aplicativo móvel desenvolvido em Flutter, visando otimizar a coleta seletiva e aumentar a precisão na separação de resíduos, promovendo o descarte adequado. Os usuários podem interagir facilmente com o aplicativo enviando ou capturando imagens e recebendo feedback instantâneo sobre o tipo de resíduo detectado, acompanhado de códigos de cores padronizados para auxiliar na correta separação dos materiais. Após testes comparativos com diferentes versões da rede, optamos por utilizar a YOLOv8n, com uma média de precisão (mAP) de 95,08% e *F1-score* de 93,12%. Desenvolvido para dispositivos móveis, o sistema é uma ferramenta acessível que contribui para a sustentabilidade ambiental, incentivando práticas responsáveis de descarte e facilitando o uso em cenários de computação em borda.**

## I. Introdução

A questão do gerenciamento de resíduos é uma preocupação crescente em nível global, especialmente considerando o aumento exponencial na geração de resíduos. Em 2016, o mundo produziu cerca de 2,01 bilhões de toneladas de resíduos sólidos urbanos, e este número está projetado para crescer para 3,4 bilhões de toneladas até 2050 [1].

No Brasil, a Política Nacional de Resíduos Sólidos (PNRS), estabelecida pela Lei nº 12.305/2010 [2], estabelece diretrizes para o gerenciamento de resíduos, promovendo responsabilidade compartilhada entre produtores, consumidores e governo. Esta lei enfatiza a importância da coleta seletiva e reciclagem, criando um marco legal que visa reduzir a quantidade de resíduos enviados para aterros sanitários e promover a reutilização de materiais.

A identificação e a classificação adequadas dos materiais recicláveis constituem etapas fundamentais para a eficácia da coleta seletiva. Resíduos como papelão, plásticos, vidros e metais apresentam valores distintos nos processos de reciclagem e demandam procedimentos específicos para sua correta reutilização. A definição precisa dos materiais a serem coletados, bem como de sua forma de separação, não apenas potencializa a eficiência do sistema de reciclagem, mas também contribui para a redução significativa do volume de resíduos destinados a aterros sanitários. De acordo com a Associação Brasileira de Empresas de Limpeza Pública e Resíduos Especiais (Abrelpe), apenas 3,5% dos resíduos gerados no Brasil foram reciclados em 2020 [3], o que indica um grande potencial de melhoria neste aspecto.

A tecnologia desempenha um papel fundamental neste processo. Este trabalho tem como objetivo contribuir para resolver o problema do gerenciamento de resíduos através da implementação de um sistema inteligente para identificação e detecção de resíduos sólidos. Utilizando ferramentas de inteligência artificial e modelos de aprendizado de máquina

como *TensorFlow Lite*, o projeto automatiza a identificação de resíduos, aumentando a eficiência na coleta e reduzindo a margem de erro humano.

A computação de borda e a inteligência artificial têm avançado rapidamente, expandindo aplicações em dispositivos móveis e sistemas de baixo consumo. Essas tecnologias buscam uma abordagem descentralizada e eficiente, onde a inferência de modelos de *deep learning* pode ocorrer diretamente em dispositivos periféricos, sem depender de servidores remotos, reduzindo latência e consumo de banda de rede [4]. Dentro deste contexto, *frameworks* como *TensorFlow Lite (TFLite)* se destacaram por sua capacidade de otimizar modelos de aprendizado de máquina para funcionar em dispositivos móveis e *IoT*. Desenvolvido pelo *Google*, o *TFLite* oferece uma solução leve e fácil de integrar, promovendo aprendizado de máquina diretamente no dispositivo [4].

Com a expansão de dispositivos de borda e suas capacidades, surgem desafios relacionados ao uso eficiente de recursos de *hardware* limitados, como processamento e memória, exigindo técnicas específicas de otimização de modelos. Em resposta, compiladores de modelos de inferência, como *TensorRT* e *Relay*, foram desenvolvidos para transformar modelos padrão de *deep learning* em versões leves prontas para dispositivos de baixo recurso.

