Reconhecimento de placa de veículos em imagens

Vitor Augusto de Souza Rego[1], César Alberto da Silva[2]

[1]Discente de Informática – Instituto Federal de Educação Ciência e Tecnologia de São Paulo, Campus Presidente Epitácio

[2]Docente de Informática – Instituto Federal de Educação Ciência e Tecnologia de São Paulo, Campus Presidente Epitácio

augusto.vitor@aluno.ifsp.edu.br, cesar@ifsp.edu.br

Abstract. *This study investigates various approaches for vehicle license plate recognition, including YOLO, Haar Cascade, and edge detection for plate identification, and Tesseract OCR and Easy OCR for character recognition. The methodology integrates convolutional and recurrent neural networks with image processing techniques. YOLO achieved over 98% accuracy in plate detection, while Tesseract OCR showed accuracy between 75% and 77% in character recognition. These techniques demonstrate high effectiveness in public security and restricted area automation applications.*

Resumo. *Este estudo explora o reconhecimento de placas de automóveis utilizando YOLO, Haar Cascade e detecção de bordas para a identificação, além de Tesseract OCR e Easy OCR para o reconhecimento dos caracteres. A metodologia combina redes neurais convolucionais e recorrentes com técnicas de processamento de imagem. YOLO obteve mais de 98% de precisão na detecção das placas, enquanto o Tesseract OCR apresentou assertividade entre 75% e 77% no reconhecimento dos caracteres, demonstrando a eficácia dessas técnicas na segurança pública e automação de entradas.*

1. Introdução

Com o avanço constante da tecnologia, os sistemas computacionais vem ganhando bastante destaque e utilização, a visão computacional é um exemplo. A Visão Computacional é uma área que vem crescendo e desempenhando um papel importante com aplicações em vários setores.

De acordo com Savekar e Kumar (2021), “a visão computacional se concentra no projeto de sistemas computacionais que possuem a capacidade de capturar, entender e interpretar informações visuais importantes contidas com dados de imagem e vídeo”.

O reconhecimento automático de placas veiculares, ou Automatic Licence Plate Recognition (ALPR), trata-se do processo de identificar letras e números que compõem a placa de um veículo em qualquer imagem, seja ela capturada por câmeras de segurança, smartphones ou outros dispositivos. O reconhecimento automático de placas veiculares detém uma variedade de aplicações, tais como sistemas de segurança pública, gestão de tráfego e controle de acesso para estacionamento ou residencial.

De acordo com Salazar et al. (2019), “o processo de funcionamento do ALPR consiste em três etapas: localização da placa, segmentação e reconhecimento dos caracteres”. A etapa de localização da placa consiste em, independentemente da imagem, encontrar a posição da placa na imagem. A segmentação consiste, por meio de técnicas, separar a região da placa da imagem original e os caracteres contidos na placa.

E por último, o reconhecimento dos caracteres consiste em processar cada caractere encontrado na placa e identificá-lo.

Entretanto, a realização dessas etapas enfrenta diversos desafios tanto para a localização da placa quanto para o reconhecimento dos caracteres. Esses problemas estão relacionados a diversos fatores:

- Iluminação: Níveis alto ou baixo de luminosidade podem ocultar características da placa. Sombras também podem criar áreas escuras sobre a placa, dificultando ou impossibilitando a detecção.
- Ângulo da Placa: Se a placa não estiver completamente visível na imagem devido ao ângulo da câmera, isso pode prejudicar ou impossibilitar sua detecção.
- Qualidade da Imagem: Imagens de baixa resolução ou com ruído significativo podem dificultar tanto a localização quanto o reconhecimento dos caracteres.
- Sujeira na Placa: Lama, poeira ou outros detritos podem ocultar caracteres e dificultar a detecção.
- Erosão e Desgaste dos Caracteres: O desgaste causado por fatores ambientais ou pelo tempo pode tornar os caracteres ilegíveis, dificultando ou impossibilitando seu reconhecimento.
- Objetos na Frente da Placa: Itens como folhas, adesivos ou outros objetos podem bloquear parcial ou totalmente a visão da placa.

Esses fatores demonstram a complexidade envolvida na localização e reconhecimento precisos das placas veiculares. Portanto, para superar esses problemas é necessário o estudo e aplicação de processamento de imagens e de redes neurais artificiais.

O objetivo deste trabalho é aplicar métodos de processamento de imagens e de redes neurais artificiais para criar um sistema capaz de localizar e reconhecer os caracteres presentes em placas de veículos.

2. Fundamentação teórica

A fundamentação teórica tem como objetivo fornecer uma revisão dos conceitos, teorias e pesquisas relevantes que formam a base do estudo. Este capítulo aborda os conceitos sobre Visão Computacional, Redes Neurais Artificiais, Redes Neurais Recorrentes, Redes Neurais Convolucionais, Processamento de Imagem, Haar Cascade, YOLO e bibliotecas e softwares utilizados no trabalho.

2.1. Visão Computacional

A Visão Computacional é um ramo da Inteligência Artificial que vem crescendo, seu objetivo é fazer com que o computador consiga “ver” e “entender” o mundo real, ou seja, seu objetivo é imitar a capacidade do olho humano, para que com isso seja capaz de realizar ainda mais tarefas.

De acordo com Brownlee (2019), “a Visão Computacional é definida como um campo de estudo que busca desenvolver técnicas para ajudar os computadores a ver e entender o conteúdo de imagens digitais, como fotografias e vídeos”. Para Silva (2020),

“busca-se fazer softwares e hardwares computacionais capazes de entender objetos, entidades e situações a partir de informação visual, como imagens ou frames de vídeo”.

Com esse crescimento, a Visão Computacional é vista em diversas aplicações, como na segurança pública, vista em radares, em veículos autônomos para detecção de objetos, na indústria, para inspeção e controle de qualidade de produtos e a mais conhecida é o reconhecimento facial.

