INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - CAMPUS BETIM
BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Rafael Enzio Moreira Teixeira

Reconhecimento Automático de Placas de Automóveis no MERCOSUL Usando YOLOv8 e Tesseract OCR

Betim
2023

Rafael Enzio Moreira Teixeira

Reconhecimento Automático de Placas de Automóveis no MERCOSUL usando YOLOv8 e Tesseract OCR

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais Campus Betim, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Orientador: Leandro Freitas de Abreu

Betim

2023

FICHA CATALOGRÁFICA

T266r Teixeira, Rafael Enzio Moreira
Reconhecimento automático de placas de automóveis no Mercosul usando YOLOv8 e Tesseract OCR / Rafael Enzio Moreira Teixeira. – 2023.
56 f. : il.

Trabalho de conclusão de curso (Bacharelado em Engenharia de Controle e Automação) - Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Câmpus Betim, 2023.

Orientação: prof. Dr. Leandro Freitas de Abreu

1. Super-resolução de imagem. 2. Detecção de objeto. 3. Reconhecimento óptico de caracteres. 4. Placa veicular. 5. Engenharia de Controle e Automação I. Teixeira, Rafael Enzio Moreira. II. Título.

CDU: 681.5

Ficha catalográfica elaborada pelo Bibliotecário Denísio Pereira Marcos CRB-6/3142

Rafael Enzio Moreira Teixeira

RECONHECIMENTO AUTOMÁTICO DE PLACAS DE AUTOMÓVEIS NO MERCOSUL USANDO YOLOV8 E TESSERACT OCR

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais Campus Betim, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Betim, 14 de dezembro de 2023.

BANCA EXAMINADORA

Prof. Dr. Leandro Freitas
DAUTI – IFMG Campus Betim (orientador)

Prof. Me. Virgil Del Duca Almeida
DAUTI – IFMG Campus Betim

Prof. Me. Fernando Thomé de Azevedo Silva
DAUTI – IFMG Campus Betim

Agradecimentos

Aos meus familiares, por não me deixarem desanimar durante estes longos anos, sem vocês eu não chegaria até aqui, o apoio emocional foi fundamental para me manter motivado e focado.

Ao meu orientador Leandro Freitas, que foi essencial tanto para meu crescimento técnico quanto para meu crescimento pessoal.

A todos os servidores, amigos e professores do Instituto Federal de Minas Gerais, unidade Betim pela oportunidade de crescimento e aprendizado.

Agradeço também aos membros da banca examinadora por dedicarem seu tempo para avaliar este trabalho.

Resumo

O reconhecimento automático de placas veiculares, é uma tecnologia que utiliza sistemas de câmeras, processamento de imagem e algoritmos de reconhecimento de caracteres para identificar e registrar as placas de veículos. Possui várias aplicações, como controle de acesso em estacionamentos, pedágios automáticos, monitoramento de tráfego, segurança de áreas restritas e aplicação da lei em caso de carros furtados ou com documentação irregular.

Neste trabalho, para realizar o reconhecimento automático de placas veiculares utilizou-se do detector de objetos *YOLOv8*, gerando assim um conjunto de imagem do objeto placa veicular. No conjunto de imagens é feito um pré-processamento composto por duas etapas, técnica de super-resolução de imagem e operações de filtragem com a biblioteca do OpenCV. Para realizar o reconhecimento óptico de caracteres da imagem e utilizado o Tesseract OCR.

O sistema de reconhecimento automático de placas veiculares foi avaliado num conjunto de 21 casos, conseguido realizar a correta identificação da placa veicular em todos os casos e o reconhecimento de caracteres em 19 casos.

Palavras-chave: *YOLOv8*, Tesseract OCR, super-resolução de imagem, OpenCV, detecção de objeto, placa veicular, reconhecimento óptico de caracteres.

Abstract

Automatic license plate recognition is a technology that uses camera systems, image processing and character recognition algorithms to identify and register license plates. It has various applications, such as access control in parking lots, automatic tolls, traffic monitoring, security in restricted areas and law enforcement in the case of stolen cars or those with irregular documentation.

In this work, the YOLOv8 object detector was used to perform automatic license plate recognition, thus generating a set of images of the license plate object. The image set undergoes a preprocessing composed of two stages: image super-resolution technique and filtering operations with the OpenCV library. Tesseract OCR was used to perform optical character recognition of the image.

The automatic license plate recognition system was evaluated in a set of 21 cases, managing to correctly identify the license plate in all cases and recognize the characters in 19 cases.

Keywords: YOLOv8, Tesseract OCR, image super-resolution, OpenCV, object detection, license plate, optical character recognition.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

YOLO – You Only Look Once

CNN – Rede neural convolucional

SRI – Super-resolução de imagem

RNN – Rede neural recorrente

RDR – Rede densa residual

LSTM – Long Short-Term Memory

OpenCV – Open Computer Vision Library

RDN – Residual Dense Network

RDB – Bloco Residual Denso

FLC – Fusão Local de Características

ALR – Aprendizagem Local de Resíduos

FGC – Fusão Global de Características

SUMÁRIO

1. INTRODUÇÃO

O crescente interesse pela automatização de processos e a introdução de novas tecnologias são fatores de grande motivação para diversas pesquisas no ramo de reconhecimento de padrões em imagens e vídeos. O avanço da tecnologia digital, em conjunto com desenvolvimento de novos algoritmos, tem permitido um número de aplicações cada vez maior para resolver problemas em diversas áreas, tais como medicina, biologia, automação industrial, sensoriamento remoto, astronomia, militar, segurança e vigilância (GONZALEZ, 2010). Alguns exemplos são sistemas de reconhecimento facial, sistemas de reconhecimento de impressões digitais e, em especial, o reconhecimento óptico de caracteres permite a identificação automática de caracteres escritos em diversos idiomas a partir de imagens de documentos e o reconhecimento automático de placas de veículos a partir de imagens ou gravações (XIE, 2018).

Dado este contexto, este trabalho tem em propor um sistema capaz de aplicar reconhecimento automático de placas veiculares. Para isso, pretende-se utilizar o algoritmo de detecção de objetos em tempo real *You Only Look Once* (YOLO), para a detecção do objeto placa veicular em um frame de vídeo. Para o reconhecimento dos caracteres que compõem a placa, será utilizado técnicas de super-resolução, OpenCV e Tesseract OCR.

1.1. Justificativa

A demanda de realizar o reconhecimento de uma placa veicular vem ganhando espaço comercial, uma vez que pode ser utilizado em inúmeras aplicações, como em controle de acesso, monitoramento e segurança. Tendo em mente que o uso de recursos humanos para anotação da placa de todos os veículos que passam em um determinado local, como em um estacionamento privado por exemplo é dispendioso. Ainda existindo a possibilidade de falha e negligência humana, com o aumento do número de veículos trafegando pelos centros urbanos, torna-se cada vez mais necessário o controle dos automóveis que passam por determinados lugares de uma maneira eficaz. Um sistema automatizado que controle a entrada desses veículos em tempo hábil, permitindo um fluxo maior de entrada em menos tempo, é capaz de

reduzir a necessidade de recursos humanos para este controle, além de minimizar os erros cometidos.

1.2 Objetivo Geral

Desenvolver uma solução baseada em visão computacional que permita automatizar o processo de identificação das placas de veículos em vídeos, sem a necessidade de intervenção humana, com uso de detecção de padrões e bibliotecas de software, na linguagem de programação Python.

1.2.1 Objetivos Específicos

- Pesquisar na literatura sobre soluções para o problema da detecção de placa de automóveis por meio de vídeo;
- Treinar o método de detecção YOLOv8 da Ultralytics para reconhecimento de placas veiculares de automóveis;
- Validar o reconhecimento do objeto placa veicular;
- Implementar o reconhecimento de caracteres da placa veicular;
- Validar o reconhecimento de caracteres da placa veicular.