O sistema proposto se enquadra neste contexto, implementando detecção de resíduos sólidos usando *YOLOv8n* e *TensorFlow Lite*, e integrando esses modelos em uma aplicação *Flutter* para uso em dispositivos móveis. O objetivo é demonstrar a efetividade e eficiência desta abordagem descentralizada para gerenciamento de resíduos, fornecendo uma ferramenta prática e acessível para detectar e classificar materiais recicláveis.

A ferramenta foi desenvolvida para ser intuitiva e fácil de usar. Após treinar o modelo *YOLOv8n* com o *dataset litter-detection*, disponível no *Kaggle* [5], contendo as classes Metal, Papelão, Vidro e Plástico, o modelo foi exportado para formato *.tflite* e integrado em uma aplicação *Flutter*. Os usuários podem selecionar uma imagem da galeria ou tirar uma foto diretamente através da aplicação. O sistema analisa a imagem e identifica a classe de resíduo presente, exibindo a confiança da classificação e usando cores distintas para cada classe, seguindo o padrão de coleta seletiva definido pela Resolução CONAMA nº 275/2001 [6].

## II. Metodologia

Este trabalho foi estruturado em uma sequência de etapas descritas no fluxograma da Figura 1, que apresenta uma visão geral do processo, desde a obtenção e preparação dos dados até a implementação e validação da ferramenta de detecção de resíduos sólidos no aplicativo *Flutter*. O processo é dividido em cinco principais etapas: aquisição da base de dados, treinamento e avaliação dos modelos, conversão para *TensorFlow Lite*, desenvolvimento do aplicativo em paralelo com a integração do modelo escolhido com a plataforma e testes funcionais do aplicativo.

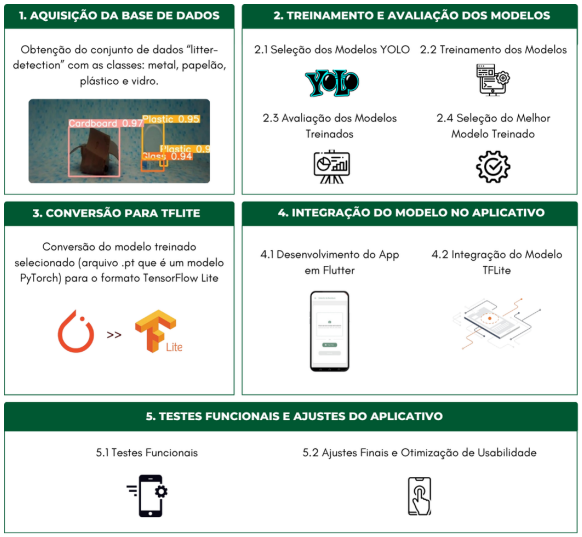


Figura 1: Metodologia empregada neste trabalho

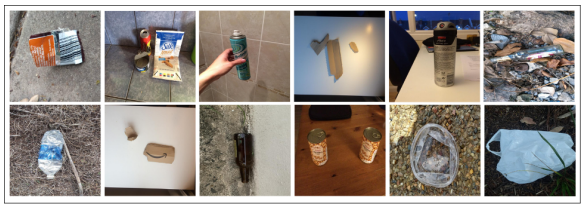

Figura 2: Exemplos de Imagens de Treinamento, Teste e Validação do *Dataset "Litter-Detection"*

### A. Aquisição da Base de Dados

A base de dados utilizada neste trabalho foi a *"litter-detection"*, disponível no *Kaggle* [5], para realizar a detecção de resíduos em imagens de diferentes ambientes. Este conjunto de dados é composto por um total de 9.841 imagens, classificadas em quatro categorias de resíduos: Metal, Papelão, Vidro e Plástico. Algumas das imagens incluem capturas feitas com câmeras específicas, como a câmera *BlueROV2*, para coleta de dados subaquáticos e terrestres, enquanto outra parte das imagens foi extraída do conjunto de dados *TACO (Trash Annotations in Context)*.

O conjunto de dados *"litter-detection"* contém 9.841 imagens, divididas em 3.194 para treinamento, 1.079 para validação e 1.000 para teste. Além disso, o conjunto de dados foi anotado com a ferramenta *Roboflow*, o que facilita a organização e o uso dos dados em projetos de visão computacional. As imagens foram processadas para garantir a consistência de tamanho (640x640) e receberam pré-processamento, como correção de orientação e aumento de contraste, para aprimorar o desempenho do modelo durante o treinamento. Na Figura 2 é possível visualizar algumas imagens do conjunto de dados.

