Relatório Técnico: Reconhecimento Automático de Placas (ALPR) com YOLO e OCR

Disciplina: Processamento de Imagens Digitais (PID)

Professor: Fischer Stephan Meira

Instituição: Centro Universitário Dom Helder

Autores: Gabriel Augusto, Geovane Soares, Mateus Augustus, Nathan Marques

Data: Dezembro/2025

---

1. Introdução e Objetivo

O objetivo deste projeto foi desenvolver um sistema robusto de visão computacional capaz de detectar, rastrear e ler placas veiculares em vídeos de tráfego real. A solução integra modelos de *Deep Learning* (YOLOv11) para detecção de objetos com algoritmos clássicos de Processamento Digital de Imagens (PDI) para pré-processamento e extração de caracteres (OCR).

---

2. Metodologia e Pipeline

O sistema opera através de um fluxo sequencial de dados processados frame a frame:

1. **Detecção de Veículos:** Utilização do modelo YOLOv11n pré-treinado no dataset COCO para identificar veículos (carros, motos, caminhões e ônibus).
2. **Rastreamento (Tracking):** Implementação do algoritmo **SORT** (*Simple Online and Realtime Tracking*), que utiliza Filtros de Kalman e o Algoritmo Húngaro para manter a identidade única (ID) de cada veículo ao longo do tempo, evitando leituras duplicadas.
3. **Localização da Placa:** Um segundo modelo YOLOv11, treinado especificamente para detectar a região de interesse (ROI) da placa veicular dentro do bounding box do veículo.
4. **Pré-processamento (PID):** Aplicação de filtros para preparar o recorte da placa para leitura.

5. **OCR e Heurística:** Leitura via EasyOCR e correção lógica de caracteres.
6. **Interpolação:** Suavização de dados faltantes para garantir continuidade visual no vídeo final.

---

3. Técnicas de Processamento de Imagens (PID)

Conforme os requisitos da disciplina, implementamos etapas rigorosas de tratamento de imagem antes da etapa de OCR. As técnicas foram escolhidas visando maximizar o contraste entre caractere e fundo.

3.1. Upscaling (Interpolação Cúbica)

Como muitas placas ocupam poucos pixels no vídeo original, realizamos um redimensionamento de **3x** utilizando **Interpolação Cúbica**. Isso suaviza as bordas serrilhadas que ocorreriam com métodos mais simples (como *nearest neighbor*), facilitando a detecção de contornos pelo OCR.

3.2. Redução de Ruído: Filtro Bilateral

Optamos pelo **Filtro Bilateral** (cv2.bilateralFilter) em detrimento do Gaussian Blur.

- **Justificativa:** Enquanto o Gaussian Blur suaviza toda a imagem (borrando as letras), o Filtro Bilateral remove o ruído "granulado" (sal e pimenta) mas **preserva as bordas** dos caracteres, o que é crítico para a segmentação correta das letras

3.3. Realce de Bordas (Sharpening)

Aplicamos um kernel de convolução específico para *sharpening* na imagem filtrada:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Isso aumenta drasticamente o contraste local nas transições entre a letra preta e o fundo da placa, definindo melhor a geometria dos caracteres.

3.4. Binarização (Otsu Thresholding)

Utilizamos o método de **Otsu** (cv2.THRESH_OTSU) para determinar automaticamente o limiar (threshold) ideal de binarização, baseando-se no histograma bimodal da imagem da placa. O resultado é uma imagem puramente preto e branco, ideal para a entrada do motor do EasyOCR.