Para o funcionamento de aplicações da Visão Computacional é utilizado os conceitos de Redes Neurais Convolucionais, Redes Neurais Recorrentes, Processamento de Imagens e o conceito de Aprendizado Profundo, que utiliza técnicas de Machine Learning.

2.2. Redes Neurais Convolucionais

As Redes Neurais Convolucionais (CNN, do inglês Convolutional Neural Networks) são um tipo de rede neural artificial comumente utilizada para extrair informações de imagens. Elas são fundamentais para a Visão Computacional, cujo objetivo é permitir que sistemas computacionais “enxerguem” e compreendam o mundo visual da mesma forma que os humanos.

A arquitetura da CNN é composta por camadas de convolução, pooling e camada totalmente conectada (do inglês, fully connected), essa arquitetura é mostrada na Figura 1.

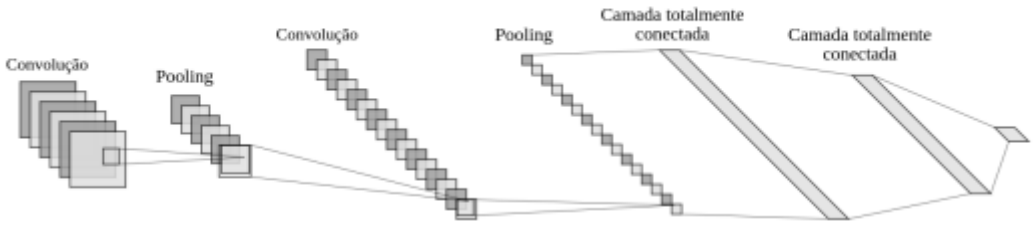

Figura 1. Essa figura é um exemplo da arquitetura de uma Rede Neural Convolucional.

2.3. Pré-processamento de imagens

O pré-processamento de imagens é uma etapa crucial quando se trata de visão computacional, essa etapa consiste em melhorar a imagem para que o sistema computacional consiga extrair as informações relevantes.

De acordo com Szeliski (2010), “a primeira etapa da visão computacional é o processamento de imagens, pois é possível através desse processo converter a imagem em uma forma adequada para que o meio computacional disponível, seja capaz de interpretá-las”. Para Marengoni e Stringhini (2009), “não é clara a fronteira entre o processamento de imagens e a visão computacional”. Pode-se dizer que o processamento de imagens é um processo onde a entrada do sistema é uma imagem e a

saída é um conjunto de valores numéricos, que podem ou não compor uma outra imagem.

Dentro desse contexto foram criadas técnicas e operações para a melhoria das imagens, como remoção de ruídos, conversões de cores, redimensionamento, equalização do histograma etc. Portanto, nessa seção serão apresentadas algumas dessas técnicas usadas para o desenvolvimento do trabalho.

2.4. Haar Cascade

O método do Haar Cascade é um algoritmo de detecção de objetos em imagens, foi introduzido por Paul Viola e Michael Jones em 2001. Essa técnica consiste em uma abordagem de Machine Learning onde utiliza três princípios, imagem integral, treinamento de classificadores e classificador em cascata.

A imagem integral é uma técnica ou algoritmo que permite com que seja feito um cálculo rápido que consiste na soma dos valores do pixel de uma determinada área, esse processo é feito para facilitar a extração das características haar de uma imagem.

O algoritmo de treinamento de classificadores utilizado neste projeto trata-se do AdaBoost (Adaptative boosting), esse algoritmo tem como técnica a combinação de vários classificadores fracos para transformar em um forte, ou seja, ele é utilizado para extrair as características haar mais relevantes. De acordo com Godoy e Morastico(2019), AdaBoost é basicamente um algoritmo que tem como objetivo construir um classificador forte a partir de combinações lineares de vários classificadores fracos.

Por último, o classificador em cascata tem como objetivo a seleção das melhores características, todo o processo de classificação é feito em estágios sequenciais. A cada estágio contém classificadores fortes para que sejam eliminadas características ou áreas que não são consideradas relevantes para o objeto treinado. Portanto, só seguem para os próximos estágios, características que são consideradas fortes, ou seja, relevantes, assim deixando eficiente o treinamento para a busca do objeto.

2.4. YOLO

O YOLO (You Only Look Once) é um modelo de detecção de objetos baseado em redes neurais convolucionais que revolucionou as técnicas disponíveis na época de sua criação. Desenvolvido por Joseph Redmon e Ali Farhadi, o modelo foi lançado em 2015 e se destacou por combinar precisão e eficiência em tarefas de detecção em tempo real. Diferentemente de métodos como A-CNN e Fast R-CNN, que processam partes da imagem de forma sequencial, o YOLO analisa a imagem inteira de uma só vez, reduzindo significativamente o tempo de processamento.

O funcionamento básico do YOLO pode ser observado na Figura 2, que ilustra o pipeline de detecção. Inicialmente, a imagem de entrada é redimensionada para um formato padrão exigido pelo modelo. Em seguida, ela é passada por uma rede neural convolucional que extrai as características relevantes e, por fim, produz caixas

delimitadoras (bounding boxes) associadas a níveis de confiança baseados no treinamento.

A arquitetura do modelo original é composta por 24 camadas convolucionais seguidas por 2 camadas totalmente conectadas, como mostrado na Figura 3. Essa configuração foi projetada para realizar a extração de características e a previsão de objetos em uma única etapa, tornando o modelo extremamente rápido e eficiente.

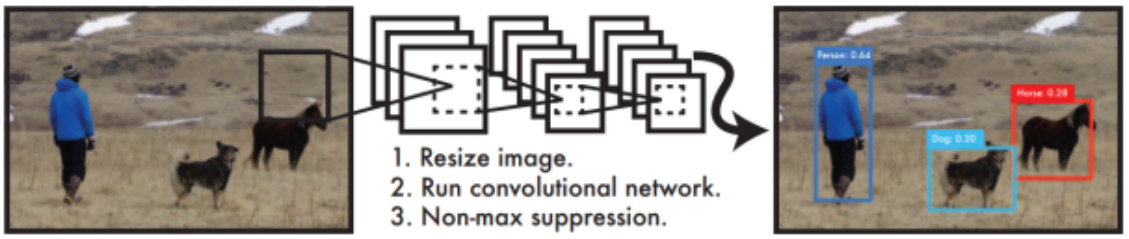

Figura 2. Essa figura apresenta o sistema de detecção do YOLO.