### B. Treinamento e Avaliação dos Modelos YOLO

*YOLO (You Only Look Once)* é uma abordagem eficiente para detecção de objetos, transformando o problema em re-

gressão para prever simultaneamente caixas delimitadoras e probabilidades de classes em uma única avaliação. Otimizada para tempo real, a rede neural reduz falsos positivos em fundos complexos, superando métodos anteriores. [7].

Para determinar a abordagem mais eficaz de detecção, diferentes versões dos modelos *YOLO* foram avaliadas, tais versões foram: *YOLOv8n, YOLOv9t* e *YOLOv10n*. Inicialmente, todos os modelos foram submetidos a um processo de treinamento, utilizando a linguagem *Python*, na pasta de treinamento do *dataset* de resíduos para avaliar o desempenho de cada um em termos de precisão, sensibilidade, média de precisão média *(mean average precision, mAP)* e *F1-score*. O modelo escolhido foi a *YOLOv8n* devido à necessidade de otimização entre desempenho e consumo de recursos. O treinamento dos modelos foi realizado no ambiente *Python* utilizando a biblioteca *Ultralytics*, que permite uma configuração robusta e flexível dos parâmetros de treino, além de suporte nativo para versões do *YOLO*. O processo incluiu o ajuste dos hiperparâmetros, como as épocas e tamanho do *batch*, otimizados para obter o melhor equilíbrio entre precisão e desempenho computacional.

### C. Conversão para TFLite

O *TensorFlow Lite (TFLite)* é um *framework* de aprendizado de máquina leve e de código aberto, projetado para permitir a inferência em dispositivos móveis e *IoT*. Integrado ao ecossistema *TensorFlow*, o *TFLite* suporta diversas plataformas, como *Android*, *iOS* e microcontroladores *(MCUs)*, tornando-se uma escolha ideal para desenvolvedores que buscam implementar soluções de aprendizado de máquina em ambientes com recursos limitados. Sua abordagem de *Edge Computing* possibilita a execução de tarefas complexas, como detecção de objetos e reconhecimento de imagem, diretamente nos dispositivos que geram os dados, reduzindo a latência, economizando largura de banda e melhorando a privacidade [4].

Esse processo de implantação é viabilizado por um fluxo de conversão eficiente, parcialmente ilustrado na Figura 3, que detalha a transformação de modelos *TensorFlow* em arquivos *TFLite* otimizados. Conforme descrito por Li Shuangfeng, a Figura 3, intitulada *"TFLite Model Conversion"* (Conversão de Modelo TFLite), mostra como um modelo criado com APIs do *TensorFlow* é processado pelo *TFLite Converter*, gerando um arquivo *.tflite* baseado em *FlatBuffers*. Esse arquivo é então executado em dispositivos de borda pelo *TFLite Interpreter*, que pode aproveitar aceleradores de *hardware*, como *GPUs*, via delegates, para otimizar a inferência. Neste trabalho, o modelo *YOLO* treinado exportado no formato *PyTorch* segue um pipeline mais amplo, conforme detalhado na Figura 4, que ilustra todo o fluxo de conversão desde o modelo original até a obtenção do arquivo *.tflite* otimizado para execução em dispositivos embarcados.

### D. Desenvolvimento e Integração do Aplicativo Flutter

O *Flutter* é um *framework* de desenvolvimento de aplicações móveis multiplataforma criado pelo *Google*, que utiliza a linguagem de programação *Dart*. Uma das principais