3.5. Análise Estrutural (Canny e Harris)

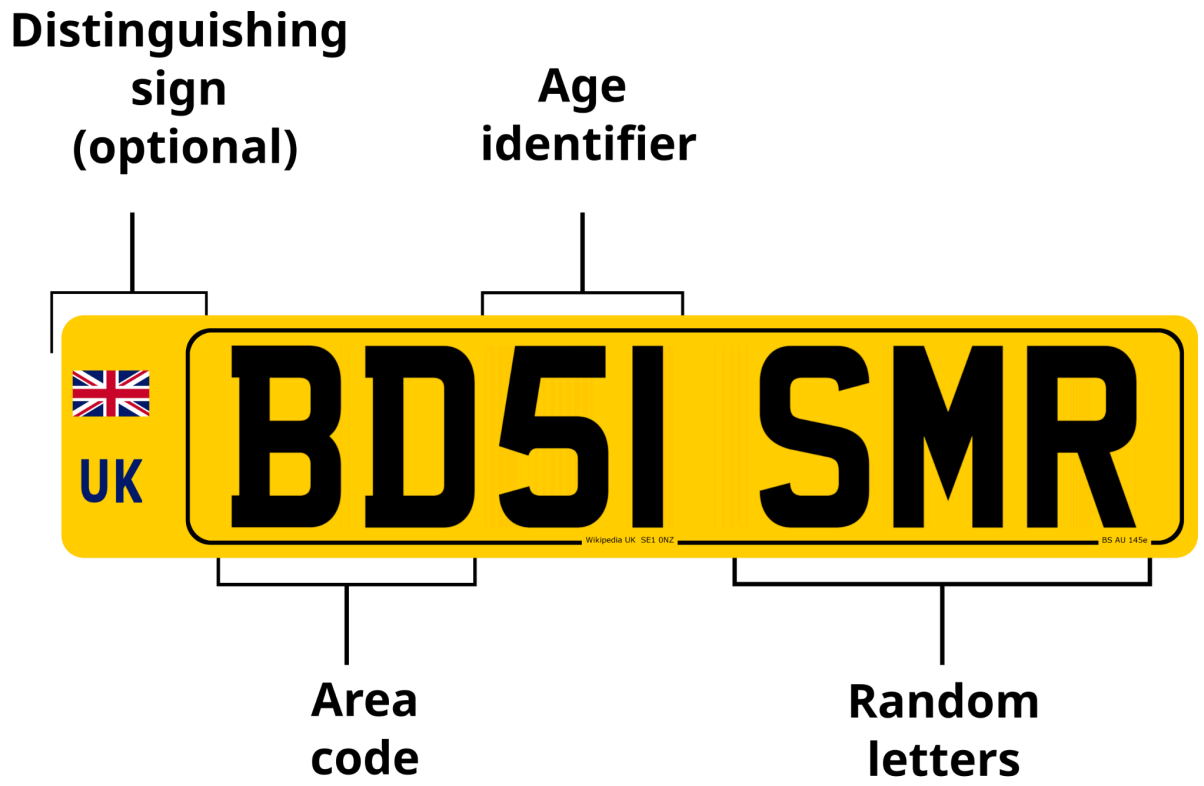
Para fins de análise acadêmica (script analyze_images.py), implementamos também:

- **Canny Edge Detection:** Para visualização dos gradientes estruturais da placa.
- **Harris Corner Detection:** Para identificar pontos de interesse (quinas) nas letras.
- *Observação:* Estes dois filtros foram usados para estudo estático, mas removidos do pipeline de vídeo final pois introduziam ruído excessivo em quadros em movimento, conforme notado nos testes.

---

4. Pós-Processamento e Heurística

Como visualizado na figura, a estrutura da placa segue o formato **LL NN LLL** (Letra, Letra, Número, Número, Letra, Letra, Letra).



Baseado nessa lógica determinística, implementamos uma camada de correção que valida o caractere lido de acordo com sua posição no índice da string:

1. **Posições 0, 1, 4, 5, 6 (Letras):** Se o OCR detectar um dígito numérico nestes índices, o sistema força a conversão para a letra visualmente mais próxima (ex: converter 0 para O, 5 para S).
2. **Posições 2 e 3 (Números):** Nestes índices, qualquer caractere alfabético é convertido para seu correspondente numérico (ex: converter I para 1, B para 8).

