

Proposta de Projeto

Detenção de Lixo no Chão utilizando Visão Computacional

Trabalho proposto por:

Frederico Cerqueira – nº64374

Gonçalo Abreu – nº64462

Joana Chuço – nº64853

Grupo 3

Docente:

Professor Nuno Cruz Grácia

Milestone 0

1.Problema

Com o crescente problema da poluição urbana, a necessidade de soluções de limpeza eficazes e inovadoras torna-se cada vez mais premente. A escassez de mão de obra, os custos elevados associados à limpeza manual e a ineficiência dos métodos tradicionais contribuem para o agravamento da situação, representando um perigo significativo para o ambiente e para a saúde pública.

Neste contexto, o nosso grupo propõe o desenvolvimento de um algoritmo de deteção de lixo no chão, concebido para ser integrado em futuros robôs autônomos de limpeza urbana. Esta solução tecnológica visa colmatar a lacuna deixada pela falta de recursos humanos e materiais, enquanto aumenta a rapidez e a eficiência da recolha de resíduos. Ao automatizar o processo de deteção e recolha de lixo, pretendemos contribuir para a criação de cidades mais limpas, sustentáveis e saudáveis.

2.Fontes

Para desenvolver um algoritmo eficaz de deteção de lixo utilizando *deep learning*, e a utilização das melhores práticas no desenvolvimento do algoritmo de deteção de lixo, serão analisadas diversas fontes e em repositórios públicos de dados e código.

Um estudo relevante é "*Uma Aplicação de Segmentação com R-CNNs da Família YOLO*" (Oliveira & Guedes, 2024), que explora o uso de modelos YOLO para a segmentação de resíduos em imagens de regiões costeiras. Este trabalho destaca a eficácia do YOLOv7 na deteção de lixo e servirá como referência para avaliar o desempenho de diferentes arquiteturas de *deep learning*.

Outro artigo é "*Sistema de deteção de resíduos em rios e nascentes utilizando...*" (Valente, Coelho, & Assis, 2024), que propõe um sistema de deteção de resíduos para dispositivos móveis e pequenos computadores. A abordagem apresentada será útil para compreender a implementação de modelos otimizados para execução em hardware com recursos limitados.

Além disso, será analisada a pesquisa "*Deep Convolutional Neural Networks Object Detector for Real-Time...*" (Melinte, Travediu, & Dumitriu, 2020), que apresenta a base de dados *TrashNet*. Este conjunto de dados contém imagens classificadas de diferentes tipos de resíduos sólidos urbanos e tem sido utilizado para treinar modelos de deep learning em tarefas de detecção e classificação de lixo.

Para aprofundar os conceitos teóricos e práticos, será utilizado o material "*Dive into Deep Learning*" (Zhang, Lipton, Li, & Smola, 2023), que oferece uma introdução detalhada às redes neurais e à visão computacional aplicada, cobrindo desde os fundamentos matemáticos até implementações práticas.

Por fim, será considerada a pesquisa "*pLitterStreet: Street Level Plastic Litter Detection and Mapping*" (Mandhati, et al., 2024), que apresenta um conjunto de dados aberto com mais de 13.000 imagens anotadas de resíduos plásticos em ambientes urbanos. Esse dataset pode ser utilizado para treinar e avaliar o modelo proposto, garantindo maior representatividade dos cenários reais.

3.Recolha de Dados

Para o desenvolvimento deste projeto, utilizaremos conjuntos de dados disponíveis no Kaggle, como dados e código. Além disso, será utilizado o conjunto de dados TACO (Trash Annotations in Context) (Proença & Simões, 2020), um dataset aberto composto por imagens de resíduos em diversos ambientes, anotadas e segmentadas segundo uma taxonomia hierárquica.

Caso os conjuntos de dados existentes não sejam suficientes para atender às necessidades específicas do projeto, consideraremos a criação de um dataset próprio, garantindo maior adequação às condições reais e aos objetivos do modelo.

