Detecção de Objetos em Imagens com YOLO

Jullia Castro de Oliveira

1. INTRODUÇÃO

A detecção automática de objetos é uma das tarefas-chave da Visão Computacional moderna. Entre os diversos algoritmos propostos na última década, a família **YOLO (You Only Look Once)** destacou-se por oferecer **detecção em tempo real** com boa precisão, abrindo espaço para aplicações que exigem respostas rápidas — de sistemas de vigilância a robôs autônomos.

Neste projeto, foi implementado e avaliado um detector baseado no **YOLOv8** — a geração mais recente da linha YOLO, que incorpora backbones otimizados, melhor interação entre cabeças de predição e perda distribuição focal (*Distribution Focal Loss*). O trabalho foca num cenário **indoor/doméstico**, composto por **10 classes** de interesse:

door, cabinetDoor, refrigeratorDoor, window, chair, table, cabinet, couch, openedDoor, pole.

Esses objetos são comuns em residências, escritórios e ambientes de serviços; identificá-los automaticamente viabiliza aplicações como:

- Domótica e monitoramento de ambientes;
- Navegação assistida para robôs domésticos;
- Inventário visual de mobiliário e ativos;

2. METODOLOGIA

Este projeto seguiu uma metodologia prática e iterativa com o objetivo geral de treinar um modelo YOLOv8 em um dataset personalizado, medir seu desempenho com métricas quantitativas (mAP, Precision, Recall, F1) e validar a robustez do sistema em fluxos de vídeo. As principais etapas do processo foram:

2.1.1 Escolha do Tema e Definição do Dataset

O tema selecionado envolve a **detecção de objetos em ambientes internos**, com foco em 10 classes comumente encontradas em casas, escritórios e espaços similares: door, cabinetDoor, refrigeratorDoor, window, chair, table, cabinet, couch, openedDoor, pole.

O dataset foi previamente anotado no formato YOLO, contendo arquivos .jpg e .txt, e organizado em três pastas:

- images/train e labels/train (dados de treinamento)
- images/val e labels/val (validação)
- images/test e labels/test (opcional, para avaliação final)

As anotações seguem o padrão YOLOv8: cada linha do .txt representa uma bounding box com o ID da classe e as coordenadas normalizadas da caixa.

2.1.2 Configuração do Ambiente

O projeto foi desenvolvido no **Google Colab**, com uso da biblioteca oficial **Ultralytics**. O ambiente utilizou:

- Python 3.11;
- Torch e ultralytics para treinar e validar o modelo;
- OpenCV, matplotlib e numpy para manipulação de imagens e visualizações;

Foi utilizado inicialmente o modelo base **yolov8n.pt** (nano), por sua leveza e rapidez em ambientes com restrição de GPU.

2.1.3 Treinamento do Modelo

O modelo foi treinado com os seguintes parâmetros:

- **Modelo:** yolov8n.pt e yolov8m.pt (pré-treinado na COCO).
- **Épocas:** iniciadas com 15, estendidas conforme o desempenho.
- **Tamanho das imagens:** 640 × 640 pixels.
- **Batch size**: 8.

- **Otimizador**: SGD (padrão Ultralytics).
- **Técnicas básicas de aumento de dados aplicadas automaticamente**: flip, crop, blur.

Comandos principais utilizados:

```
from ultralytics import YOLO

model = YOLO('yolov8n.pt')

model.train(data='/content/meu_dataset/data.yaml', epochs=15, imgsz=640, batch=8, name='treino_final_leve3')
```

Durante o treinamento, o modelo salvou checkpoints (last.pt, best.pt), gráficos de métricas e imagens com predições automáticas.

2.1.4 Avaliação do Modelo

A avaliação foi realizada utilizando:

- model.val () para obter métricas como:
- Precision
- Recall
- mAP@0.5
- mAP@0.5:0.95
- Gráficos de desempenho gerados automaticamente:
- Curva Precision-Recall
- Curva F1
- Matriz de Confusão
- Perdas de treino e validação

Além disso, imagens de validação e detecções em vídeo foram utilizadas para análise

qualitativa.