1.3 Organização do trabalho

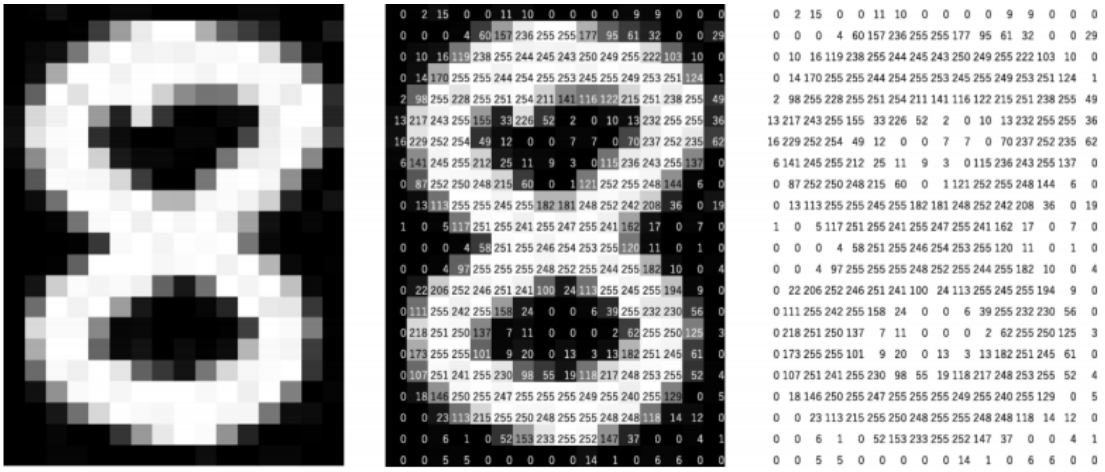
O primeiro capítulo deste trabalho contempla a introdução e contextualização do problema que será estudado. No segundo capítulo será tratado e definido toda a fundamentação necessária para a compreensão e aplicação do projeto. No terceiro capítulo os materiais e métodos serão definidos. No quarto os resultados obtidos e a interpretação dos resultados. Por fim, no quinto, tem-se conclusão e futuras melhorias.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Visão computacional

Define-se visão computacional como o conjunto de métodos e técnicas através dos quais, sistemas computacionais podem ser capazes de interpretar imagens, ou seja, capazes de transformar um conjunto de dados digitais representando uma imagem em uma estrutura de dados descrevendo a semântica deste conjunto em um contexto qualquer (WANGENHEIM, 2010). Na Figura 1 é apresentada a maneira mais simples de representar a imagem, sendo está na forma de uma matriz.

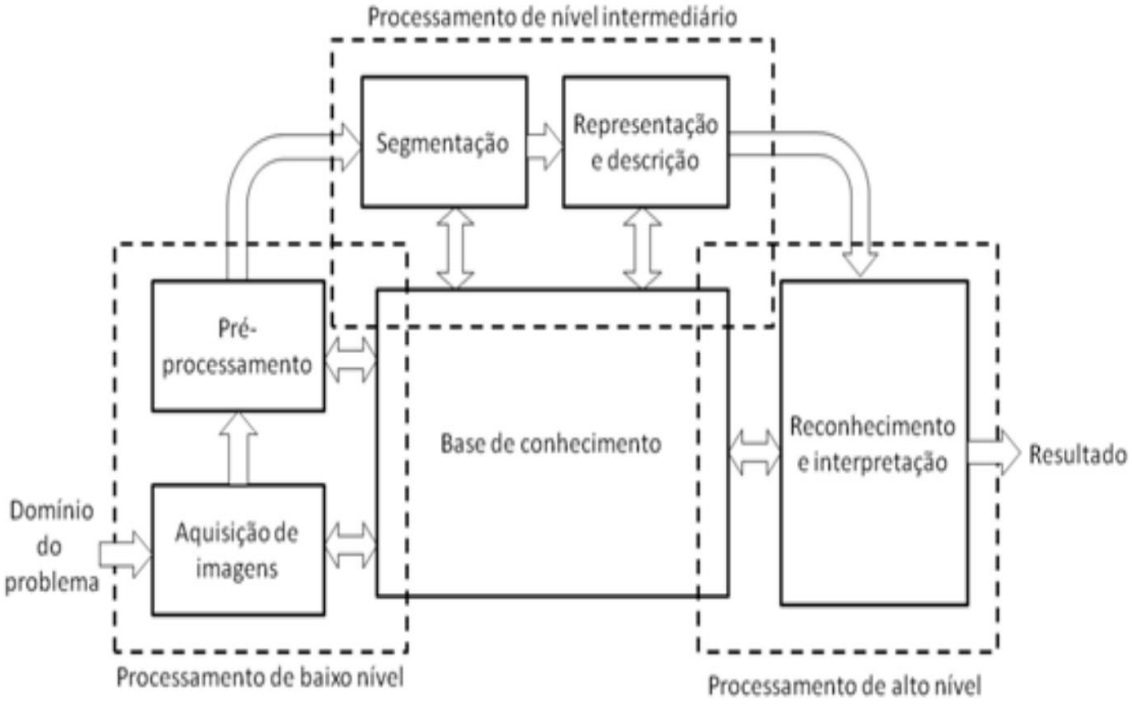
Figura 1 – Representação de como uma imagem e vista por um computador.



Fonte: Analytics Vidhya.

Já houve grande dificuldade em processar dados de imagens, mas com os avanços da computação, temos hoje a Visão Computacional, que está presente nos mais diversos campos de atuação, manipulando dados simples ou complexos (BOGGIONE, 2004). Mas, para que se possa utilizar todas as informações úteis contidas nas imagens, às vezes é necessário passar por um processo de melhoria e seleção de dados (BITTENCOURT, 2009). O Processamento de Imagens e composto por basicamente 6 etapas, conforme a Figura 2, não necessariamente há passagem por todas as etapas.

Figura 2 – Etapas básicas de um Processamento de Imagens.



Fonte: (GONZALEZ RAFAEL, 2002).

A aquisição de imagens é a primeira etapa, onde a imagem é adquirida, pode ser realizada por sensores, ou de forma digital por uma câmera. O pré-processamento é a segunda etapa, responsável por aplicar técnicas de transformações lineares e não lineares às imagens com o objetivo de realçar detalhes na imagem que dificultam a localização de objetos. A utilização destas técnicas para a localização de placas é comum na literatura. Os trabalhos de JYOTHI (S. JYOTHI et al.,2017), WANG (WANG, 2009) e CHEN (CHEN, 2013) podem exemplificar o uso dessas técnicas para melhorar as condições da imagem para localizar placas. Na etapa de segmentação, a imagem é dividida em sub-imagens da imagem inicial de acordo com sua significância, desde que possuam alguma característica importante para a aplicação. A representação consiste nas diversas formas de armazenar as regiões obtidas após a segmentação. A nova representação possui informações sobre a forma e topologia dos objetos. Para complementar a representação, a descrição serve para extrair características estruturais. No reconhecimento, um rótulo é associado a cada objeto segmentado, enquanto a interpretação associa um significado a cada conjunto de objetos. Podendo recorrer à base de conhecimento quando se dispõe de conhecimento prévio do resultado esperado. A base de conhecimento, além de controlar a interação do

processo, é responsável por guiar a operação de cada etapa no processamento (GONZALEZ RAFAEL, 2002).