4.Algoritmo utilizado

Para demonstrar a capacidade de inferência a partir de imagens não vistas, serão implementados modelos pré-treinados para detecção de resíduos. Dentre as opções disponíveis, utilizaremos o *Single Shot MultiBox Detector* (Liu, et al., 2016), um modelo otimizado para detecção em tempo real, equilibrando precisão e eficiência computacional. O pipeline envolverá o pré-processamento dos dados, o treinamento do modelo e a inferência sobre novas imagens, utilizando conjuntos de dados previamente anotados.

Além disso, a implementação será realizada com a biblioteca *TensorFlow*, que fornecerá suporte às operações de aprendizagem profunda e otimização do modelo. Como referência prática, analisaremos abordagens já disponíveis, como a implementação do notebook *Garbage Detection with TensorFlow* (Ceunen, 2021), que apresenta um fluxo de trabalho semelhante.

Caso o desempenho do modelo não atenda às expectativas, serão exploradas alternativas como YOLOv8 ou Faster R-CNN, que podem oferecer melhor precisão dependendo dos requisitos computacionais do sistema.

5.Resultados esperados

Os resultados serão avaliados tanto qualitativamente quanto quantitativamente para garantir a eficácia do modelo na detecção de resíduos.

Qualitativamente, espera-se que o modelo crie imagens com caixas delimitadoras ao redor dos objetos identificados, acompanhadas de rótulos que indicam a classe prevista para cada item. Essas inferências serão baseadas nos dados previamente utilizados para o treinamento, permitindo uma visualização clara da capacidade do modelo em reconhecer diferentes tipos de resíduos. Além disso, o objetivo é que a detecção ocorra em tempo real, utilizando uma câmara para identificar automaticamente os resíduos no chão enquanto a câmara grava.

Quantitativamente, a avaliação será realizada por meio de métricas de desempenho, como *precision*, *recall* e *mean Average Precision (mAP)*, que medirão a precisão e a abrangência das detecções. Também serão analisados o número total de objetos detetados, a distribuição das classes previstas e a exatidão da localização das caixas delimitadoras em relação aos objetos reais. A análise de tempo de inferência por quadro será um fator essencial para validar a viabilidade da aplicação em tempo real, garantindo que o modelo opere com baixa latência e alta eficiência. Estes resultados permitirão comparar o desempenho do modelo com abordagens existentes e identificar possíveis melhorias.

Milestone 1

1.Data Pipeline e Pré-processamento

O presente trabalho tem como objetivo desenvolver um sistema de classificação automática de resíduos urbanos com base em visão por computador, utilizando o modelo YOLOv8 para detecção e categorização.

Começamos por utilizar o TACO (*Trash Annotations in Context*), que contém imagens anotadas de lixo em ambientes reais e segue o formato COCO. Este *dataset* apresenta um número elevado de classes específicas (aproximadamente 60), o que aumenta a complexidade do modelo.

Para reduzir a complexidade associada às múltiplas classes originais do *dataset* TACO, foi aplicado um processo de pré-processamento e transformação das anotações. As categorias detalhadas foram agrupadas em cinco superclasses: plástico, papel, metal, vidro e outro. Este mapeamento foi realizado através da análise semântica dos nomes das categorias, com base em palavras associadas ao tipo de material. Por exemplo, categorias como "*plastic*", "*styrofoam*" e "*wrapper*" foram associadas à classe "plástico"; "*can*" e "*foil*" à classe "metal"; "*paper*", "*cardboard*", "*carton*", "*newspaper*", "*magazine*" e "*carded*" à classe "papel"; e "*glass*", "*bottle*", "*jar*", "*window*" e "*mirror*" à classe "vidro". As restantes categorias foram classificadas como "outro".

Este processo foi implementado através da função `merge_classes_to_superclasses`, que percorre as anotações no formato COCO e substitui os identificadores das categorias originais pelos correspondentes às cinco superclasses. A função também renomeia os ficheiros de imagem, substituindo eventuais subdiretórios no nome do ficheiro por um formato plano, para garantir consistência nos caminhos de acesso. Em seguida, copia esses ficheiros para um novo diretório organizado, preservando as imagens relevantes para treino. Por fim, cria um ficheiro JSON contendo as anotações

modificadas, a nova lista de categorias (limitada às superclasses), e os metadados originais.