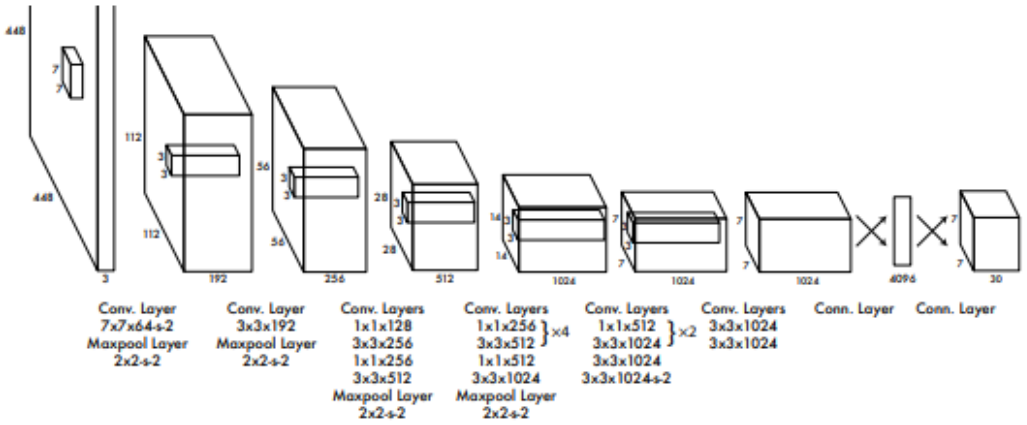

Figura 3. Essa figura apresenta a arquitetura do YOLO.

O YOLO V8 Ultralyrics foi lançado em 2023, sendo a versão mais recente e mais eficiente em relação às versões anteriores. A arquitetura do YOLOv8 possui algumas diferenças em relação a versões anteriores, ele emprega arquiteturas de backbone e neck de última geração, o termo backbone refere-se a parte inicial da rede neural convolucional e o termo neck a camada intermediária que liga o backbone a camada de saída, resultando em uma melhor extração de características e desempenho na detecção de objetos, adota uma cabeça (do inglês, head) Ultralytics dividida sem ancoragem, esse termo significa que não é definido âncoras para as caixas delimitadoras (do inglês, bounding boxes), o que contribui para uma melhor precisão e um processo

de detecção mais eficiente em comparação com abordagens baseadas em âncoras e uma variedade de modelos pré-treinados.

O YOLOv8 possui diversos modelos pré-treinados como YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, YOLOv8x, todos eles foram treinados utilizando o COCO dataset. O COCO dataset é um conjunto de dados de detecção de objetos, segmentação e legendagem em grande escala.

2.5. OpenCV

OpenCV, ou Open Source Computer Vision Library, é uma biblioteca de código aberto amplamente utilizada para processamento de imagens e visão computacional. Ela fornece uma grande variedade de algoritmos e ferramentas para análise e manipulação de imagens, detecção e rastreamento de objetos, reconhecimento de padrões, reconhecimento facial, calibração de câmeras, entre outras tarefas relacionadas à visão por computador (Awari, 2023).

2.6. Tesseract OCR

O Tesseract OCR é um software livre e de código aberto amplamente utilizado para o reconhecimento óptico de caracteres (ROC) ou em inglês Optical Character Recognition (OCR). O OCR é uma tecnologia que converte imagens contendo texto em texto digital editável. A versão 4.0 é fundamentada em um novo mecanismo de OCR, baseado em Redes Neurais Recorrentes (RNN) do tipo LSTM (Long Short-Term Memory, em português: memória de curto longo prazo).

3. Desenvolvimento e resultados

Nesta seção, são apresentados os experimentos realizados e os resultados finais obtidos ao longo do desenvolvimento deste trabalho. Além disso, discutem-se as dificuldades encontradas, as soluções adotadas e as melhorias realizadas para aprimorar a assertividade do sistema. Os resultados apresentados têm como objetivo validar a metodologia empregada, com comparações entre diferentes abordagens e ferramentas utilizadas, demonstrando a eficiência da solução proposta.

Para a realização deste trabalho, foi utilizado o dataset RodoSol-ALPR disponibilizado por Laroca et al. (2022), que contém 5000 imagens de carros brasileiros com a placa no formato antigo e 5000 imagens de carros brasileiros com o novo formato de placa Mercosul.

3.1. Detecção da placa

Nesta seção, são apresentados os experimentos realizados e os resultados finais obtidos na primeira etapa deste trabalho, que consiste em detectar a placa na imagem. Ademais, discutem-se as dificuldades encontradas, as soluções adotadas e as melhorias realizadas para aprimorar a eficiência do sistema.

3.1.1 Detecção de borda

No início do trabalho, a primeira ideia de algoritmo a ser utilizado para a detecção da placa foi empregar o conceito de detecção de bordas, visto que a placa veicular possui bordas que envolvem a placa.

O primeiro teste foi realizado nas 5000 imagens de carros com placa no formato antigo. Cada imagem é passada por uma função de pré-processamento para que as bordas contidas na imagem fiquem visíveis para a próxima etapa.

A Tabela 1 apresenta os resultados obtidos nesse primeiro teste, contemplando todas as variações implementadas visando o devido reconhecimento.