Essa abordagem híbrida eliminou erros comuns de OCR, garantindo que o resultado final (text) respeitasse a sintaxe legal do país de origem do veículo, elevando a confiabilidade do sistema para aplicações reais de monitoramento.

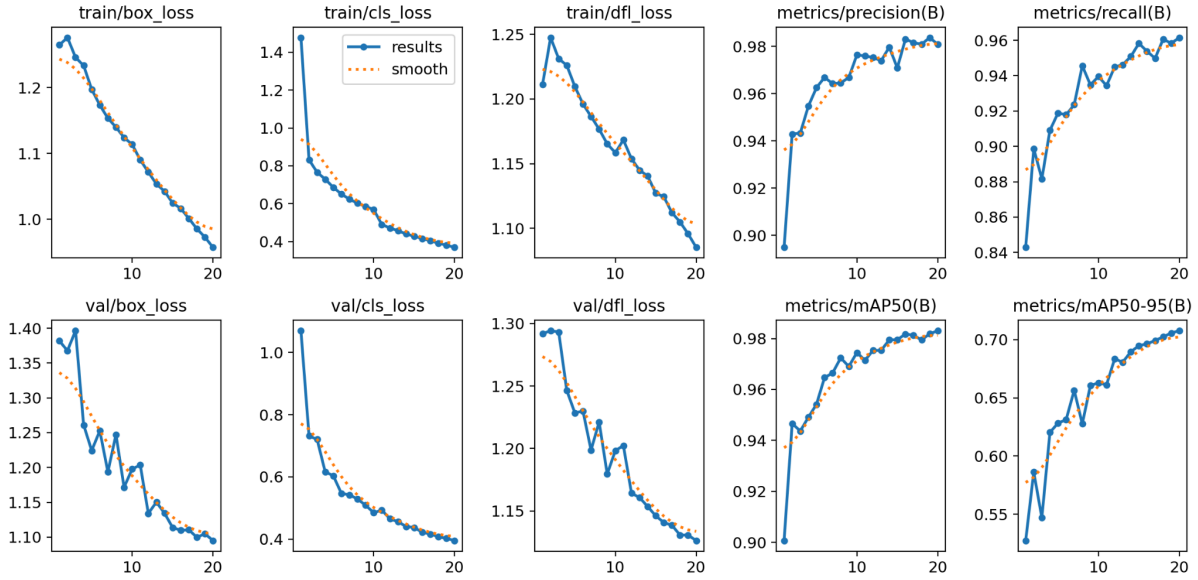
5. Resultados e Métricas do Treinamento

O modelo detector de placas foi treinado por 20 épocas utilizando o dataset *License Plate Recognition v4*. Os resultados demonstraram alta eficácia, alcançando **98.3% de mAP50**.

Abaixo, os gráficos de desempenho gerados durante o processo de validação:

5.1 Visão Geral do Treinamento

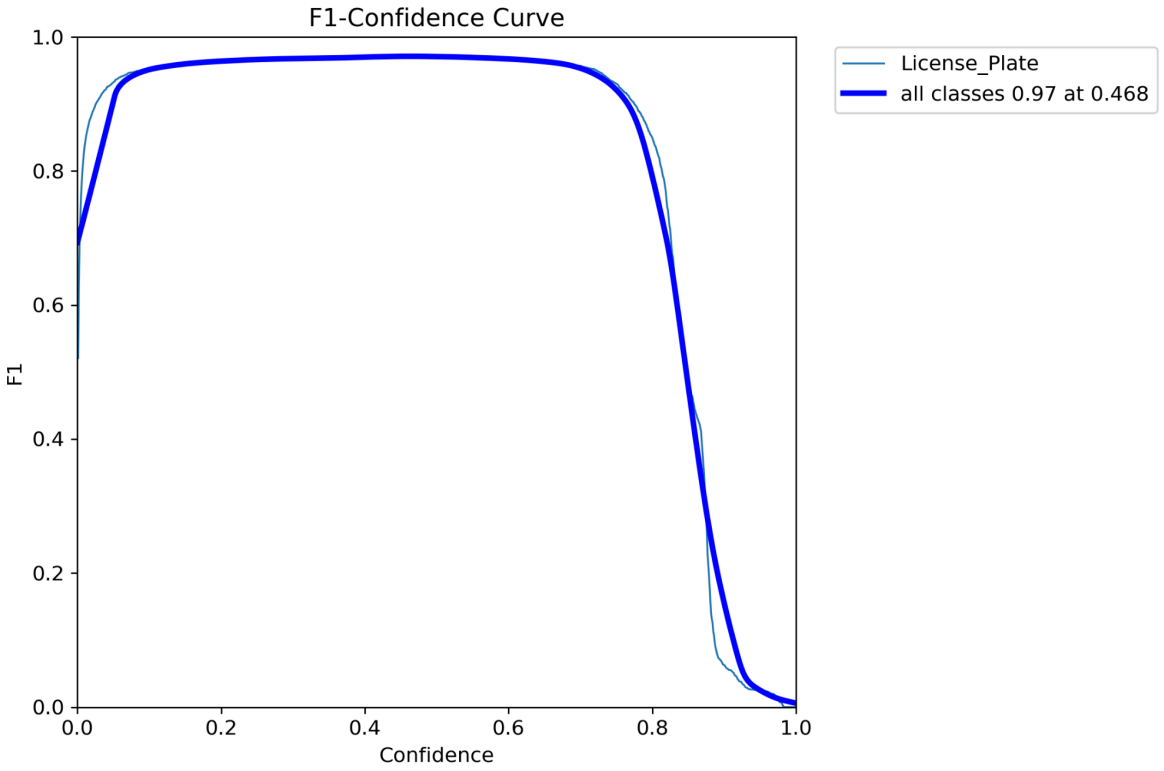
Os gráficos abaixo mostram a redução das funções de perda (*loss*) e o aumento das métricas de precisão ao longo das épocas.