Em seguida, o *dataset* foi convertido para o formato YOLO, utilizando a biblioteca *pylabel*, que tratou da transformação das anotações COCO para o formato utilizado por modelos YOLOv8. A conversão resultou em dois diretórios principais: um com as imagens e outro com os ficheiros *.txt* contendo as anotações, organizados em subconjuntos de treino e validação, segundo uma divisão de 80% e 20% (*train_pct=0.8*, *val_pct=0*, *test_pct=0.2*), respetivamente. Também foi criado um ficheiro *dataset.yaml* com a definição das classes e os caminhos para os dados.

2. Baseline

O modelo escolhido foi o YOLOv8 nano (*yolov8n*), que possui uma profundidade reduzida e cerca de 3,2 milhões de parâmetros. O modelo foi treinado com imagens redimensionadas para uma resolução de entrada de 640×640 pixels, que é o valor padrão utilizado pelo YOLOv8. O otimizador utilizado foi o SGD (*Stochastic Gradient Descent*), que é o otimizador padrão no YOLOv8. Foi utilizado um modelo pré-treinado, mantendo as primeiras dez camadas congeladas durante as fases iniciais de treino (*freeze=10*).

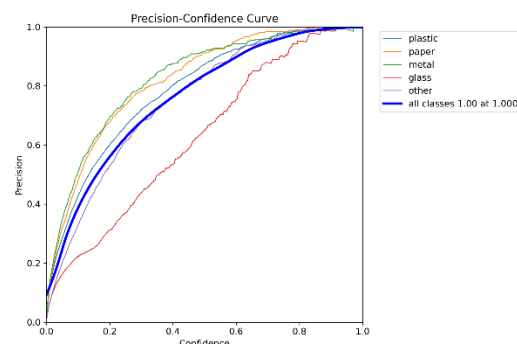
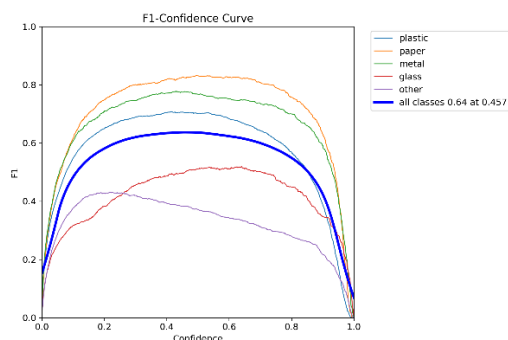
O processo de treinamento foi dividido em três fases, num total de 100 épocas: 20 épocas com uma taxa de aprendizagem de 1e-3, 40 épocas com 5e-4 e 40 épocas com 1e-4.

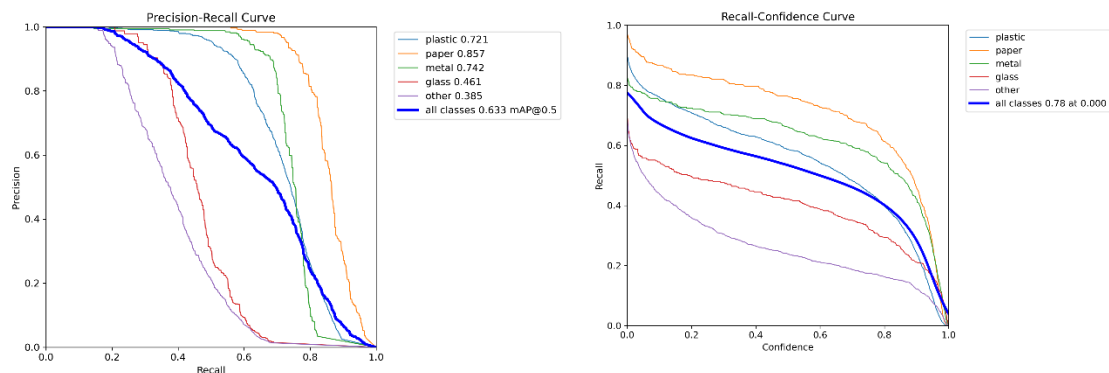
Durante o treinamento, foram aplicadas transformações de *data augmentation* incluídas na *framework* YOLOv8, como modificações de matiz, saturação e brilho (HSV) com valores de *hsv* (*hsv_h=0.015*, *hsv_s=0.7*, *hsv_v=0.4*), além de espelhamento horizontal e vertical (*flipud=0.5*, *fliplr=0.5*). Esses valores correspondem aos padrões (*default*) da função de treinamento utilizada.