Tabela 1. Resultados por detecção de bordas

Processamento Imagem	Processamento placa	Placas encontradas	Placas correta	Tesseract OCR
Binarização, suavização e o algoritmo canny	Redimensionamento e a suavização	1150	580	Linguagem padrão (eng) e psm 6
Binarização, suavização e o algoritmo Sobel	Redimensionamento	1193	583	Linguagem padrão (eng) e psm 6
Binarização, suavização e o algoritmo Sobel	Redimensionamento	1279	527	Linguagem padrão (eng) e psm 6
Binarização, suavização e o algoritmo Sobel	Redimensionamento	1151	639	Linguagem padrão (eng) e psm 8

Dado esses resultados é possível notar que esse algoritmo não é eficaz para resolver o problema, um dos maiores problemas neste algoritmo foi a identificação da placa em carros brancos e cinzas, pois as bordas das placas não ficam completamente visíveis e se perdem no contraste do carro.

3.1.2 Haar Cascade

Visto que a primeira abordagem não obteve resultados satisfatórios, uma nova abordagem foi pesquisada e utilizada para tentar resolver o problema da localização das placas em imagens.

Como foi abordado anteriormente, para a utilização do método do Haar Cascade, foi necessário a utilização de imagens positivas e negativas para que o treinamento do classificador fosse realizado.

Para o primeiro teste foram utilizadas 3000 imagens positivas e 1041 imagens negativas, e o treinamento foi feito em 10 estágios. Outros testes foram realizados

alterando a quantidade de imagens positivas, tanto para placas antigas quanto para placas no formato Mercosul. A quantidade de imagens negativas manteve-se igual para todos os testes.

A Tabela 2 apresenta os resultados obtidos para as placas no formato antigo e a Tabela 3 para o formato Mercosul.

Tabela 2. Resultados Haar Cascade (placa antiga)

Distribuição treino/teste	Processamento placa	Placas encontradas	Placas corretas	Tesseract OCR
3000 imagens para treino e 2000 imagens para teste	Conversão para cinza e equalização do histograma	1297	581	Linguagem padrão (eng) e psm 10
2500 imagens para treino e 2500 imagens para teste	Conversão para cinza e equalização do histograma	1665	824	Linguagem padrão (eng) e psm 10
3500 imagens para treino e 1500 para teste	Redimensionamento e conversão de cores BGR para RGB e para escala de cinza	1038	657	Linguagem padrão (eng) e psm 10

Tabela 3. Resultados Haar Cascade (placa Mercosul)

Distribuição treino/teste	Processamento placa	Placas encontradas	Placas corretas	Tesseract OCR
3500 imagens para treino e 1500 imagens para teste	Redimensionamento e conversão para cinza	1297	581	Linguagem padrão (eng) e psm 10
4000 imagens para treino e 1000 imagens para teste	Redimensionamento e conversão para cinza	1665	824	Linguagem padrão (eng) e psm 10
4000 imagens para treino e 1000 imagens para teste	Redimensionamento, conversão para cinza, suavização, equalização do histograma e binarização de Otsu	271	785	Linguagem padrão (eng) e psm 10

Com os resultados obtidos, percebe-se que a técnica do Haar Cascade foi mais eficaz do que o primeiro algoritmo, mas ainda assim não obteve um resultado acima dos 70%. Mesmo encontrando as regiões da placa, a técnica ainda retorna falsos positivos (aponta regiões da imagem onde não contém a placa).

3.1.2 YOLOv8

Uma terceira abordagem foi utilizada buscando melhorar o resultado para a localização das placas. Nesse terceiro método dispôs-se do YOLOv8. Para a utilização do YOLOv8 é necessário o treinamento do modelo de objeto que se deseja detectar na imagem. Nessa perspectiva, as imagens do dataset foram divididas em três conjuntos, de modo a permitir que o YOLOv8 consiga treinar o modelo desejado. O primeiro conjunto são das imagens de treinamento, o segundo são as imagens para testes e por fim, o conjunto para validação. Para o treinamento ficaram 3500 imagens e 750 para teste e 750 para validação.

Foram treinados dois modelos, um focado em precisão e o outro focado em velocidade de processamento. O primeiro modelo foi treinado com base no YOLOv8x, que apresenta a maior acurácia, porém com um tempo de processamento mais alto. Já o segundo modelo utiliza o YOLOv8s, que possui acurácia e tempo de processamento menores em comparação ao primeiro. Para ambos treinamentos foram utilizadas 100 épocas, ou seja, o conjunto de imagens foi passado pelo modelo 100 vezes.

A Tabela 4 apresenta os resultados obtidos utilizando a detecção pelos modelos treinados utilizando o YOLOv8x e a Tabela 5 apresenta os resultados utilizando o modelo YOLOv8s.

Em relação ao tempo de processamento que o YOLO leva para analisar e processar cada detecção, o YOLOv8x demora cerca de 1570 milissegundos que corresponde a 1,57 segundos e o YOLOv8s demora aproximadamente 145 milissegundos que é cerca de 0,145 segundos.

Tabela 4. Resultados utilizando o modelo YOLOV8x

Quantidade de imagens	Pré-processamento da placa	Placas encontradas	Placas com caracteres corretos	Biblioteca de reconhecimento de caracteres
750 placas antigas	Redimensionamento e conversão para RGB e escala de cinza	750	271	Tesseract OCR: Linguagem padrão (eng) e psm 10
750 placas antigas	Redimensionamento e conversão para RGB e escala de cinza	750	308	Easy OCR: linguagem padrão (eng e pt)
750 placas antigas	Redimensionamento, conversão para RGB e escala de	750	77	Tesseract OCR: Linguagem padrão (eng) e

	cinza e binarização de Otsu			psm 10
750 placas antigas	Redimensionamento, conversão para RGB e escala de cinza e binarização de Otsu	750	170	Easy OCR: linguagem padrão (eng e pt)
750 placas antigas	Redimensionamento, conversão para RGB e escala de cinza, equalizada, suavizada e binarização de Otsu	750	248	Tesseract OCR: Linguagem padrão (eng) e psm 10
750 placas antigas	Redimensionamento, conversão para RGB e escala de cinza, equalizada, suavizada e binarização de Otsu	750	100	Easy OCR: linguagem padrão (eng e pt)
750 placas antigas	Conversão para RGB e escala de cinza	750	264	Tesseract OCR: Linguagem padrão (eng) e psm 10
750 placas antigas	Conversão para RGB e escala de cinza	750	310	Easy OCR: linguagem padrão (eng e pt)