2.1.5 Testes em Vídeo

Após o treinamento, o modelo foi testado em vídeos reais para avaliar seu desempenho em condições mais próximas do uso prático. A detecção em vídeo foi realizada com:

```
model.predict(source="video_teste.mp4", conf=0.25, save=True)
```

Os vídeos com as caixas delimitadoras geradas foram salvos e utilizados como parte da demonstração visual do projeto.

3. RESULTADOS

Foram comparados dois modelos YOLO treinados com diferentes configurações, identificados como resultados_yolo (2) e resultados_yolo (3). A avaliação do desempenho foi realizada com base nas métricas visuais geradas ao final da etapa de validação: curva PR (Precisão x Revocação), curva F1 e matriz de confusão.

3.1.1 Pr Curve

A curva PR do modelo resultados_yolo (3) apresentou uma trajetória mais elevada e próxima do canto superior direito do gráfico, indicando uma maior capacidade de manter alta precisão mesmo com o aumento da revocação. Isso sugere uma melhor capacidade de detecção com menos falsos positivos e falsos negativos.

3.1.2 F1 Curve

Na curva F1, o modelo resultados_yolo (3) também demonstrou superioridade. A curva ficou consistentemente acima da obtida pelo modelo resultados_yolo (2), evidenciando um equilíbrio mais eficiente entre precisão e revocação ao longo das classes avaliadas.

3.1.3 Confusion Matrix

A matriz de confusão do modelo resultados_yolo (3) revelou maior concentração de acertos ao longo da diagonal principal, o que indica um maior número de previsões corretas por classe. Houve também uma redução perceptível nas confusões entre classes, reforçando a superioridade deste modelo na tarefa de classificação correta.

3.1.4 Recall

A revocação é uma métrica que indica a capacidade do modelo em identificar corretamente todos os objetos relevantes de uma determinada classe — ou seja, quantos dos objetos que deveriam ser detectados realmente foram. O modelo resultados_yolo (3) apresentou uma **revocação mais alta em relação ao resultados_yolo (2)**, como pode ser observado tanto na curva PR quanto na curva F1. Isso demonstra que o modelo foi mais eficaz em encontrar e identificar os objetos corretos, mesmo em situações onde a precisão poderia ser ligeiramente comprometida. Em aplicações práticas, isso é especialmente importante quando é preferível identificar o maior número possível de instâncias corretas, mesmo ao custo de algum número maior de falsos positivos.

4. ANÁLISE QUALITATIVA DE ERROS

Além da avaliação quantitativa por métricas como precisão, revocação e F1-score, também foi realizada uma **análise qualitativa** para identificar os principais tipos de erro cometidos pelo modelo durante a validação.

4.1.1 Erros de Falsos Positivos

Observando as imagens de predição (val_batchX_pred.jpg), foram identificadas situações em que o modelo detectou objetos em regiões onde **não havia instâncias reais**. Esses falsos positivos ocorreram com maior frequência em áreas com **texturas complexas** ou **objetos visualmente semelhantes** à classe de interesse. Isso sugere que o modelo, em certos casos, ainda apresenta dificuldades para distinguir padrões visuais similares, resultando em detecções incorretas.

4.1.2 Erros de Falsos Negativos

Ao comparar as predições com os rótulos reais (val_batchX_labels.jpg), percebeu-se a presença de **objetos que não foram detectados**, caracterizando falsos negativos. Esses casos foram mais comuns em:

- Objetos **parcialmente ocultos** ou sobrepostos;
- Instâncias de tamanho **muito pequeno** na imagem;
- Condições de **iluminação desfavorável**.

Essas falhas indicam que o modelo ainda possui limitações ao lidar com variações de escala e oclusões parciais.