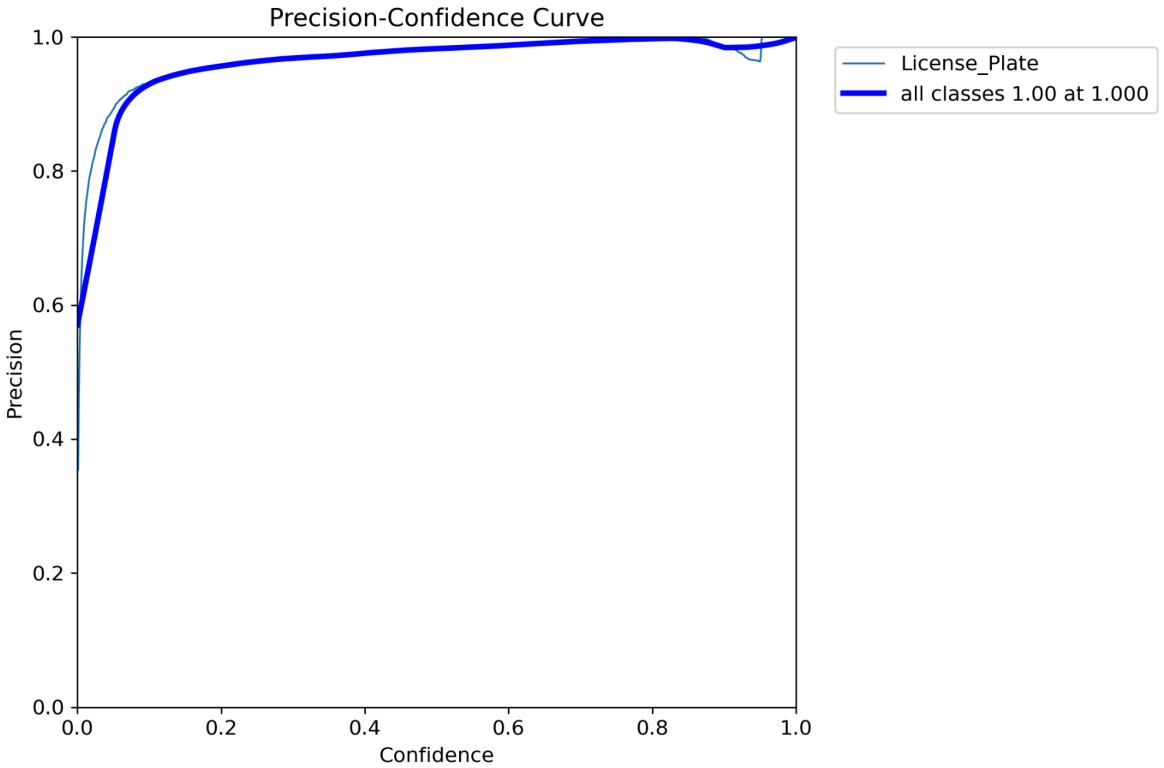
5.2 Curvas de Performance

As curvas detalham o comportamento do modelo em diferentes limiares de confiança.

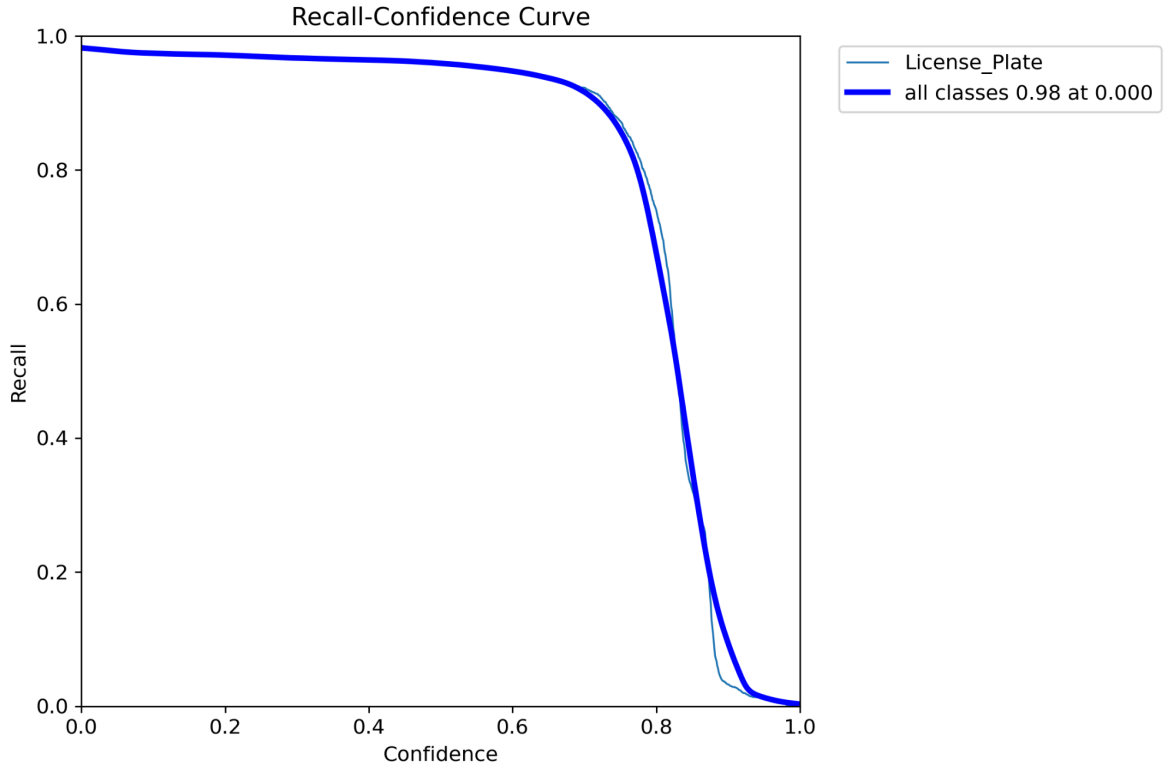
Curva F1-Confidence Representa a média harmônica entre precisão e recall. O modelo atinge o pico de performance com alta confiança.