No desenvolvimento de aplicações da Visão Computacional, deve-se tomar muito cuidado na escolha de cada componente do sistema. Para que se possa escolher bons algoritmos de Processamento de Imagens e alcançar bons resultados na interpretação das informações, é interessante obter como entrada do sistema uma imagem de boa qualidade (DAVIES, 2005). Com isso em mente deve-se levar em conta alguns parâmetros do sistema de visão (JAIN, 1995):

- Campo de Visão: área visível do objeto em estudo que incide sobre o sensor.
- Distância de Trabalho: distância da parte frontal das lentes até a superfície do objeto.
- Profundidade de Campo: representa a maior distância (em termos profundidade no campo de visão) que pode ser mantida em foco no objeto de estudo para uma determinada distância de trabalho.
- Resolução: menor porção do objeto em estudo que pode ser distinguida pelo sistema, é bem conhecida pela expressão “resolução espacial”.
- Tamanho do Sensor: representa o tamanho da área ativa do sensor.

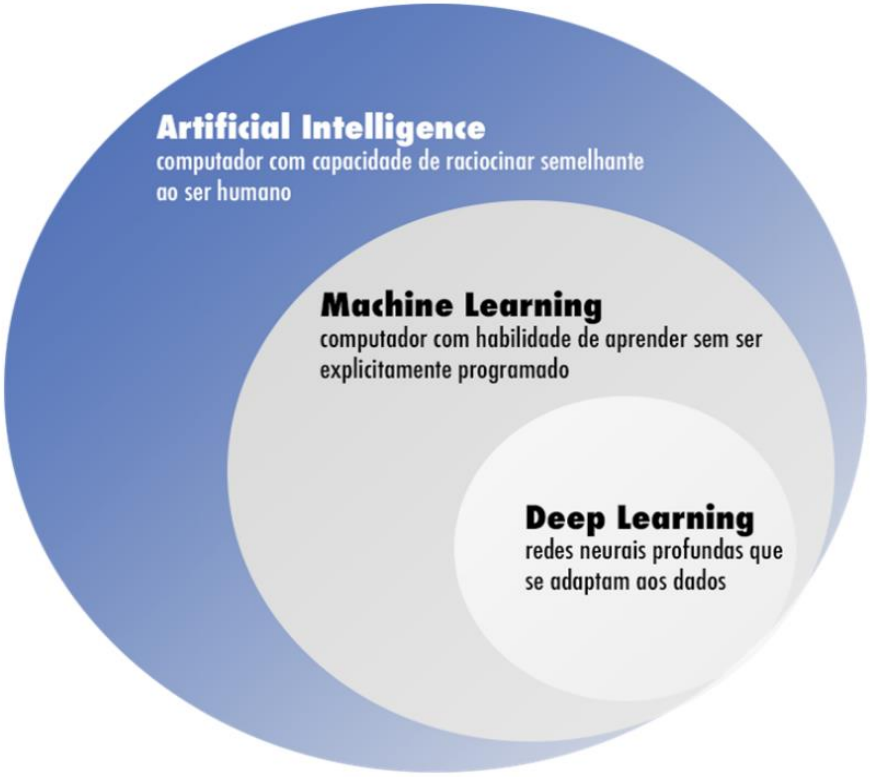
2.2 Aprendizado profundo

Pesquisas nessa área datam dos anos 40, passando por três grandes picos de relevância referenciados por outros nomes: *cybernetics* entre as décadas de 40 e 60, *connectionism* entre os anos 80 e 90 e, por fim, aprendizado profundo a partir de meados dos anos 2000 (GOODFELLOW, 2016). O aprendizado profundo ou rede neural se refere a uma classe de técnicas de aprendizado de máquina. Alguns dos importantes motivos para a popularidade do aprendizado profundo atualmente são o aumento drástico de processamento nos computadores com a utilização de placas de vídeo Unidade de Processamento Gráfico, em conjunto com um custo menor de hardware e os recentes avanços na pesquisa de aprendizado de máquina e processamento de informações (DENG et al., 2013).

A Figura 3 apresenta o contexto do aprendizado profundo com relação às áreas de Inteligência Artificial e Aprendizado de Máquina. A área de Aprendizado Profundo está

contida dentro da área de Aprendizado de Máquina, esta área por sua vez dentro da área de Inteligência Artificial.

Figura 3 – O aprendizado profundo (*Deep Learning*), Aprendizado de máquina (*Machine Learning*) e inteligência artificial (*Artificial Intelligence*).



Fonte: (Alessandro Cauduro, 2018).

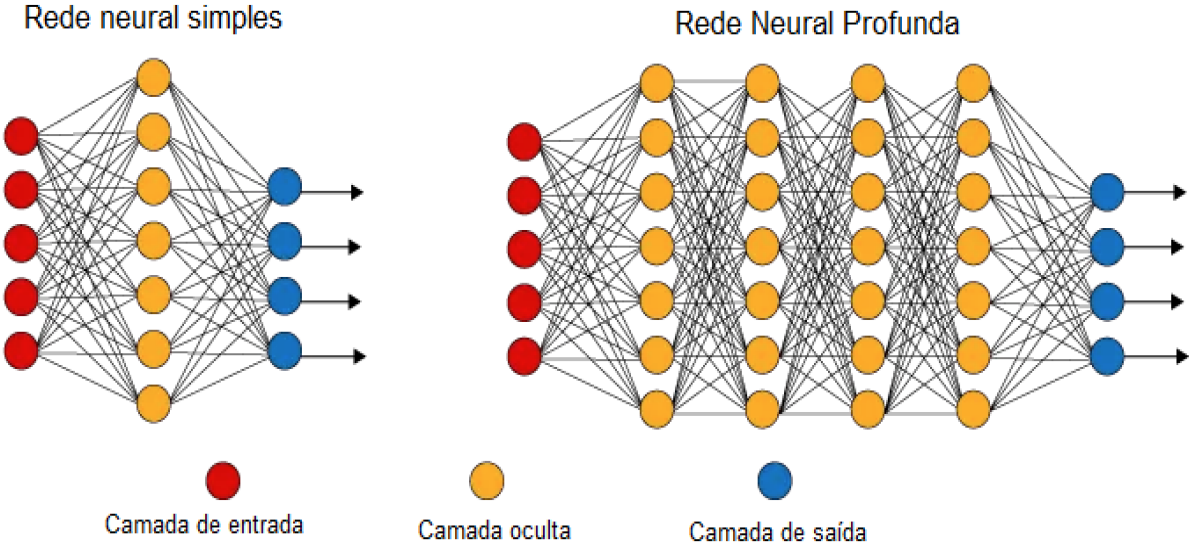
No *Machine Learning overfitting* e *underfitting* são conceitos que se referem ao ajuste do modelo.

- *overfitting* ocorre quando o modelo se adaptou muito bem aos dados de treinamento, mas não generaliza bem para novos dados, ou seja, não consegue lidar com dados que não foram vistos durante o treinamento. Isso acontece porque o modelo decorou o conjunto de dados de treinamento, mas não aprendeu os padrões importantes para generalizar para novos dados;
- *underfitting* ocorre quando o modelo não se adapta bem sequer aos dados de treinamento. Isso acontece quando o modelo é muito simples em relação aos dados que está tentando modelar, e não consegue.

Pode-se determinar se um modelo preditivo está fazendo o *underfitting* ou *overfitting* dos dados de treinamento consultando o erro de previsão nos dados de treinamento e nos dados de avaliação. Compreender o ajuste de modelo é importante para entender a causa raiz da precisão insatisfatória de modelo.

O aprendizado profundo usa camadas de neurônios matemáticos para processar dados, compreender a fala humana e reconhecer objetos visualmente. A informação é passada através de cada camada, com a saída da camada anterior fornecendo entrada para a próxima camada. A primeira camada em uma rede é chamada de camada de entrada, enquanto a última é chamada de camada de saída, as camadas entre as duas são referidas como camadas ocultas. Cada camada é tipicamente um algoritmo simples e uniforme contendo um tipo de função de ativação. A seguir temos a Figura 4 mostrando a representação de Rede Neural Simples e Rede Neural Profunda.