Este pipeline define a configuração base do projeto e servirá como ponto de partida para futuras otimizações, incluindo ajustes de arquitetura, hiperparâmetros e estratégias de aumento de dados.

O treinamento completo do modelo teve uma duração aproximada de 20 horas com um processador 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz, utilizando um computador com recursos limitados. Para comparação, o mesmo processo levou mais de 3 horas quando executado no Google Colab, aproveitando o T4 GPU.

3. Preliminary results and training curves:





Observa-se uma variação no desempenho entre as classes, com 'papel' e 'metal' a apresentarem métricas geralmente mais elevadas em comparação com classes como 'vidro' e 'outros'. A classe 'plástico', apesar de ter o maior número de instâncias no conjunto de dados, não demonstra necessariamente o desempenho mais alto em todas as métricas, o que sugere que a quantidade de dados por si só não garante o melhor desempenho. As curvas de precisão-*recall* e F1-confiança revelam o compromisso inerente entre a precisão e a capacidade do modelo de detetar todos os objetos relevantes (*recall*), influenciado pelo limiar de confiança utilizado para as previsões. A análise da distribuição das caixas delimitadoras indica uma tendência para os objetos estarem localizados centralmente nas imagens e apresentarem uma variedade de tamanhos e proporções.

- **Curva F1-Confiança:** O pico da curva para cada classe indica o limiar de confiança ideal para obter o melhor equilíbrio entre precisão e *recall* para essa classe específica. A curva geral (azul) mostra que um limiar de confiança em torno de 0.457 parece oferecer um bom compromisso entre precisão e *recall* para todas as classes combinadas, resultando num F1-score máximo de 0.64.
- **Curva Precisão-Confiança:** À medida que o limiar de confiança aumenta, a precisão tende a melhorar para todas as classes, sugerindo que quando o modelo está mais confiante nas suas previsões, estas são mais propensas a serem corretas. No entanto, essa maior precisão geralmente vem com um custo em termos de *recall* (observado na curva *Recall-Confiança*).
- **Curva Precisão-Recall:** As áreas sob as curvas (AP) para cada classe fornecem uma medida do desempenho geral da classe, independentemente do limiar de confiança. Classes como 'paper' e 'metal' têm áreas maiores, indicando um melhor equilíbrio entre precisão e *recall* em diferentes limiares, enquanto 'glass' e 'other' apresentam um desempenho inferior. A mAP de 0.633 sugere um desempenho médio razoável, mas com espaço para melhorias em algumas classes.
- **Curva Recall-Confiança:** À medida que o limiar de confiança aumenta, o *recall* tende a diminuir para todas as classes. Isso indica que, ao tornar o modelo mais seletivo (aumentando a confiança necessária para uma deteção), ele perde mais objetos verdadeiros. A curva mostra o ponto em que o *recall* começa a cair significativamente para cada classe em função do aumento da confiança.

4. Próximos Passos:

Com base nas matrizes de confusão e na análise do desempenho do modelo, foram identificadas áreas específicas onde a classificação pode ser melhorada, especialmente na distinção entre objetos e o fundo da imagem. Verifica-se que há uma confusão considerável entre as classes 'vidro', 'plástico' e 'outros' com o 'background', o que indica que o modelo tem dificuldades em distinguir corretamente objetos dessas classes em determinados contextos. As classes 'vidro' e 'outros', em particular, apresentam baixas taxas de precisão, sendo frequentemente confundidas com o fundo.