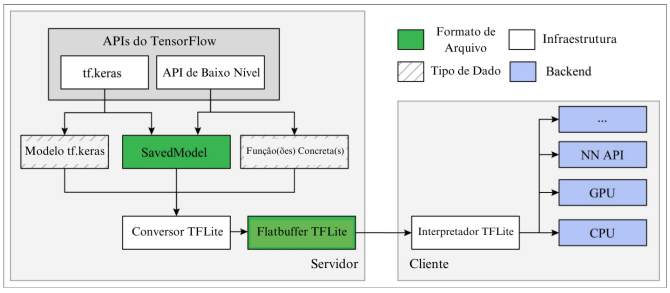


Figura 3: Conversão de modelo *TFLite*

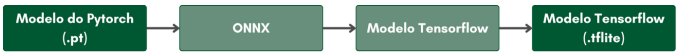


Figura 4: Fluxo de Conversão do Modelo para *TFLite*

vantagens do *Flutter* é sua capacidade de criar aplicações nativas para *iOS* e *Android* a partir de um único código base, reduzindo significativamente o tempo de desenvolvimento e os custos de manutenção. O *framework* oferece um desempenho próximo ao nativo através de sua arquitetura baseada em *widgets*, permitindo a criação de interfaces de usuário altamente personalizáveis e responsivas. Além disso, o *Flutter* possui um extenso ecossistema de *plugins* e bibliotecas que facilitam a integração com funcionalidades específicas do dispositivo, como câmera, armazenamento e processamento de *machine learning*, tornando-o uma escolha ideal para aplicações que requerem recursos avançados como inferência de modelos de IA diretamente no dispositivo.

O desenvolvimento da interface do aplicativo no *Flutter* foi projetado para oferecer uma experiência de uso intuitiva, com foco na facilidade de navegação e acessibilidade das funcionalidades principais. A interface permite ao usuário selecionar ou capturar uma imagem diretamente, proporcionando uma experiência integrada e fluida, como demonstrado na Figura 5. O aplicativo exibe as opções de detecção e apresenta a previsão de classe de resíduo. A demonstração das telas do sistema ilustra como essas funcionalidades são organizadas, facilitando a interação do usuário com o aplicativo.

A integração do modelo *TFLite* ao aplicativo *Flutter* foi realizada utilizando a biblioteca *"tflite_flutter"*, que possibilita a execução de modelos *TensorFlow Lite* diretamente no dispositivo móvel. Essa biblioteca permite que o modelo embarcado execute inferências nas imagens fornecidas pelo usuário, detectando quatro tipos de resíduos — metal, papel/papelão, vidro e plástico —, o que é crucial para garantir uma experiência fluida e responsiva. A arquitetura do sistema, dividida em três camadas principais, conforme apresentada na Figura 6, inclui a camada de Interface, com uma página inicial minimalista contendo a logo e um botão de início, uma tela de captura para tirar fotos ou selecionar da galeria, e uma tela de resultados visualmente intuitiva; a camada de Processamento, responsável pela lógica de tratamento de imagens e filtragem de resultados; e a camada de *Machine Learning*, onde o modelo *TFLite* opera. Essas funções desenvolvidas na

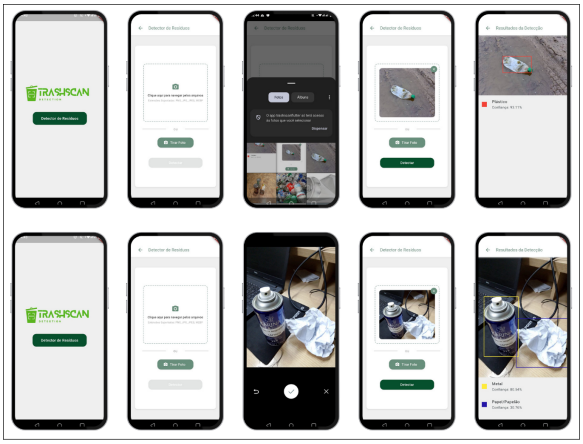

Figura 5: Fluxo do Aplicativo *TrashScan* e Resultados da Detecção de Resíduos

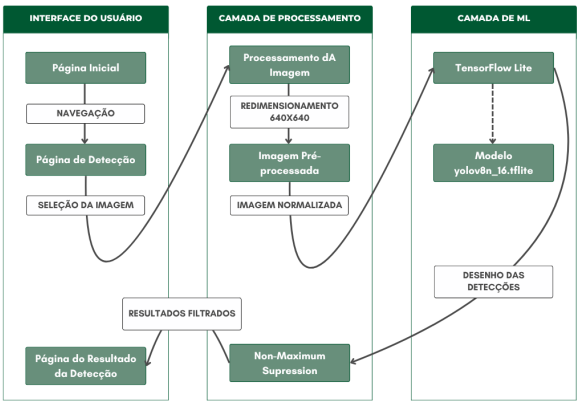


Figura 6: Arquitetura do Sistema

aplicação oferecem uma resposta rápida e eficaz, essencial para a usabilidade do aplicativo, funcionando inteiramente no dispositivo sem necessidade de conexão com a internet.