Figura 4 – Rede Neural Simples e Rede Neural Profunda.



Fonte: (DEEP Learning Book, 2022).

Os seguintes aspectos tornam as redes neurais especiais (TONY YIU, 2019):

- Cada neurônio é seu próprio modelo em miniatura com sua própria tendência e conjunto de recursos e pesos de entrada.
- Cada neurônio individual alimenta vários outros neurônios individuais em todas as camadas ocultas do modelo. Assim, acabamos com modelos plugados em outros modelos de uma forma em que a soma é maior do que suas partes. Isso

permite que as redes neurais se ajustem a todos os cantos e fendas de nossos dados, incluindo as partes não lineares.

- A versatilidade da abordagem de muitos modelos interconectados e a capacidade do processo de retro propagação para definir de forma eficiente e otimizada os pesos e vieses de cada modelo permite que a rede neural aprenda de forma robusta com os dados de maneiras que muitos outros algoritmos não podem.

Para criação de redes neurais de aprendizado profundo, o TensorFlow é um dos principais *frameworks* do mercado. Com ele, é possível agilizar e facilitar o processo de obtenção de dados, treinar modelos, realizar predições e refinar resultados futuros (TENSORFLOW, 2021). TensorFlow é um *framework* código aberto para Python, Javascript, C/C++ e Go, trazendo um desenvolvimento simplificado para o programador, por meio de um modelo baseado em fluxo de dados. Sua execução se passa sobre uma aplicação de alta performance escrita em C/C++. A equipe de engenheiros da Google, uniu as facilidades do Python com a alta performance do C/C++. Sua arquitetura se divide em três partes principais, sendo elas (ARAUJO, 2017):

- Pré-processamento dos dados;
- Construção dos modelos;
- Treinamento e estimativas do modelo criado.

2.3 Reconhecimento da placa veicular de automóveis

Na literatura têm sido propostos diversos métodos que buscam realizar o reconhecimento de placas veiculares em imagens ou vídeo. Na Figura 5 são apresentados os modelos de placas veiculares brasileiras.

Figura 5 – Placa veicular brasileira padrão MERCOSUL.

Fonte: (TECNOBLOG, 2018).

A tarefa de localizar placas veiculares possui diversos obstáculos, como: nível de ruído, condições de iluminação, ângulo de perspectiva da cena, objetos que podem ser confundidos com placas veiculares, distância entre a placa e a câmera, entre outros que influenciam o desempenho (FONSECA GALINDO, 2016).

Diversas técnicas já foram abordadas ao longo do tempo, e nota-se que a tendência das pesquisas está direcionada em termos de técnica e algoritmo como: processamento de imagens, aprendizagem de máquina e aprendizagem profunda. Exemplos de algoritmos baseados em aprendizagem profunda para executar a detecção de placas veiculares: Single Shot Multibox Detector - SSD (LIU, 2016), Faster R-CNN (REN et al., 2015) e RNC com o YOLOv2 (LAROCA et al., 2018). Tais algoritmos foram utilizados para detectar padrões em diferentes países, sendo o Brasil um deles. Alguns outros trabalhos foram desenvolvidos na área:

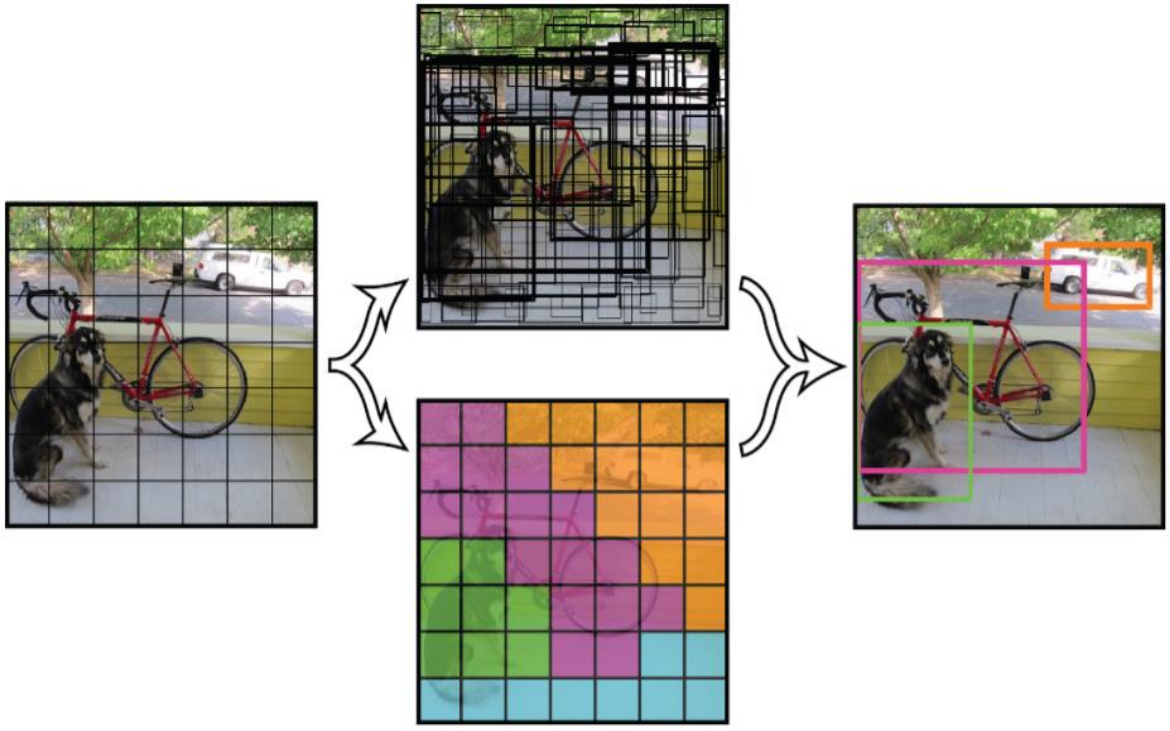
- Xie em 2018 - A new cnn-based method for multi-directional. IEEE Transactions on Intelligent Transportation Systems, 2018 (XIE, 2018);
- Cavalcanti em 2015 - Brazilian vehicle identification using a new embedded plate recognition system (CAVALCANTI et al., 2015);
- Gonçalves em 2016 - License plate recognition based on temporal redundancy (GONÇALVES, 2016);
- Yuan em 2016 - A robust and efficient approach to license plate detection (YUAN et al., 2016);
- Montazzolli e Jung em 2017 - Real-time brazilian license plate detection and recognition using deep convolutional neurais networks (MONTAZZOLLI, 2017).

2.4 *YOLO*

Em 2015 foi apresentada uma estratégia para possibilitar a detecção de objetos em tempo real (J.REDMON, 2016), a qual foi denominada *You Only Look Once* (*YOLO*). Para o seu funcionamento o *YOLO* utiliza uma rede neural profunda, cuja arquitetura é chamada de *Darknet*.

As soluções para classificação de objetos costumavam utilizar uma abordagem de alto custo computacional, em que a imagem é analisada pelo classificador milhares de vezes em diferentes escalas para gerar pontuações em regiões e identificar objetos, como é o caso das técnicas *R-CNN* e *Fast R-CNN* (J.REDMON, 2016). O algoritmo *YOLO* baseia-se na arquitetura das Redes Neurais Convolucionais, e possui grande capacidade de generalização, com bons resultados para detecção de vários objetos. A Figura 6 mostra o modelo *YOLO* divide uma imagem em grade e para cada célula, destaca caixas delimitadoras atribuindo um valor de confiança e a probabilidade de o objeto ser de uma determinada classe.