Tabela 5. Resultados utilizando o modelo YOLOV8s

Quantidade de imagens	**Pré-processamento da placa**	**Placas encontradas**	**Placas com caracteres corretos**	**Biblioteca de reconhecimento de caracteres**
750 placas antigas	Conversão para RGB e escala de cinza	748	251	Tesseract OCR: Linguagem padrão (eng) e psm 10
750 placas antigas	Conversão para RGB e escala de cinza	748	304	Easy OCR: linguagem padrão (eng e pt)
750 placas antigas	Conversão para RGB	748	262	Tesseract OCR: Linguagem padrão (eng) e psm 10

750 placas antigas	Conversão para RGB	748	306	Easy OCR: linguagem padrão (eng e pt)
5000 placas mercosul	Conversão para RGB	4926	225	Tesseract OCR: Linguagem padrão (eng) e psm 10
5000 placas mercosul	Conversão para RGB	4926	570	Easy OCR: linguagem padrão (eng e pt)
5000 placas mercosul	Redimensionamento, conversão para RGB e escala de cinza	4926	572	Tesseract OCR: Linguagem padrão (eng) e psm 10
5000 placas mercosul	Redimensionamento, conversão para RGB e escala de cinza	4926	803	Easy OCR: linguagem padrão (eng e pt)

De acordo com a Tabela 4, foi verificada que a taxa de detecção da placa na imagem é de 100%, ou seja, todas imagens para validação do modelo foram detectadas com confiança maior que 80%, que é a métrica utilizada neste trabalho para ser considerada uma localização válida. Ainda de acordo com a Tabela 4, verificou-se que para o Tesseract OCR, a taxa de acerto dos caracteres encontrados varia entre 10,26% e 36,13%, já para o Easy OCR, a taxa de acerto varia entre 13,33% e 41,33%.

De acordo com a Tabela 5, utilizando o modelo YOLOv8s para o treinamento, a taxa de detecção varia entre 98,52% e 99,73%, um resultado expressivo, considerando que o modelo apresenta menor acurácia em comparação ao anterior, mas oferece maior velocidade de processamento, também tendo como parâmetro a confiança da detecção acima de 80%. Ainda de acordo com a Tabela 5, para o reconhecimento de caracteres nas placas antigas, a taxa de acerto do Tesseract OCR varia entre 33,55 % e 35,02%, já para as placas com o formato Mercosul, varia entre 4,56% e 11,61%. Para o Easy OCR, a taxa de acerto para as placas antigas varia entre 40,64% e 40,90% e para as placas com o formato Mercosul, varia entre 11,57% e 16,30%.

Portanto, conclui-se que o YOLOv8s apresentou resultados consistentes e satisfatórios para a tarefa de detecção de placas, enquanto as bibliotecas de reconhecimento de caracteres enfrentaram maiores dificuldades, com resultados que variaram de baixos a moderados, dependendo do tipo de placa analisada.

3.2. Reconhecimento dos caracteres

Antes de iniciar os experimentos, foi realizado um novo treinamento para criar um modelo do YOLO capaz de detectar as classes de "placa antiga" e "placa Mercosul".

Para esse novo modelo, utilizou-se um conjunto de imagens contendo tanto placas no padrão antigo quanto no padrão Mercosul.

O novo modelo foi treinado a partir do modelo pré-treinado YOLOv8m, o tempo de detecção desse modelo, leva cerca de 784 milissegundos ou 0,784 segundos. O dataset utilizado permanece o mesmo, porém com uma nova distribuição da quantidade de imagens para cada etapa, além da inclusão de uma nova classe de classificação. Foram organizados três conjuntos de imagens para o processo de treino, teste e validação, contendo, respectivamente, 6.750, 1.500 e 1.500 imagens.

Visto em testes anteriores que há uma confusão com caracteres parecidos, foi feita uma função de pós-processamento dos caracteres retornados pelo Tesseract OCR ou pelo Easy OCR que foram utilizados nos experimentos.

As funções de pós-processamento dos caracteres foram feitas pensando nos padrões dos caracteres que formam a placa, sendo "LLLNNNN" para o padrão de placas antigas e "LLLNLNN" para o Mercosul, onde L representa letras de A a Z e N representando os números de 0 a 9. Portanto, cada função tem o objetivo de trocar os caracteres semelhantes, caso onde deveria conter uma letra apareça um número e vice-versa.

Também foi desenvolvida uma função que apresenta as maiores confusões de caracteres, ou seja, é apresentado o caractere esperado e com qual ele foi confundido, essa função é utilizada para verificar a dificuldade que as bibliotecas encontram no reconhecimento dos caracteres.

3.2.1 Primeiro experimento

Para o primeiro experimento para verificar a assertividade do Tesseract OCR no reconhecimento dos caracteres contidos na placa, foi realizado um teste com 10.000 imagens, sendo 5000 imagens com placa padrão antigo e 5000 com padrão Mercosul.

Este primeiro experimento foi feito utilizando o Tesseract OCR e o Easy OCR, ambas bibliotecas de reconhecimento de caracteres não foram treinadas, foram utilizadas com seus formatos padrão e vale ressaltar que não foi utilizado primariamente o pré-processamento da placa.

A Tabela 6 apresenta os resultados obtidos nesse primeiro experimento, apresentando a assertividade obtida.