A implementação inclui a preparação da imagem para inferência através de métodos específicos, que convertem a imagem redimensionada em um tensor adequado para o modelo. Durante a inferência, a imagem é carregada e redimensionada para 640x640 *pixels* — tamanho exigido pelo modelo —, seguida pela normalização dos *pixels* de 0-255 para uma escala de 0-1, melhorando a precisão das previsões. O modelo, carregado na inicialização da página, analisa a imagem utilizando um sistema de grades (80x80, 40x40 e 20x20), totalizando 8400 pontos de análise, gerando para cada ponto 8 valores: 4 para a localização do objeto (centro x, centro y, largura e altura) e 4 para as confidências das classes de resíduos. Os resultados da inferência são processados com o algoritmo Non-Maximum Suppression (NMS), que elimina detecções duplicadas e mantém apenas as mais confiáveis, assegurando que apenas as previsões mais relevantes sejam apresentadas. O contexto de navegação do Flutter então redireciona o usuário à página de resultados, exibindo as detecções e a imagem original, proporcionando uma interação completa e eficiente que facilita a compreensão dos resultados.

Para facilitar a identificação de resíduos, o aplicativo exibe cores da coleta seletiva (vermelho para plástico, azul para papel/papelão, verde para vidro, amarelo para metal) junto às classes e níveis de confiança, promovendo categorização correta, conscientização ambiental e uma interface intuitiva que incentiva práticas sustentáveis de reciclagem.

### *E. Testes Funcionais e Ajustes do Aplicativo*

Para garantir a eficácia do modelo, foram realizados testes funcionais em condições reais, avaliando o aplicativo com diferentes resíduos e em diversos ambientes. A precisão das classificações foi monitorada para garantir que o sistema estivesse detectando corretamente os tipos de resíduo e respondendo conforme o esperado. Além disso, o desempenho da aplicação em dispositivos móveis foi avaliado, verificando o tempo de resposta e a usabilidade da interface. Após os testes iniciais, ajustes foram feitos na interface do aplicativo e no tempo de inferência, visando otimizar a experiência do usuário. A interface foi refinada para ser mais intuitiva, enquanto o desempenho do modelo foi ajustado para minimizar o uso de recursos do dispositivo, garantindo uma execução rápida e eficiente.

### *F. Métricas de Avaliação*

Para fundamentação da escolha do modelo entre as versões *YOLOv8n, YOLOv9t* e *YOLOv10n*, foram utilizadas algumas métricas de detecção conhecidas e amplamente utilizadas na literatura para avaliar nossa rede. Tais métricas serão descritas abaixo:

**mAP@0.5**: A média da média de precisões *(Mean Average Precision)* calculada com um limiar de 0.5 para a interseção sobre união (IoU). Essa métrica calcula a média da precisão em várias classes e é utilizada para avaliar a eficácia do modelo em detectar corretamente objetos em uma imagem. Ela fornece uma visão geral de quão bem o modelo está detectando objetos e separando-os das áreas de fundo.

$$\text{mAP@0.5} = \frac{1}{C}\sum_{c=1}^{C}\text{AP}_c \tag{1}$$

**Precisão**: Representa a razão entre a quantidade de objetos detectados corretamente e o total de detecções realizadas como objetos. Em termos práticos, quanto maior a precisão, menor a chance de o modelo gerar falsos positivos (objetos detectados incorretamente como positivos).

$$\text{Precisão} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

**Sensibilidade *(Recall)***: É a razão entre a quantidade de objetos detectados corretamente como positivos e a quantidade total de objetos que são, de fato, positivos. A sensibilidade elevada indica que o modelo está identificando a maioria dos objetos relevantes, com menos falsos negativos.