Dado este cenário, os próximos passos devem concentrar-se em melhorar a capacidade do modelo para diferenciar corretamente objetos das diferentes classes, reduzir os falsos positivos (especialmente em objetos pequenos ou inexistentes) e equilibrar o desempenho entre as categorias:

- **Refinar a distinção entre 'vidro' e 'background'**, analisando exemplos problemáticos e reforçando o treino com dados adicionais ou aumentados, que destaquem situações visuais desafiantes, como variações de iluminação ou oclusões parciais.
- **Melhorar a detecção da classe 'outros'**, que aparenta ser demasiado heterogênea.
- **Reduzir falsos positivos de objetos muito pequenos**, através de filtros de pós-processamento que eliminem detecções abaixo de um determinado tamanho mínimo. Adicionalmente, ajustar a sensibilidade do modelo a pequenas escalas pode ajudar a mitigar este tipo de erro.
- **Reforçar o treino com exemplos negativos (background)**, incluindo imagens ou regiões claramente sem objetos, para ensinar o modelo a evitar ativações incorretas, sobretudo em áreas pequenas ou ruidosas.
- **Equilibrar o desempenho entre 'plástico' e 'background'**, reforçando o treino com amostras onde a diferença entre as duas classes seja clara, e aplicando aumentos de dados que variem as condições visuais do plástico.

Milestone 2

Seguindo as recomendações para otimizações iniciais no treino com YOLO, aplicámos com sucesso a técnica de *Mosaic & Copy-Paste augmentation*, mantendo os valores por defeito da *Ultralytics*, mas ajustando os parâmetros `--degrees 10` e `--scale 0.5` para ampliar a variação dos dados sintéticos. Isto contribuiu para melhorar a visibilidade de fragmentos de vidro e potenciar amostras raras. Além disso, ativámos o modo `--rect`, permitindo um redimensionamento. A retreinação automática dos anchors não foi possível realizar devido à versão do YOLO utilizada. No que diz respeito à verificação de rótulos (*Label-sanity script*), gerámos 200 *bounding boxes* aleatórias, mas inspecionámos manualmente apenas cerca de 10, sem detetar erros evidentes nas classes após a fusão por superclasses. As *class-imbalance weights* não foram aplicadas, uma vez que a versão do YOLO, também não suportava a passagem direta do parâmetro `--cls_weights`.

1. Métodos de treino:

No processo de treino do modelo YOLOv8-s, configurou-se o para realizar 40 épocas com uma taxa de aprendizagem inicial de $5e-4$. Foi aplicada a técnica de congelamento (`freeze=1`) das primeiras camadas da rede, preservando os pesos pré-treinados dessas

camadas para acelerar a convergência e evitar a degradação do conhecimento já aprendido, principalmente no início do treinamento. O treino foi realizado com imagens redimensionadas para 640x640 pixels (`imgsz=640`) e batch size de 16, utilizando o modo de retângulo (`rect=True`) para otimizar o processamento. Para aumentar a robustez do modelo, foram aplicadas diversas técnicas de augmentação de dados, incluindo variações controladas de matiz, saturação e brilho (`hsv_h=0.015`, `hsv_s=0.7`, `hsv_v=0.4`), flips horizontais e verticais com probabilidade de 50%, rotações de até 10 graus e escalas de até 50%.

Quanto à função de perda e ao otimizador utilizados no treinamento, a biblioteca *Ultralytics YOLO* adota configurações padrão na chamada de `model.train()`. O modelo utiliza uma função de perda composta própria do YOLO, que combina erros de localização, classificação e confiança. O otimizador padrão é o *Stochastic Gradient Descent* com momentum, configurado internamente pela biblioteca para garantir um bom desempenho e estabilidade no treinamento. Além disso, no seu caso, foi utilizado o agendamento cossenoidal da taxa de aprendizagem, que reduz a *learning rate* de forma suave ao longo das épocas, ajudando o modelo a alcançar uma convergência mais estável e refinada.