Tabela 6. Resultados do primeiro experimento

Placas reconhecidas	% de acertos	Biblioteca
2953 placas antiga corretas	59,06% de assertividade	Tesseract: Linguagem padrão (eng) e psm 10
1985 placas mercosul corretas	39,7% de assertividade	Tesseract: Linguagem padrão (eng) e psm 10
2524 placas antigas corretas	50,48% de assertividade	Easy OCR: eng,pt

1774 placas mercosul corretas	35,48% de assertividade	Easy OCR: eng,pt

O primeiro experimento apresentou uma assertividade maior no reconhecimento dos caracteres contidos na placa com padrão antigo do que com o padrão Mercosul, e o Tesseract OCR apresentou resultados melhores. Portanto, para os próximos experimentos não foi utilizado o Easy OCR por conta da sua assertividade e também devido ao tempo necessário para que ele retorne os caracteres encontrados na placa.

Os caracteres da placa antiga que mais apresentaram confusão para reconhecimento em ordem maior ocorrência, foram: Q -> O, 9 -> 3, M -> W e V -> Y. Já para a placa com padrão Mercosul, as maiores ocorrências foram: F -> E, Q -> O, 9 -> 1, 4 -> 5 e J -> I.

Algumas das dificuldades em reconhecer os caracteres são as semelhanças visuais na estrutura do caractere, como “Q” e “O”, ainda mais em um cenário que as imagens contendo a placa são de baixa ou média qualidade e com desgastes na placa. Portanto é notável que para o Tesseract e o Easy, reconhecer caracteres de uma fonte que não é pré-treinada e com imagens sem pré-processamento algum é um desafio.

3.2.2 Segundo experimento

Neste experimento visando melhorar a assertividade no reconhecimentos dos caracteres, a ideia a se seguir foi de utilizar uma função para pré-processar a placa para que fossem realçados os caracteres contidos na placa, depois de alguns testes foi definida uma função de pré-processamento para cada tipo de placa.

Pensando na melhoria, foi feito um treinamento do Tesseract OCR, para que ele pudesse entender as diferenças das fontes das placas antiga e mercosul, pois ele não é pré-treinado com as fontes, para que assim aumentasse a assertividade no reconhecimento dos caracteres. O treinamento do Tesseract OCR foi feito utilizando o Google Colab.

A tabela 7 apresenta os resultados obtidos nesse segundo experimento, no qual foi realizado o pré-processamento da placa e o treinamento do Tesseract para cada fonte da placa.

Tabela 7. Resultados do segundo experimento

Placas reconhecidas	% de acertos	Biblioteca
3233 placas antiga corretas	64,66% de assertividade	Tesseract: Mandatory e psm 10
2413 placas mercosul	48,26% de assertividade	Tesseract: Fe-Font e psm 10

O segundo experimento apresentou uma melhoria de 5,6% na assertividade das placas antiga e de 8,56% para Mercosul, os resultados ainda não foram satisfatórios e continuou apresentando confusões em relação aos caracteres semelhantes, mais

especificamente para o padrão antigo: Q -> O, 9 -> 3, M -> W e V -> Y e mercosul: Q -> O, F -> E, J -> I e 4 -> 5.

3.2.3 Terceiro experimento

Neste último experimento, objetivou-se aprimorar o treinamento do Tesseract, visto que os resultados encontrados para o reconhecimento de caracteres não atingiram 70%.

Neste sentido, para melhorar o treinamento foi necessário ajustar dois parâmetros, o parâmetro "maxpages" apresentado na Figura 24 e "max_iterations" disposto na Figura 25. Esses parâmetros aumentaram respectivamente a quantidade de páginas contendo as palavras que serão processadas e a quantidade de iterações que o treinamento irá executar.

A nova fonte foi nomeada como Mandatory1, a qual foi utilizada para esse experimento para reconhecer as placas antigas, e Fe-Font1 para a placa Mercosul. Assim, realizou-se o teste das novas fontes, juntamente com a fonte padrão "eng" para que fosse verificado se aumentaria a assertividade. A Tabela 8 apresenta os resultados obtidos neste experimento.

Tabela 8. Resultados do terceiro experimento

Placas reconhecidas	% de acertos	Biblioteca
3766 placas antiga corretas	75,32 % de assertividade	Tesseract: eng + Mandatory1 e psm 10
3603 placas antiga corretas	72,06 % de assertividade	Tesseract: Mandatory1 e psm 10
2900 placas mercosul corretas	58,00 % de assertividade	Tesseract: eng + Fe-Font1 e psm 10
3922 placas mercosul corretas	78,44 % de assertividade	Tesseract: Mandatory1 e psm 10

Após os experimentos, para a placa antiga a melhor combinação de idiomas do Tesseract foi "Mandatory1" + "eng" e para as placas Mercosul foi Mandatory1, as fontes que propiciaram melhor distinção entre os caracteres. Mas ainda sim houve bastante dificuldade em relação aos caracteres semelhantes. As confusões dos caracteres da placa antiga foram maiores em Q -> O, D-> 0, O -> Q, O -> D, pois dependendo do desgaste da placa a diferença de um para outro é mínima.

Portanto, para a versão final que irá apresentar a interface do trabalho desenvolvido, os idiomas ou linguagens serão: “Mandatory1 + eng “ para as placas antigas e “Mandatory1” para Mercosul.

3.2.4 Resultado final

A versão final ou resultado final, apresenta a interface gráfica do sistema que era tido como um dos objetivos do trabalho, assim como a criação do back-end para a simulação dos dados fornecidos pela API do SENATRAN.

O back-end foi desenvolvido em Node.js juntamente com o framework Express.js, e o banco de dados conta com todas as placas presentes no dataset RodoSol-ALPR.

Os dados retornados pela API são fictícios, apenas as placas são verdadeiras que estão dispostas no dataset utilizado, esses dados como:ano, modelo, cor do veículo, multas e débito são apresentados na Figura 4.