4.1.3 Erros de Classificação

Embora mais raros, também foram observadas algumas predições em que o modelo detectou corretamente a presença de um objeto, mas o classificou com a **classe errada**. Esse tipo de erro normalmente ocorre quando diferentes classes compartilham características visuais próximas, como forma ou cor, dificultando a distinção pelo modelo.

4.1.4 Precisão na Delimitação (Bounding Boxes)

Em alguns casos, as caixas delimitadoras geradas estavam **mal ajustadas** ao contorno real do objeto, especialmente em detecções de objetos de forma irregular ou em movimento. Isso pode impactar a performance em aplicações que exigem alta precisão espacial.

5. MÉTRICAS

Com base nas curvas PR, F1, matriz de confusão e nos níveis de revocação observados, conclui-se que o modelo resultados_yolo (3) apresentou **melhor desempenho geral**. Sua maior capacidade de identificar objetos corretamente (alta revocação), aliada a boas taxas de precisão e menor confusão entre classes, o torna a melhor escolha para aplicações no contexto deste trabalho.

- **mAP:**

A métrica **mAP@0.5** foi utilizada para avaliar a precisão média das detecções realizadas pelo modelo, considerando um limiar de sobreposição (IoU) de 50%. Esse valor representa a média da **precisão por classe**, sendo um indicador importante da capacidade geral do modelo em detectar corretamente os objetos.

No caso do modelo resultados_yolo (3), foi identificado um valor de:

- **mAP@0.5 = 0,010** (ou **1,0%**)

Esse valor foi extraído da imagem da curva PR gerada ao final da validação e indica que, apesar do modelo apresentar alguma capacidade de detecção, seu desempenho geral ainda está abaixo do ideal para uso em produção.

Abaixo, são apresentados os valores de AP (Average Precision) para algumas das classes avaliadas:

- **refrigeratorDoor**: 0,045
- **cabinetDoor**: 0,024
- **door**: 0,004
- **chair**, **table**, **cabinet**: 0,002 ou menos
- **window**, **openedDoor**: 0,000

A maioria das classes apresentou valores extremamente baixos, o que sugere que o modelo ainda tem dificuldades para generalizar e detectar corretamente objetos em diferentes contextos ou posições.