2. Experiências:

A experiência A teve como objetivo principal melhorar o desempenho do modelo YOLOv8-s na detecção de resíduos difíceis, especialmente nas classes *glass* e *other*, que representam objetos pequenos ou visualmente complexos. A hipótese era que o uso de *feature maps* maiores, alcançados através de uma maior resolução de entrada (768×768), em conjunto com *batch size* de 8 e AMP, levaria a um ganho de 4 a 6 pontos percentuais no mAP das classes *glass* e *other*. No entanto, os resultados não confirmaram essa expectativa.

A comparação direta com o modelo padrão, que utilizou uma resolução menor, mostrou que apesar da classe *other* realmente ter apresentado uma melhoria modesta, a classe *glass* teve uma queda significativa no desempenho, tanto em precisão como em mAP. Além disso, o desempenho global do modelo com a configuração da experiência A foi inferior ao do padrão, sugerindo que o aumento da resolução não trouxe os benefícios esperados e, em vez disso, pode ter prejudicado a capacidade de generalização do modelo.

Tabela 1-Resultados do padrão

Classe	Imagens	Instâncias	Precisão (P)	Revocação (R)	mAP@50	mAP@50-95
all	1209	3842	0.778	0.510	0.571	0.454
plastic	849	1625	0.853	0.604	0.720	0.568
paper	258	359	0.898	0.781	0.859	0.737
metal	224	452	0.904	0.560	0.630	0.522
glass	78	226	0.419	0.354	0.257	0.164
other	477	1180	0.819	0.253	0.392	0.280

Tabela 2- Resultados da experiência A

Classe	Imagens	Instâncias	Precision (P)	Recall (R)	mAP@50	mAP@50-95
all	1209	3842	0.477	0.378	0.379	0.246
plastic	849	1625	0.577	0.546	0.542	0.339
paper	258	359	0.586	0.552	0.567	0.393
metal	224	452	0.558	0.482	0.504	0.337
glass	78	226	0.267	0.133	0.080	0.0406
other	477	1180	0.400	0.175	0.204	0.120

Na experiência B, que exigia ajustes na função de perda (`--fl_gamma` e `--cls_weights`), não pôde ser implementado devido a incompatibilidades com a versão do YOLOv8 que estamos a utilizar, a qual não suporta diretamente estas opções.

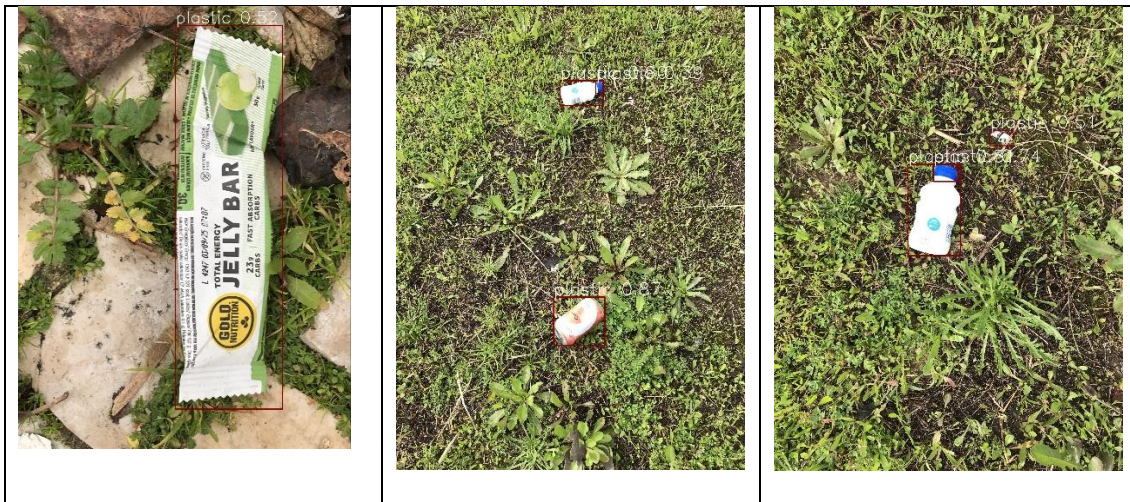
Na experiência C ainda não foi concluído até à data de entrega da milestone. Isto deve-se ao elevado tempo de execução em CPU no Google Colab, aliado a interrupções frequentes quando se utilizavam recursos mais rápidos como GPU e TPU, que terminavam a sessão antes da conclusão do treino. A solução sugerida pelo professor, que poderia ter mitigado estas limitações, infelizmente chegou de forma tardia, o que atrasou ainda mais o processo.