$$\text{Sensibilidade} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

**F1-Score**: Calcula uma média harmônica entre a precisão e a sensibilidade, equilibrando essas duas métricas para uma avaliação mais robusta do modelo. Ele é especialmente útil em contextos onde é importante ter um equilíbrio entre a capacidade do modelo de encontrar objetos e evitar falsos alarmes.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Sensibilidade}}{\text{Precisão} + \text{Sensibilidade}} \tag{4}$$

## III. Resultados e Discussão

### A. Configuração do Treinamento e Desempenho dos Modelos

As três variantes nano das arquiteturas *YOLO* (*YOLOv8n, YOLOv9t* e *YOLOv10n*) foram treinadas em 500 épocas, utilizando *batch size* de 8. A avaliação dos modelos foi conduzida exclusivamente no conjunto de teste independente (1.000 imagens), garantindo uma avaliação imparcial da capacidade de generalização. A Tabela 1 apresenta as métricas de desempenho para cada rede, incluindo a Precisão, Sensibilidade, *F1-Score* e *mAP@0.5*. O *YOLOv8n* destacou-se como o melhor modelo, alcançando um *mAP@0.5* de 95,08%, precisão de 96,13% e sensibilidade de 90,29%, resultando em um *F1-score* de 93,12%. Este modelo demonstrou convergência eficiente, atingindo seu desempenho ótimo na época 399. O *YOLOv10n* e *YOLOv9t* apresentaram níveis de desempenho similares, com valores de *mAP@0.5* de 94,85% e 94,78% respectivamente. Destaca-se que o *YOLOv10n* alcançou a maior precisão (96,93%) entre todos os modelos testados, embora com sensibilidade *(recall)* ligeiramente inferior (91,57%). Ambos os modelos atingiram seu melhor desempenho próximo ao final do treinamento (épocas 491 e 489 respectivamente).

Foi realizada uma análise de estabilidade nas últimas 50 épocas de treinamento de cada modelo (Tabela 2), empregando medidas estatísticas incluindo média, desvio padrão e coeficiente de variação (CV). A escolha por 50 épocas foi motivada pela necessidade de avaliar a convergência e a consistência dos modelos após um período suficiente de treinamento, permitindo identificar padrões estáveis de desempenho nas fases finais do processo. Esta análise foi implementada através de um *script Python* abrangente que processou os resultados de treinamento do arquivo *results.csv* de cada modelo, calculando métricas de estabilidade.

A análise revelou que o *YOLOv8n* e *YOLOv9t* foram os modelos mais estáveis, ambos exibindo um coeficiente de variação de 0,02%. Isso indica um desempenho altamente consistente durante a fase final do treinamento. O modelo *YOLOv10n* também apresentou boa estabilidade, com coeficiente de variação de 0,04%.

### B. Justificativa da Seleção do Modelo

A seleção do *YOLOv8n* como modelo ótimo para a aplicação foi fundamentada em uma análise abrangente e multifacetada que considerou não apenas métricas de desempenho isoladas, mas também requisitos específicos de aplicações de computação de borda e as limitações inerentes a dispositivos móveis.

*1) Primazia do mAP@0.5 como Métrica Determinante:* A métrica *mAP@0.5 (Mean Average Precision)* foi estabelecida como critério principal de seleção por representar a avaliação mais rigorosa e abrangente do desempenho de modelos de detecção de objetos. Diferentemente de métricas individuais como precisão e sensibilidade, que avaliam aspectos específicos do desempenho, o *mAP@0.5* fornece uma medida integrada que considera simultaneamente a capacidade de localização precisa *(IoU* $\geq$ *0.5)* e classificação correta. Esta métrica é reconhecida como padrão-ouro na literatura de visão computacional por sua capacidade de capturar nuances de performance que métricas isoladas podem mascarar, sendo particularmente crítica em aplicações multiclasse como a classificação de resíduos sólidos.

O *YOLOv8n* demonstrou superioridade inequívoca no *mAP@0.5* com 95.08%, superando o *YOLOv10n* (94.85%) e *YOLOv9t* (94.78%). Esta vantagem de 0.23 pontos percentuais sobre o *YOLOv10n*, embora aparentemente modesta, representa uma melhoria estatisticamente significativa na capacidade geral de detecção e localização de objetos, traduzindo-se em maior confiabilidade operacional em cenários reais de aplicação.

