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 autónomos 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 detecção 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 deteção 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.Algortimo utilizado

Para demonstrar a capacidade de inferência a partir de imagens não vistas, serão implementados modelos pré-treinados para deteção de resíduos. Dentre as opções disponíveis, utilizaremos o *Single Shot MultiBox Detector* (Liu, et al., 2016), um modelo otimizado para deteção 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 deteção 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 deteção 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 deteções. 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 deteção 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.Basiline

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 pixéis, que é o valor padrão utilizado pelo YOLOv8. O optimizador utilizado foi o SGD (*Stochastic Gradient Descent*), que é o optimizador padrão no YOLOv8. Foi utilizado um modelo pré-treinado, mantendo as primeiras dez camadas congeladas durante as fases iniciais de treino (`freeze=10`).

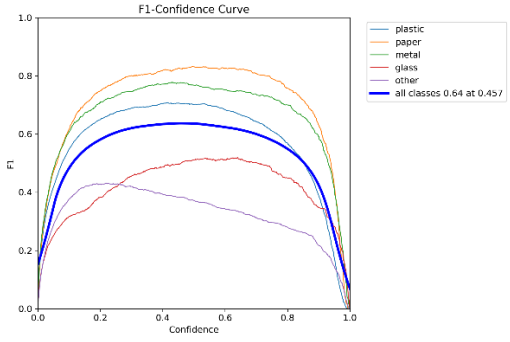
O processo de treinamento foi dividido em três fases, num total de 100 épocas: 20 épocas com uma taxa de aprendizagem de 1e-3, 40 épocas com 5e-4 e 40 épocas com 1e-4.

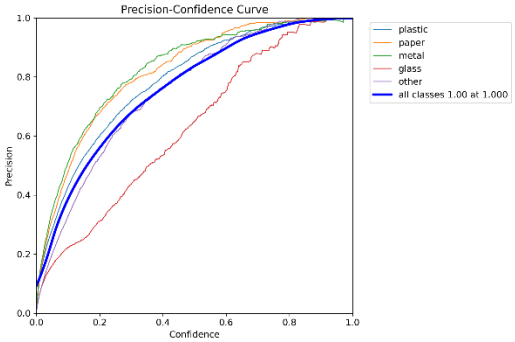
Durante o treinamento, foram aplicadas transformações de *data augmentation* incluídas na *framework* YOLOv8, como modificações de matiz, saturação e brilho (HSV) com valores de hsv (`hsv_h=0.015, hsv_s=0.7, hsv_v=0.4`), além de espelhamento horizontal e vertical (`flipud=0.5, fliplr=0.5`). Esses valores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

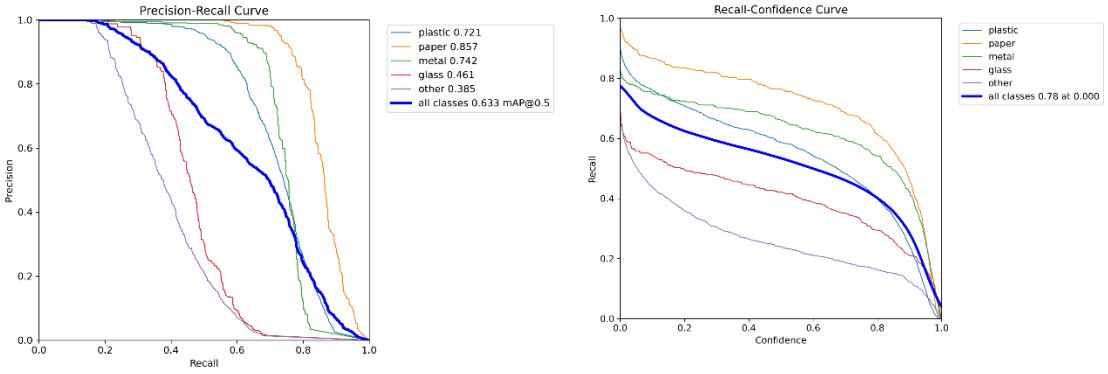
Este pipeline define a configuração base do projeto e servirá como ponto de partida para futuras otimizações, incluindo ajustes de arquitetura, hiperparâmetros e estratégias de aumento de dados.

O treinamento completo do modelo teve uma duração aproximada de 20 horas com um processedor 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 deteção da classe 'outros'**, que aparenta ser demasiado heterogénea.
- **Reduzir falsos positivos de objetos muito pequenos**, através de filtros de pós-processamento que eliminem deteções 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.

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.