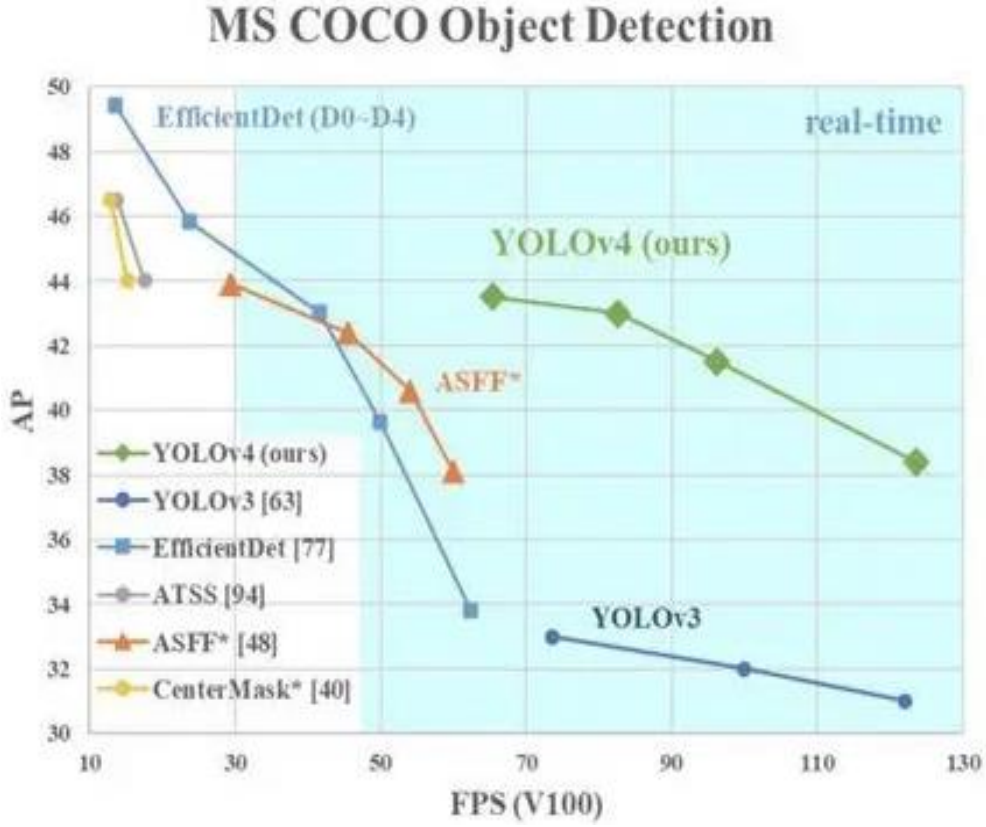
Figura 6 – Estratégia *YOLO* para detecção de objetos.



Fonte: (J. REDMON, 2016).

A evolução do algoritmo, em sua primeira versão, *YOLO* foi desenvolvida por 4 pesquisadores: Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. Até sua terceira versão, ela foi aprimorada por Joseph Redmon e Ali Farhadi. A quarta versão, foi lançada em abril de 2020, sendo oficializada após a publicação do *paper "YOLOv4: Optimal Speed and Accuracy of Object Detection*" por Alexey Bochkovskiy, Chien-Yao Wang e Hong-Yuan Mark Liao. As principais características que podem ser destacadas na quarta versão são melhoria na velocidade de inferência, acurácia e ser mais eficiente para rodar em uma Unidade de Processamento Gráfico.

Figura 7 – Comparação entre *YOLO*v4 e as outras abordagens mais eficientes atualmente.



Fonte: (J.REDMON, 2016).

Como pode ser visto na Figura 7, além de ser mais rápido e mais preciso que o EfficientDet, o *YOLO*v4 também supera o RetinaNet/MaskRCNN (Facebook Pytorch/Detectron) no dataset COCO. Após o *YOLO*v4 surgiu-se diversas versões, sendo estas (UTRALYTICS, 2023):

- O *YOLO*v5 melhorou ainda mais o desempenho do modelo e adicionou novos recursos, como otimização de hiperparâmetros, rastreamento integrado de experimentos e exportação automática para formatos de exportação populares;
- *YOLO*v6 foi de código aberto pela Meituan em 2022 e está em uso em muitos dos robôs de entrega autônomos da empresa;
- *YOLO*v7 adicionou tarefas adicionais, como estimativa de pose no conjunto de dados de pontos-chave COCO;
- *YOLO*v8 é a versão mais recente do *YOLO* da empresa Ultralytics. Como um modelo de última geração (SOTA), o *YOLOv8* baseia-se no sucesso das versões anteriores, introduzindo novos recursos e melhorias para o desempenho, a flexibilidade e a eficiência. *YOLOv8* oferece suporte a uma gama completa de tarefas de IA de visão, incluindo detecção, segmentação, estimativa de pose, rastreamento e classificação. Essa versatilidade permite que os usuários aproveitem os recursos do *YOLOv8* em diversos aplicativos e domínios.

Algumas métricas de perda para se avaliar o desempenho do treinamento do *YOLOv8* são:

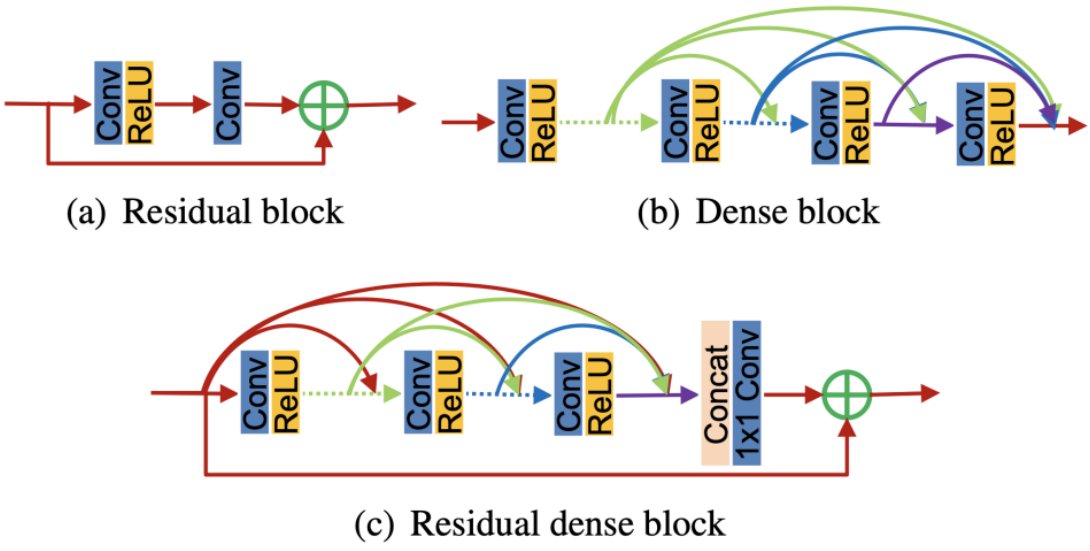
- *box_loss:* métrica de perda, que mede o erro nas coordenadas e dimensões previstas da caixa delimitadora em comparação com os objetos de verdade;
- *cls_loss:* métrica de perda, que mede a exatidão da classificação de todas as caixas delimitadoras previstas;
- *dfl_loss*: métrica de perda, que mede o erro nas camadas de convolução deformáveis, que são projetadas para melhorar a capacidade do modelo de detectar objetos com diversas escalas e proporções;
- *precision:* métrica que apresenta a qualidade das detecções, ou seja, qual o percentual de detecções está correto;
- *recall:* analisa a proporção de objetos que são de fato placas, com todas as classificações realizadas e classificadas.