*2) Estabilidade Operacional Superior:* A análise de estabilidade revelou-se um diferencial crítico na seleção do modelo. Conforme demonstrado na Tabela 2, o *YOLOv8n* apresentou estabilidade no *mAP@0.5* (desvio padrão de 0.02%, CV de 0.02%). Esta estabilidade multidimensional, particularmente evidente no *mAP@0.5*, equipara-se ao *YOLOv9t* e supera significativamente o *YOLOv10n* (CV de 0.04% no *mAP@0.5*).

A estabilidade superior no *mAP@0.5* é especialmente relevante pois indica consistência na capacidade de detecção e localização ao longo do treinamento, sugerindo maior robustez do modelo em condições operacionais variadas. Em contextos de computação em borda e aplicações móveis, onde recursos computacionais são limitados e condições ambientais podem variar significativamente, a estabilidade do modelo torna-se um requisito crítico para garantir performance previsível e confiável.

*3) Robustez em Ambientes Heterogêneos:* O *dataset "litter-detection"* apresenta características desafiadoras, incluindo imagens subaquáticas e terrestres com variações significativas de iluminação, texturas de fundo complexas e condições de captura diversas. Neste contexto heterogêneo, o *mAP@0.5* superior do *YOLOv8n* demonstra maior robustez e capacidade de adaptação a cenários diversos, indicando performance mais consistente em condições reais de aplicação onde a variabilidade ambiental é esperada.

### C. Análise Qualitativa dos Resultados

A alta precisão do *YOLOv8n* (96,13%) e o *mAP@0.5* de 95,08% indicam que o modelo é altamente confiável para identificar resíduos em imagens de ambientes variados, como contextos urbanos e subaquáticos. No entanto, a experiência prática com o aplicativo revelou que a usabilidade é significativamente aprimorada pela interface visual, que utiliza cores padronizadas conforme a Resolução CONAMA nº 275/2001 [6].

Essa abordagem facilita a identificação imediata das categorias de resíduos pelos usuários, promovendo maior engajamento na separação correta de materiais recicláveis.

Durante os testes funcionais, observou-se que o sistema é particularmente eficaz em cenários com resíduos bem definidos e isolados, como uma garrafa *PET* em uma superfície limpa. Contudo, em ambientes com múltiplos resíduos sobrepostos ou fundos complexos, como lixões ou áreas urbanas densas, o modelo ocasionalmente apresenta dificuldades na distinção entre classes, como plásticos e vidros transparentes. Esses desafios são consistentes com limitações reportadas na literatura para modelos *YOLO* em cenários de alta variabilidade visual [7].

Comparado a métodos tradicionais de classificação manual ou sensores físicos, o sistema proposto é mais acessível e escalável, especialmente para comunidades com recursos limitados. Integrado via *Flutter*, o aplicativo permite uso por catadores e cidadãos, funcionando *offline* e viabilizando a detecção de resíduos em áreas remotas.

### D. Limitações do Modelo

Embora o modelo *YOLOv8n* tenha apresentado desempenho robusto, com *mAP@0.5* de 95,08%, algumas limitações devem ser consideradas. A escolha da versão nano do *YOLOv8* foi motivada pela necessidade de eficiência em dispositivos móveis, mas isso implica uma capacidade reduzida para detectar resíduos em imagens com múltiplos objetos pequenos ou em cenários com alta densidade de resíduos. O modelo também pode apresentar falsos positivos em casos de resíduos com características visuais semelhantes, como plásticos transparentes confundidos com vidro. A implementação em computação de borda, embora eficiente, enfrenta desafios relacionados à variabilidade de hardware em dispositivos móveis. Dispositivos com menor capacidade computacional podem sofrer com tempos de inferência mais longos, impactando a experiência do usuário.