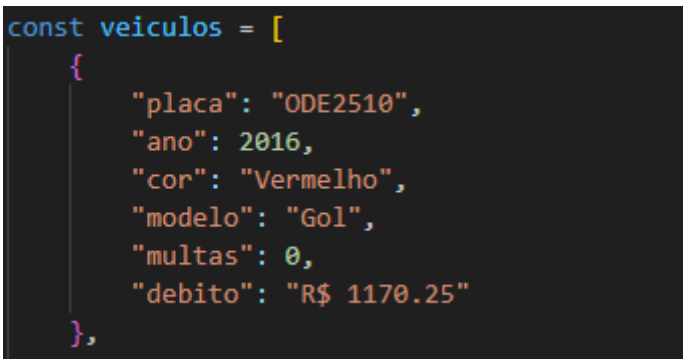

Figura 4. Essa figura apresenta os dados retornados pela API.

Para a interface gráfica, ela foi totalmente desenvolvida em Python juntamente com TkInter que é uma biblioteca padrão do Python para lidar com interface gráfica.

A interface gráfica foi dividida ao meio, do lado esquerdo é apresentada a imagem da placa detectada retornada pelo YOLO, e abaixo a mesma imagem mas pré-processada por uma função de pré-processamento e a classe da placa detectada (antiga ou Mercosul). Ao lado direito da interface, são apresentados todos os dados retornados pela API caso seja efetuado com sucesso o reconhecimento dos caracteres

contidos na placa, se não é apresentado “Não encontrado” para todas as informações, com exceção do dado “Placa” que é retornado pelo Tesseract OCR.

A Figura 5 apresenta um caso de sucesso, no qual é reconhecido corretamente todos os caracteres, e a Figura 6 apresenta um caso onde ele não consegue efetuar corretamente o reconhecimento dos caracteres.

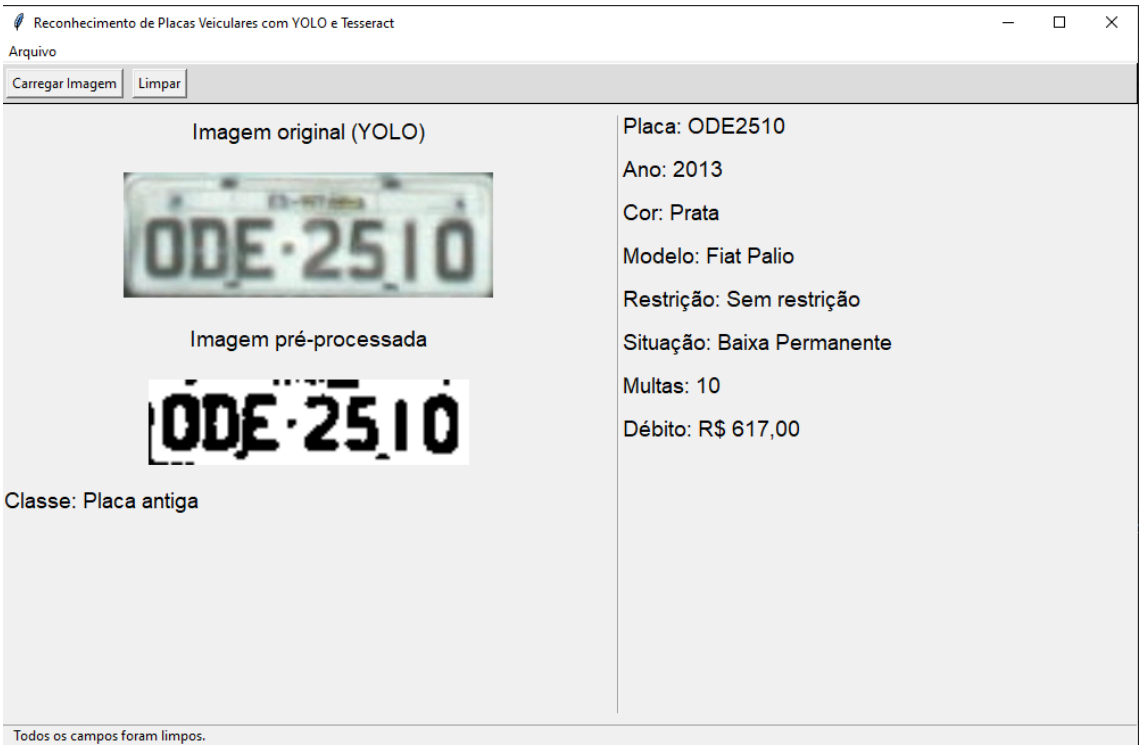

Figura 5. - Interface gráfica (caso sucesso).