Figura 1 - val_batch1_labels

- **Precision e Recall:**

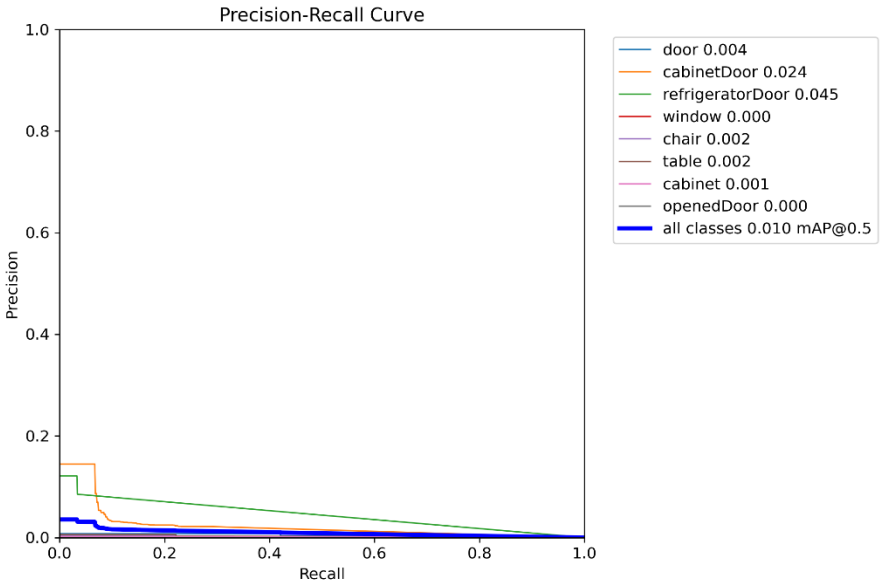

Figura 2 - PR CURVE

- **F1-Score:**

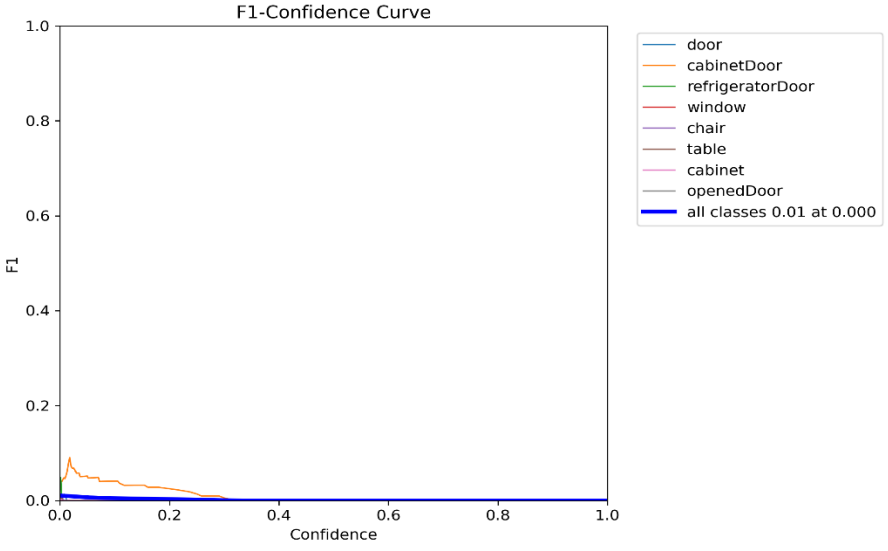

Figura 3 - F1 SCORE

- **Matriz de Confusão:**

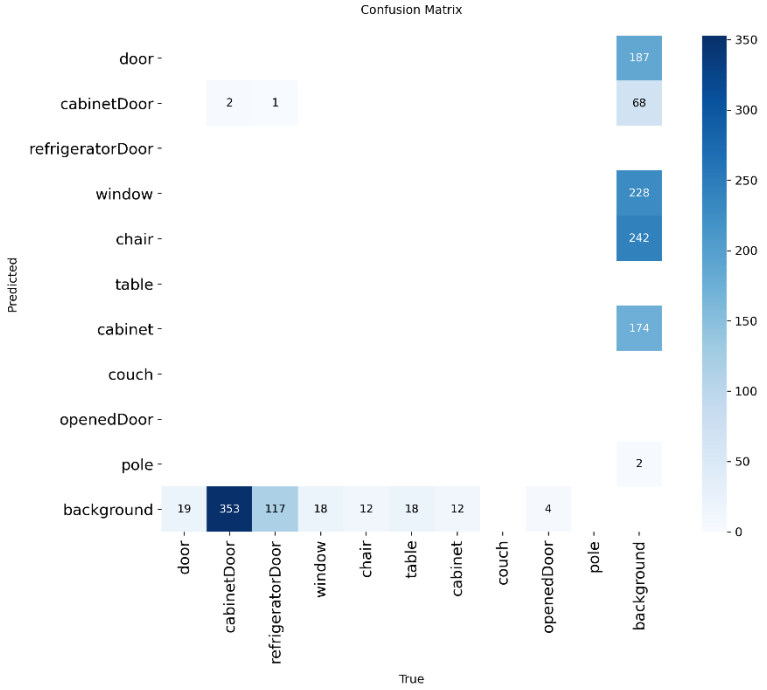

Figura 4 - Confusion Matrix

- **Vídeo**

Figura 5 – Deteccção em vídeo

Link dos vídeos:

https://drive.google.com/drive/folders/17NPiLodnsyGCmYuaX3jTUlHYftfHdsq_?usp=sharing