Tabela I: Desempenho das *YOLO's* Avaliadas: Acurácia e Métricas Relacionadas

| **Modelo** | **mAP@0.5** | **Precisão** | **Sensibilidade** | **F1-Score** |
|---|---|---|---|---|
| YOLOv8n | 95.08% | 96.13% | 90.29% | 93.12% |
| YOLOv9t | 94.78% | 96.24% | 92.17% | 94.16% |
| YOLOv10n | 94.85% | 96.93% | 91.56% | 94.17% |

Tabela II: Análise de Estabilidade (Últimas 50 épocas)

| **Modelo** | **mAP@0.5** | | | **Precisão** | | **Sensibilidade** | |
|---|---|---|---|---|---|---|---|
| | **Média** | **DP** | **CV** | **Média** | **DP** | **Média** | **DP** |
| YOLOv8n | 94.99% | 0.02% | 0.02% | 94.33% | 0.39% | 91.99% | 0.30% |
| YOLOv9t | 94.75% | 0.02% | 0.02% | 95.95% | 0.37% | 92.51% | 0.40% |
| YOLOv10n | 94.80% | 0.04% | 0.04% | 96.41% | 0.04% | 91.32% | 0.18% |

## IV. Trabalhos Futuros

Como sugestões para trabalhos futuros, a detecção em tempo real poderia ser explorada através do uso do fluxo de vídeo da câmera do dispositivo, permitindo a identificação contínua de resíduos de forma dinâmica. Outra ideia seria a integração com sistemas de gerenciamento de resíduos, desenvolvendo *APIs* que conectem a aplicação aos sistemas municipais de coleta seletiva, com recursos de geolocalização para indicar pontos de coleta próximos ao usuário e otimizar o descarte adequado. Além disso, expandir o *dataset* para incluir mais categorias de resíduos além de metal, papel/papelão, vidro e plástico pode aumentar a capacidade de detecção do modelo, tornando-o mais abrangente para reciclagem e sustentabilidade.

## V. Conclusão

Este trabalho desenvolveu uma solução inovadora para a detecção e classificação de resíduos sólidos recicláveis utilizando redes neurais da arquitetura *YOLO*. Após uma avaliação minuciosa de três modelos nano (*YOLOv8n, YOLOv9t* e *YOLOv10n*), o *YOLOv8n* foi selecionado como o modelo ideal para a aplicação móvel. Os resultados demonstraram uma ótima performance do *YOLOv8n*, com as seguintes métricas: *mAP@0.5* de 95,08%, precisão de 96,13%, sensibilidade de 90,29% e *F1-Score* de 93,12%. A conversão do modelo para *TensorFlow Lite* e sua integração na aplicação *Flutter* permitiu a criação de uma ferramenta acessível e intuitiva para detecção de resíduos. A interface visual com cores padronizadas da coleta seletiva facilita a identificação de resíduos, promove consciência ambiental e incentiva a reciclagem. Testes e ajustes garantiram uma aplicação responsiva e eficiente para dispositivos móveis com recursos limitados. A estabilidade do modelo, evidenciada pelo baixo coeficiente de variação de 0,02%, reforça sua confiabilidade sob condições reais de operação.

## VI. Agradecimentos

Agradecemos a Deus em primeiro lugar e também expressamos nossa gratidão ao Laboratório de Processamento de Imagens, Sinais e Computação Aplicada (LAPISCO) pelo suporte técnico e pelos recursos disponibilizados, fundamentais para a realização desta pesquisa.

## Referências

[1] S. Kaza, L. C. Yao, P. Bhada-Tata, and F. Van Woerden, *What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050*, ser. Urban Development. Washington, DC: World Bank, 2018.

[2] Brazil, "Establishes the national solid waste policy and provides other measures - law no. 12,305, of august 2, 2010," Brasília, DF, August 2 2010.

[3] Brazilian Association of Public Cleaning and Waste Companies Specials (ABRELPE), "Panorama of solid waste in brazil 2020," 2020.

[4] S. Li, "Tensorflow lite: On-device machine learning framework," *Journal of Computer Research and Development*, vol. 57, no. 9, pp. 1839–1853, 2020.

[5] D. Martinovci, "Litter detection: On-land and underwater image recognition dataset for litter detection," Available on Kaggle. [Online]. Available: https://www.kaggle.com/datasets/davianmartinovci/litter-detection/data

[6] Brazil. National Council for the Environment (CONAMA), "Defines the color code for the different types of waste in selective collection - resolution no. 275, of 25 april 2001." Brasília, DF, April 25 2001.

[7] J. Redmon *et al.*, "You only look once: Unified, real-time object detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, USA, 2016, pp. 779–788.