2.5 Super-resolução de Imagem

Em áreas como medicina, biologia, automação industrial, vigilância, sensoriamento remoto, existe uma grande demanda de imagens com alta resolução (S. CHAUDHURI, 2001) (W. SHI, 2013) (W. W. ZOU, 2012). Com essas imagens tem-se uma análise e visualização dos detalhes presentes nos dados mais precisas. Apesar de existirem sensores comerciais de alta resolução, muitas vezes o custo dessas imagens se torna proibitivo para o uso em algumas aplicações, sobretudo devido ao custo.

Nos últimos anos, houve um aumento significativo no interesse por técnicas que utilizam super-resolução de imagem (R. TIMOFTE, 2013) (N. A. V. BEÇA, 2008) (S. SCHULTER, 2015) (J.-B. HUANG, 2015) (J. KIM, 2016) (ZHANG, 2018). Em 2018, Zhang apresentou em seu artigo um modelo conhecido como *residual dense network (RDN)*, alcançando resultados de desempenho superior às construções disponíveis para a tarefa de super-resolução de imagens. Com base no princípio dos blocos residuais introduzidos em (HE et al., 2016), o *RDN* procura criar um bloco residual denso (RDB) para extrair um grande conjunto de características locais por meio de camadas convolutivas densas. Ao fundir essas características locais, o modelo pode aprender novos atributos relevantes e garantir a estabilidade durante o treinamento. Os conceitos mencionados são ilustrados na Figura 8.

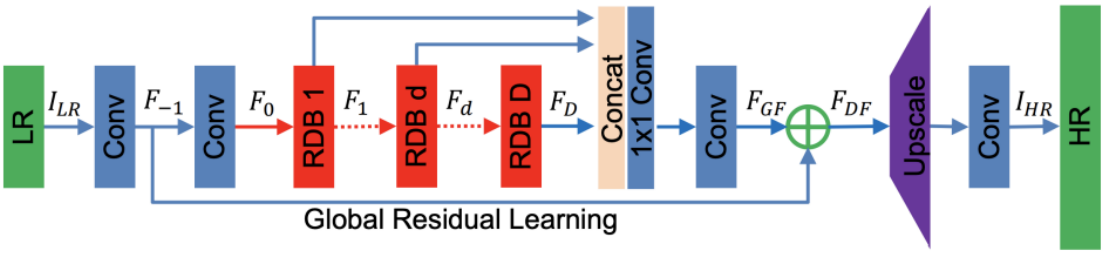
Figura 8 – Arquitetura do RDB.



Fonte: (ZHANG, 2018).

A *RDN* possui uma arquitetura em quatro partes, sendo elas a rede para extração de características rasas (SFENet), blocos RDB, fusão das características densas (DFF) e por fim rede de aumento de amostragem (UPNet), como apresentado na Figura 9.

Figura 9 – Arquitetura do RDN.



Fonte: (ZHANG, 2018).

O Bloco Residual Denso (RDB) desempenha o papel fundamental como módulo básico de construção. Dentro de cada RDB, as conexões densas entre cada camada possibilitam a utilização completa das camadas locais. A Fusão Local de Características (FLC) não apenas estabiliza o treinamento em uma rede mais ampla, mas também controla adaptativamente a preservação das informações dos RDBs atuais e anteriores (ZHANG, 2018).

Adicionalmente, o RDB possibilita conexões diretas entre o RDB anterior e com cada camada do bloco atual, resultando em um mecanismo de memória contínua. Com a Aprendizagem Local de Resíduos (ALR) se aprimora ainda mais o fluxo de informações e gradientes (ZHANG, 2018). Além disso, introduz a Fusão Global de Características (FGC) para extrair características hierárquicas no espaço de baixa resolução. Ao utilizar de forma completa características locais e globais, o RDN realiza uma fusão densa de características e supervisão profunda (ZHANG, 2018). Avaliações extensivas em *benchmarks* demonstram claramente que a *RDN* supera os métodos mais avançados atualmente disponíveis.

2.6 Open Computer Vision Library

OpenCV é uma biblioteca de software de código aberto de visão computacional. Originalmente desenvolvido pela Intel Research em 1999, mais tarde foi suportado pela Willow Garage e depois pela Itseez, sendo agora mantida por toda uma comunidade de desenvolvedores, com suporte a linguagens tais como C, C++,

Python, Java, dentre outras. A biblioteca é multiplataforma e gratuito para uso sob a licença Apache 2. A estruturada principal é composta por:

- Funcionalidades essenciais;
- Processamento de Imagem;
- Leitura e Escrita de Arquivos de Imagem;
- Entrada e Saída de Vídeo;
- Interface Gráfica de Alto Nível;
- Análise de Vídeo;
- Calibração de Câmera e Reconstrução 3D;
- Estrutura para Recursos 2D;
- Detecção de Objetos;
- Módulo de Redes Neurais Profundas;
- Aprendizado de Máquina;
- Clusterização e Busca em Espaços Multidimensionais;
- Fotografia Computacional;
- Montagem de Imagens;
- API de Gráficos.

O OpenCV ajuda o computador a compreender o conteúdo da imagem digital, como fotografias e vídeos, por meio de uma variedade de filtros de imagem que podem ser aplicados para realizar diversas operações de processamento de imagem. Esses filtros são amplamente utilizados em tarefas como redução de ruído, realce de bordas, suavização e outras técnicas de manipulação de imagem. Abaixo exemplos de filtros disponível no OpenCV:

- Filtro bilateral: Suavizar a imagem, mantendo as bordas bem definidas;
- Filtro de Thresholding: Converte a imagem em uma imagem binária com base em um limiar;
- Filtro de Histograma de Equalização: Melhora o contraste da imagem;
- Filtro de Canny: Detecta bordas na imagem usando o algoritmo de Canny;
- Filtro de Suavização: Reduz o ruído e os detalhes finos na imagem;
- Filtro Gaussiano: Semelhante ao filtro de suavização, mas com uma distribuição gaussiana para um efeito mais suave.

Além dos filtros no OpenCV tem-se as operações morfológicas, são uma classe de operações não lineares que nos permitem remover ruído de uma imagem e extrair a forma ou a estrutura da imagem. Elas são frequentemente usadas em visão computacional para pré-processamento de imagem, geralmente essas transformações são aplicadas em imagens em escala de cinza ou binárias. Algumas das operações morfológicas mais comuns no OpenCV incluem:

- Erosão: Usada para encolher áreas brancas em uma imagem binária. Faz isso corroendo as bordas das regiões brancas;
- Dilatação: Usada para expandir áreas brancas em uma imagem binária. Faz isso dilatando as bordas das regiões brancas;
- Abertura: Uma sequência de uma erosão seguida por uma dilatação. É útil para remover ruídos brancos pequenos em uma imagem binária;
- Fechamento: Uma sequência de uma dilatação seguida por uma erosão. É útil para preencher pequenos buracos pretos em uma imagem binária.

2.7 Tesseract OCR

O Tesseract OCR é uma biblioteca de código aberto para reconhecimento óptico de caracteres, desenvolvido originalmente pelos laboratórios Hewlett-Packard entre 1984 e 1994. Em seguida o Google adquiriu o Tesseract transformando o projeto em código aberto no ano de 2005 (SMITH, 2007). Desde então, o Tesseract OCR tem sido constantemente aprimorado pela comunidade de desenvolvedores em todo o mundo, dentre esses aprimoramentos a biblioteca originalmente para a linguagem C++ foi modificada, permitindo o funcionamento da mesma em programação Python. Por tal motivo, o nome desta biblioteca para a linguagem Python tornou-se PyTesseract.
Sua arquitetura tem como entrada, imagens binárias com regiões definidas de textos poligonais opcionais. O primeiro passo é a análise de componentes conectados em que esboços do mesmo serão guardados, assim é possível detectar textos invertidos e os reconhecer facilmente como um texto preto no branco. Nessa fase, os esboços são agrupados, por enquadramento, em “*Blobs*” (SMITH, 2007). “*Blobs*" refere-se a regiões ou áreas conectadas de uma imagem que compartilham características semelhantes, como cor, intensidade ou textura.