Relativamente a experiência D, que envolvia o *fine-tuning* com YOLOv8-seg-s, este foi descartado por estar dependente duma versão segmentadora do YOLO que não está em uso no nosso pipeline atual.

A experiência E não foi realizado porque o *dataset* pLitterStreet foi especificamente concebido para a deteção de resíduos plásticos em ambientes urbanos, como sacos, garrafas, copos, máscaras, palhinhas e outros objetos similares. Dada esta natureza altamente especializada, o conjunto de dados não inclui imagens nem anotações de resíduos de vidro ou invólucros brilhantes de outros materiais. Assim, treinar o modelo com este *dataset* não contribuiria para o objetivo central da experiência E, que era aumentar o *recall* para objetos com aparência brilhante, como vidro e embalagens reflexivas.

Na experiência F, aplicamos um filtro de tamanho no pós-processamento das deteções feitas pelo modelo YOLOv8. O *framework* Ultralytics não possui, por padrão, um filtro automático para eliminar caixas de deteção muito pequenas diretamente na inferência. Para resolver isso, foi implementada uma função que, após a inferência, elimina todas as caixas cuja área seja menor que um valor mínimo definido (*threshold*), de modo a remover falsos positivos muito pequenos que prejudicam a precisão e manter apenas as deteções relevantes, melhorando a qualidade dos resultados sem afetar o *recall*.

Tabela 3- Inferência do Modelo



A inferência do modelo apresenta um desempenho quase perfeito nas fotos capturadas diretamente do chão. Em geral, o sistema identifica corretamente os resíduos, embora ocasionalmente haja detecções incorretas de plástico em objetos que não pertencem a essa categoria. Foi também observado que o modelo consegue detectar com maior precisão o lixo quando as imagens são tiradas de um ângulo mais distante. O tempo médio de detecção por imagem é rápido, aproximadamente 3 milissegundos, o que demonstra a eficiência do sistema.

3.Resultados:

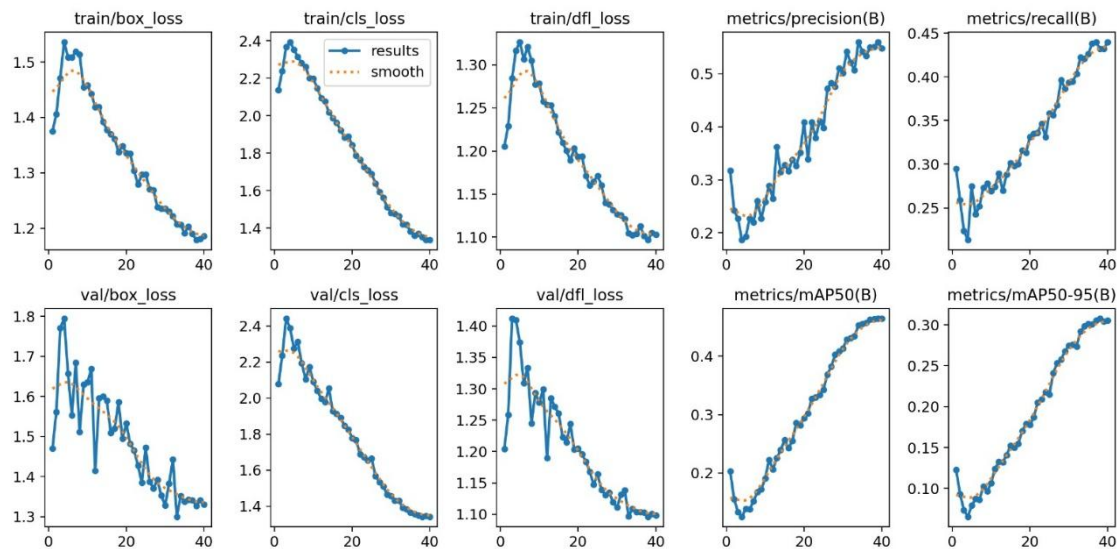


Figura 1- Learning Curves

O modelo foi treinado por 40 épocas e os resultados de desempenho foram monitorados por meio das perdas de treinamento e validação (*box loss*, *classification loss*, *distribution*

focal loss), além das métricas de avaliação: precisão (*precision*), revocação (*recall*) e *mAP* nas variantes *mAP50* e *mAP50-95*.

As curvas de perda demonstram uma tendência de convergência estável, com as perdas de *train/box_loss*, *train/cls_loss* e *train/dfl_loss* diminuindo consistentemente ao longo das épocas. As perdas de validação apresentaram maior variação, mas também evidenciaram uma tendência de queda, sugerindo que o modelo não sofreu *overfitting* significativo durante o treinamento.

No que diz respeito às métricas de desempenho:

- A precisão aumentou de forma contínua, atingindo aproximadamente 0.54 ao final do treinamento.
- O *recall* também evoluiu positivamente, alcançando 0.45.
- O *mAP@50*, métrica padrão na detecção de objetos, cresceu de 0.20 na primeira época para cerca de 0.55, indicando uma melhora significativa na qualidade das detecções.
- O *mAP@50-95*, que considera uma gama mais ampla de IOUs, subiu de 0.12 para aproximadamente 0.30.

Estes resultados indicam que o modelo aprendeu de forma eficaz, com ganhos contínuos ao longo do treinamento, culminando em métricas que demonstram um bom equilíbrio entre precisão e generalização.

Referências

- Ceunen, B. (2021). *Garbage Detection with TensorFlow*. Obtido de kaggle: <https://www.kaggle.com/code/bouweceunen/garbage-detection-with-tensorflow>
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. (2016). SSD: Single Shot MultiBox Detector. Em *Computer Vision – ECCV 2016* (pp. 21–37). Springer International Publishing. doi:10.1007/978-3-319-46448-0_2
- Mandhati, S. R., Deshapriya, N. L., Mendis, C. L., Gunasekara, K., Yrle, F., Chaksan, A., & Sanjeev, S. (2024). pLitterStreet: Street Level Plastic Litter Detection and Mapping. Obtido de <https://arxiv.org/abs/2401.14719>
- Melinte, D., Travediu, A.-M., & Dumitriu, D. (2020). Deep Convolutional Neural Networks Object Detector for Real-Time Waste Identification. *Applied Sciences*, 10(20), 7301. doi:<https://doi.org/10.3390/app10207301>
- Oliveira, M., & Guedes, E. (2024). Detecção de Lixo em Áreas Costeiras: Uma Aplicação de Segmentação com R-CNNs da Família YOLO. *Anais do Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais (WCAMA)*, 11-20. doi:10.5753/wcama.2024.1902
- Proença, P., & Simões, P. (2020). TACO: Trash Annotations in Context for Litter Detection. *arXiv*.

- Valente, M., Coelho, A., & Assis, R. (2024). Sistema de detecção de resíduos em rios e nascentes utilizando visão computacional e inteligência artificial. *Revista de Gestão e Secretariado*, 15(11), e4497.
doi:<https://doi.org/10.7769/gesec.v15i11.4497>
- Zhang, A., Lipton, Z., Li, M., & Smola, A. (2023). *Dive into Deep Learning*. Cambridge University Press.