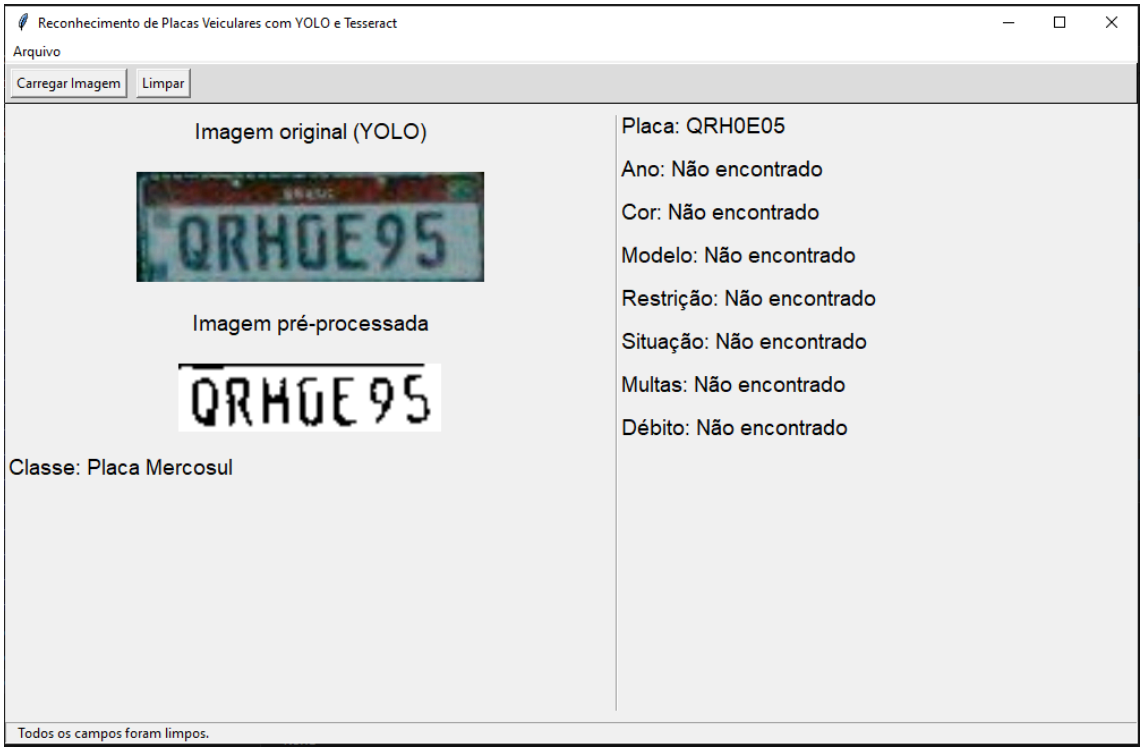

Figura 6. - Interface gráfica (caso erro).

4. Conclusão

Neste trabalho, foi realizado um estudo sobre processamento de imagens e redes neurais artificiais, implementando três abordagens para a detecção de placas veiculares em imagens: Detecção de Bordas, Haar Cascade e YOLO. Cada método foi avaliado quanto à sua eficácia, juntamente com o uso de bibliotecas especializadas para o reconhecimento óptico de caracteres (OCR).

Os resultados demonstraram que a abordagem baseada na biblioteca YOLO se destacou, alcançando taxas de precisão superiores a 98%, mostrando-se mais eficiente em comparação às demais, que, embora apresentem resultados regulares, não atingiram o mesmo desempenho.

Na etapa de reconhecimento de caracteres, foi aplicado um processo de pré-processamento para realçar as imagens das placas, seguido do uso das bibliotecas Easy OCR e Tesseract OCR. O Tesseract OCR foi treinado com as fontes Mandatory e Fe-Font, que não estão disponíveis nativamente, resultando em uma melhoria significativa na assertividade, que variava entre 39,7% e 59,06% (sem treinamento), e

passou a variar entre 75,32% e 78,44% (após treinamento). Apesar dessa evolução, desafios relacionados à qualidade das imagens e à similaridade entre caracteres persistiram.

Em resumo, o YOLO mostrou-se a abordagem mais eficaz para a detecção de placas, enquanto o Tesseract OCR, após treinamento, destacou-se como a melhor solução para o reconhecimento de caracteres.

Para trabalhos futuros, sugere-se aprimorar o treinamento do Tesseract OCR para lidar com as similaridades entre caracteres e melhorar o processo de pré-processamento das placas, com o objetivo de realçar ainda mais os caracteres e facilitar o reconhecimento.

Referências

Awari. Aprenda OpenCV com Python: tudo o que você precisa saber para se tornar um expert em visão computacional. Disponível em: https://awari.com.br/aprenda-opencv-com-python-tudo-o-que-voce-precisa-saber-para-se-tornar-um-expert-em-visao-computacional/?utm_source=blog&utm_campaign=projeto+blog&utm_medium=Aprenda%20Opencv%20Com%20Python:%20Tudo%20O%20Que%20Voc%C3%AA%20Precisa%20Saber%20Para%20Se%20Tornar%20Um%20Expert%20Em%20Vis%C3%A3o%20Computacional. Acesso em: 29 mai. 2024.

Brownlee, J. Deep Learning for Computer Vision: Image Classification, Object Detection, and Face Recognition in Python. Machine Learning Mastery, 2019. E-book.

Godoy, S. P.; Morastico, A. C. Segurança física com reconhecimento facial. Trabalho de Conclusão de Curso (Graduação em Engenharia da Computação) – Centro Universitário Internacional UNINTER, Curitiba, 2019. Disponível em: https://repositorio.uninter.com/bitstream/handle/1/1382/TCC%20ENG%20COMP%20Samuel%20Pereira%20de%20Godoy%2c%20Andrey%20Cesar%20Morastico%2c%202019.pdf?sequence=1&isAllowed=y. Acesso em: 29 mai. 2024.

Laroca, R.; Cardoso, E. V.; Lúcio, D. R.; Estevam, V.; Menotti, D. On the Cross-dataset Generalization in License Plate Recognition. International Conference on Computer Vision Theory and Applications (VISAPP). pp. 166-178.2022.

Maregoni, M.; Stringhini, S. Tutorial: Introdução à Visão Computacional usando OpenCV. Revista de Informática Teórica e Aplicada, [S. l.], v. 16, n. 1, p. 125–160, 2010. DOI: 10.22456/2175-2745.11477. Disponível em: https://seer.ufrgs.br/index.php/rita/article/view/rita_v16_n1_p125. Acesso em: 27 mai. 2024.

Redmon, J.; Divid, A.; FarHadi, A. You only look once: unified real-time object detection. In: CONFERENCE ON COMPUTER VISION AND PATTERN

RECOGNITION (CVPR), 2015, Las Vegas, NV, USA. Anais [...]. [S.l.: s.n.], 2015. Disponível em: https://arxiv.org/pdf/1506.02640. Acesso em: 2 jun. 2024.

Salazar, J. A. D.; Barbosa, F. G. O.; Stremmer, M. R.; Pinto, T. L. F. C. Sistema automático para reconhecimento de placas veiculares: comparação entre SVM-HOG e Deep Learning. Anais do 14º Simpósio Brasileiro de Automação Inteligente (SBAI), Ouro Preto, MG, Brasil, 2019.

Silva, T. Visão computacional e racismo algorítmico: branquitude e opacidade no aprendizado de máquina. Revista da Associação Brasileira de Pesquisadores Negros, v. 12, n. 31, p. 428-448, 2019-2020. Disponível em: https://abpnrevista.org.br/site/article/view/744/774. Acesso em: 27 mar. 2024.