Os “*Blobs*” são organizados em linhas de texto, e as linhas e regiões são analisadas para texto de espaçamento fixo ou proporcional. As linhas de texto são

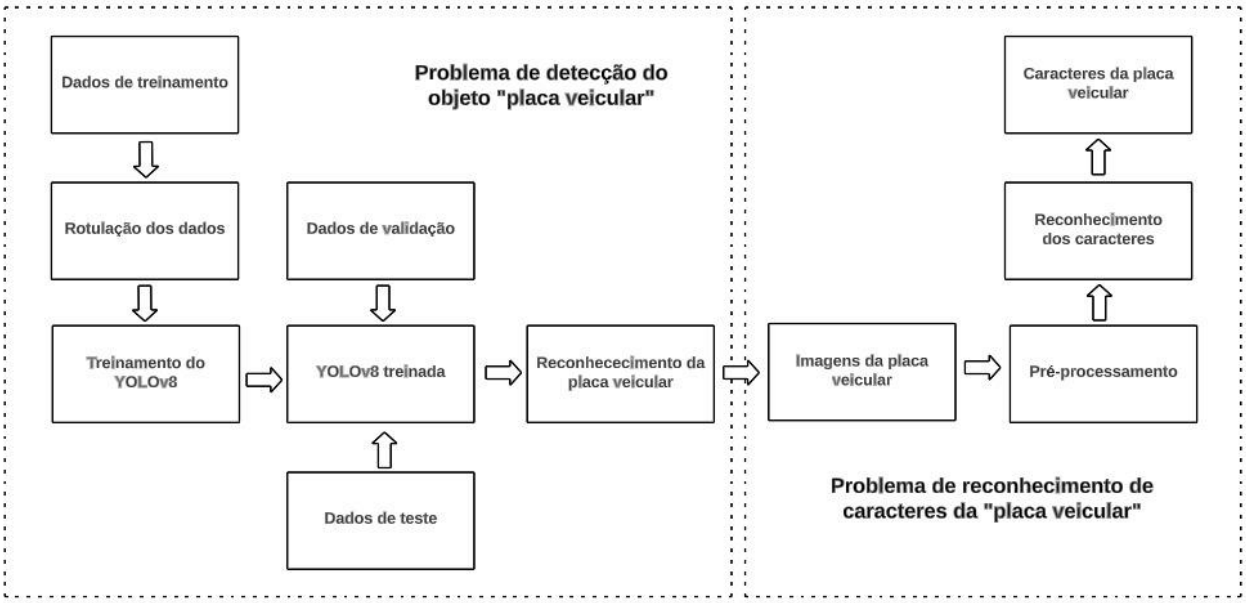
quebradas em palavras através das diferenças de espaçamento de caracteres. Textos de argumentos definidos são então separados em caracteres. O texto proporcional é separado em palavras utilizando espaços definidos e espaços incertos. O reconhecimento passa por dois passos, no primeiro as palavras são reconhecidas em cada turno. Cada palavra, considerada satisfatória, é então passada para um classificador adaptativo como dados treinados. Assim, o classificador adaptativo tem maiores chances de reconhecer melhor textos ao decorrer da página. Como o classificador adaptativo pode ter aprendido algo útil tarde demais para contribuir perto do topo da página, é executada uma segunda passagem sobre a página, na qual palavras que não foram reconhecidas com precisão o suficiente são reconhecidas novamente. Tem-se como a fase final verificar espaços incertos e checar hipóteses alternativas para localizar textos em minúsculo (SMITH, 2007).

No momento em que este texto foi escrito, o Tesseract 5.3.2 era a versão mais recente. A partir da versão 4.0.0, o Tesseract usa arquitetura baseada em LSTM, que fornece uma solução para o problema do gradiente de fuga que pode ocorrer.

3. METODOLOGIA

O desenvolvimento do algoritmo se dá por meio do editor de código-fonte Visual Studio Code na linguagem Python, o detector de objetos *YOLO*v8 da Ultralytics é utilizado para detectar o objeto placa veicular, o *RDN* e a biblioteca OpenCV realiza o processo de pré-processamento de imagem, o Tesseract OCR realiza a extração dos caracteres. O repositório com o algoritmo e dados se encontra no GitHub uma plataforma de hospedagem de código-fonte e arquivos (RAFAEL ENZIO, 2023). Para apresentação do processo realizado ao longo do trabalho é apresentado a Figura 10.

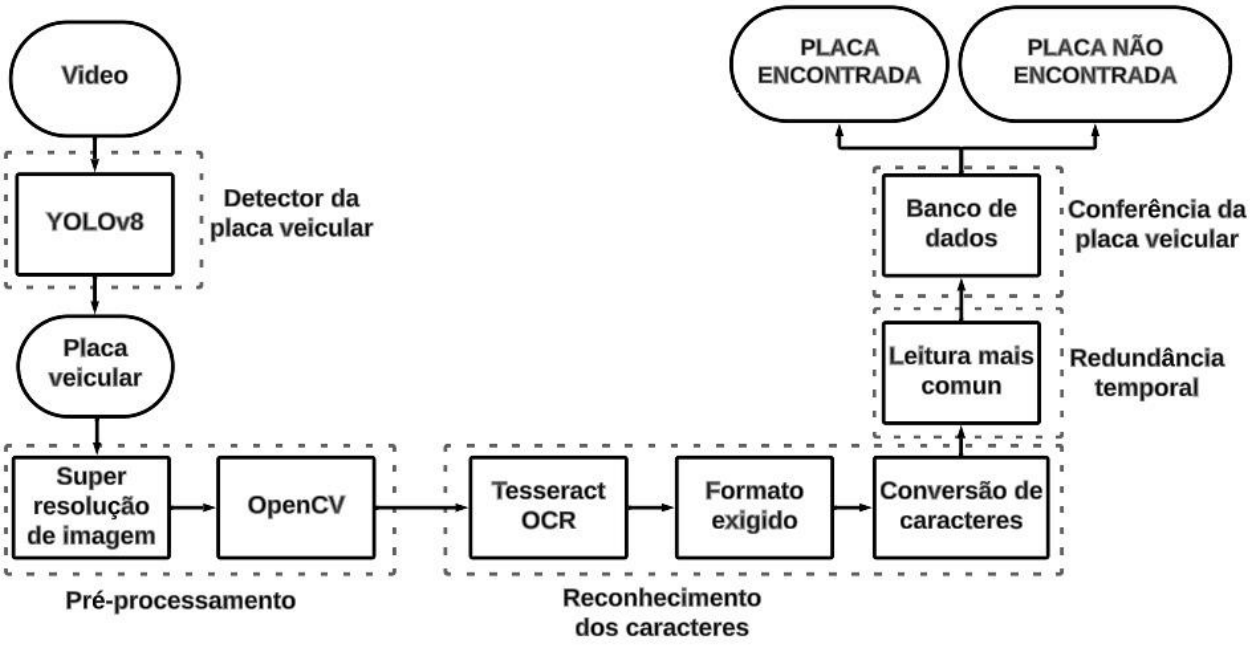
Figura 10 – Fluxograma do processo desenvolvido.



Fonte: Autor.

A Figura 10 apresenta o fluxo das atividades desenvolvidas para resolver os objetivos específicos do trabalho, que se divide em dois principais problemas detecção do objeto placa veicular e reconhecimento de caracteres da placa veicular. Na Figura 11 é apresentado a solução baseada em visão computacional para o objetivo geral.

Figura 11 – Fluxograma da solução baseada em visão computacional.



Fonte: Autor.