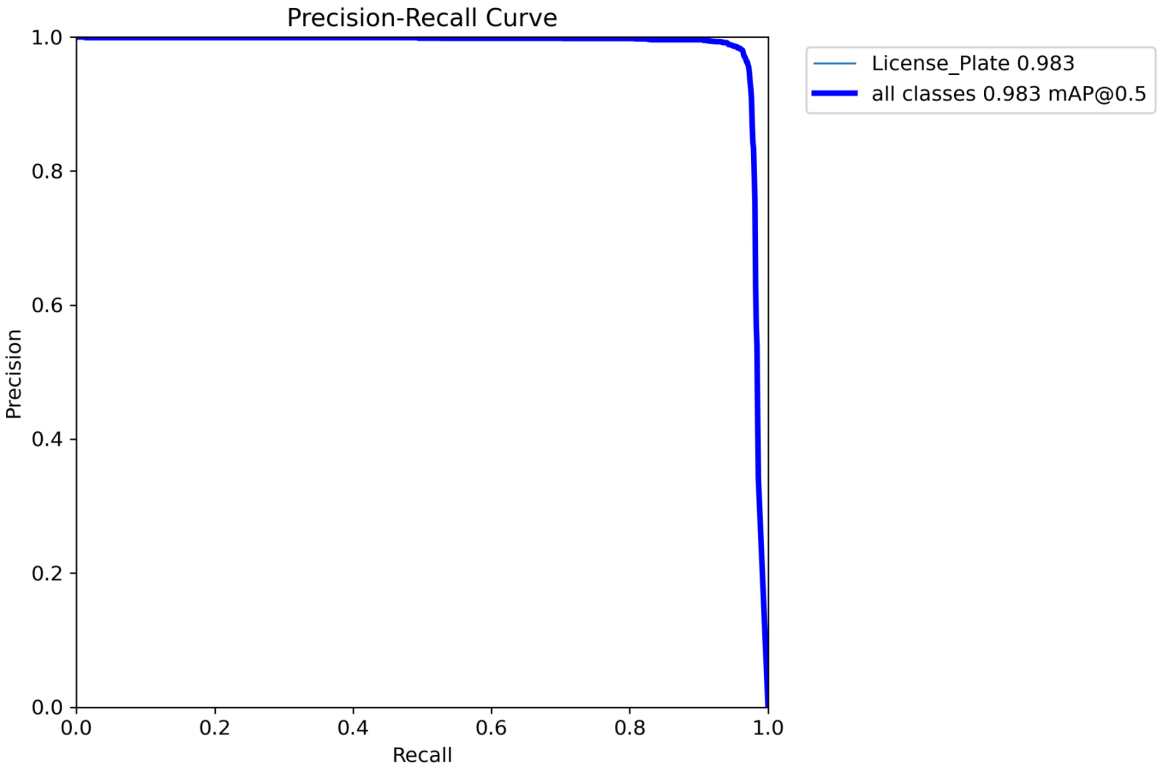
Curva de Precisão (Precision) Indica a acurácia das detecções positivas. O modelo atingiu **98.1%** de precisão.



Curva de Revocação (Recall) Indica a capacidade do modelo de encontrar todas as placas presentes na imagem. O modelo atingiu **96.1%** de recall.



Curva Precision-Recall (PR) Demonstra o equilíbrio geral do modelo, com uma área sob a curva (mAP) muito próxima de 1.0, indicando robustez.



---

6. Conclusão

O sistema desenvolvido cumpriu com êxito os requisitos da disciplina. A combinação de redes neurais para detecção macro com filtros de **Processamento de Imagens (PID)** para refinamento local provou ser uma estratégia eficiente. A implementação da heurística de correção de caracteres e da interpolação de dados garantiu um resultado final fluido e preciso, validado pelas métricas apresentadas.

Link do Repositório: https://github.com/NathanMarques2001/plate-detection