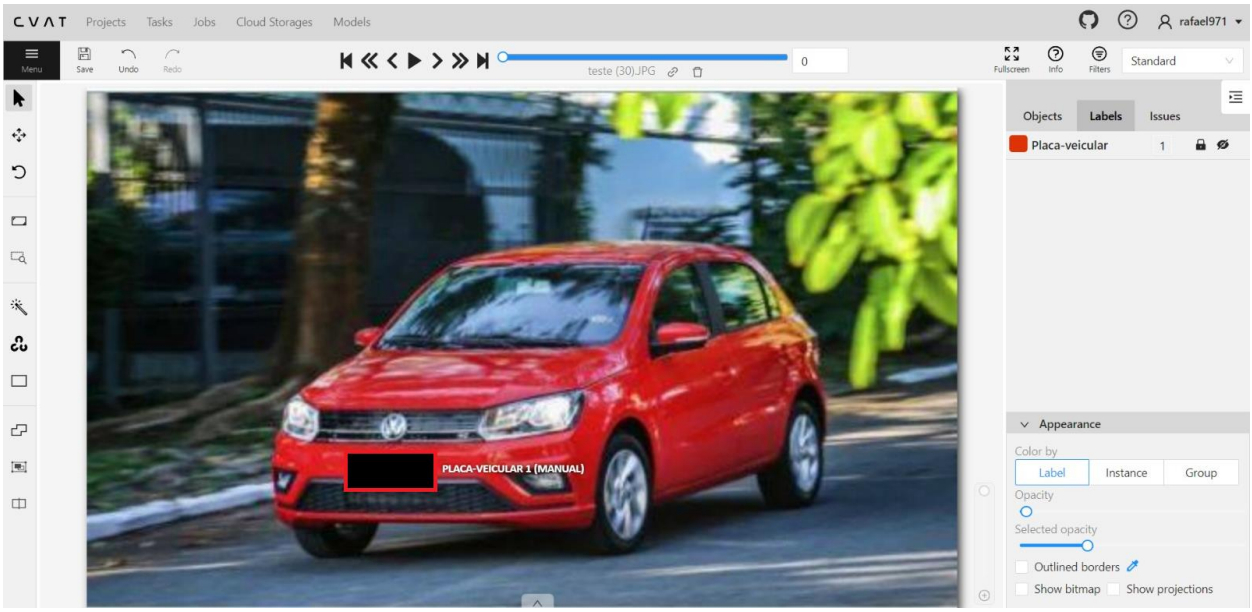
Tem como entrada um vídeo que irá passar pelo detector da placa veicular e retorna imagens do objeto Placa veicular, em seguida as imagens tem um pré-processamento que prepara as imagens para o reconhecimento dos caracteres, aplica-se o processo de redundância temporal no resultado do Tesseract OCR, que por fim é conferido retornando a saída PLACA ENCONTRADA ou PLACA NÃO ENCONTRADA.

3.1 Dados

3.1.1 Dados de treinamento

As imagens utilizadas para treinamento fazem parte do Banco de Dados Sense-ALPR (GABRIEL RESENDE GONÇALVES, 2018), sendo utilizado 500 imagens deste banco de dados, com todas as imagens utilizadas sendo do período diurno, em que existe uma melhor iluminação e possibilita maior clareza da placa veicular. Os dados precisam passar pelo processo de rotulação, que consiste em associar rótulos da classe “*license-plate*” ao objeto placa veicular por meio de caixas delimitadoras, como demostrado na Figura 12.

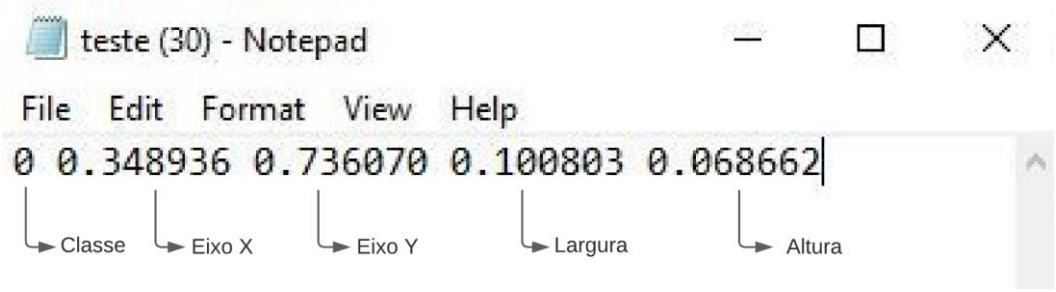
Figura 12 – Rotulação da imagem de treinamento.



Fonte: Autor.

A rotulação gera um arquivo de texto apresentado na Figura 13, com a classe que especifica o tipo de objeto, as coordenadas eixo X e Y do centro do retângulo, por fim a altura e largura do retângulo.

Figura 13 – Coordenadas da rotulação da Figura 12.



Fonte: Autor.

Ferramentas específicas de rotulação, como *LabelImg*, *VGG Image Annotator*, *COCO Annotator*, entre outras, são frequentemente utilizadas para simplificar esse processo, a ferramenta utilizada em questão foi o CVAT(CVAT.ai).

3.1.2 Dados de teste

Os dados de teste são constituídos por um conjunto de 34 imagens de própria autoria, com o objetivo de testar o detector de objetos *YOLOv8* após o treinamento e verificar sua performance em identificar a placa veicular em diferentes situações, como placa frontal, traseira, placa em diferentes ângulos, múltiplas placas, placas próximas, distantes, imagens com boa resolução, baixa resolução, durante a noite, placa de outro país, placa comercial, modelo antigo de placa e modelo MERCOSUL.

3.1.3 Dados de validação

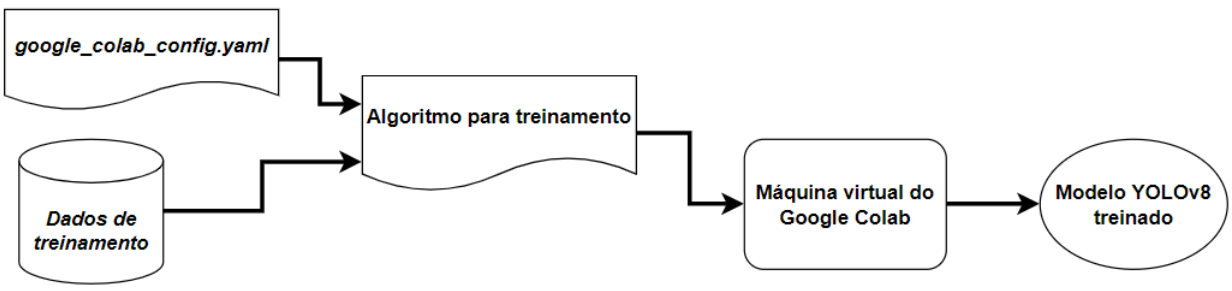
Os dados de validação são constituídos por vídeos dos veículos com a placa do MERCOSUL, com uma média de duração de 4 segundos, com 19 vídeos durante o dia e dois durante o período noturno. Coletadas pelo próprio autor, por meio da câmera de smartphone, no cenário de entrada de estacionamento. As coletas foram feitas nesse cenário devido aos veículos trafegarem em uma menor velocidade e se ter uma visão frontal dos veículos, fornecendo com isso uma visualização da placa veicular de forma mais nítida.

3.2 Detector de objetos *YOLO*v8

3.2.1 Treinamento personalizado do modelo *YOLO*v8

Na Figura 14 é apresentado o fluxograma com o procedimento para treinamento do modelo *YOLOv8*.

Figura 14 – Procedimento para treinamento do modelo YOLOv8.



Fonte: Autor.

Para realizar os treinamentos, inicialmente é preciso colocar os Dados de treinamento mencionados na Seção 3.1.2, em uma pasta chamada *data*, logo após, é criado o arquivo *google_colab_config.yaml* com as informações de caminhos e número de classes. No Algoritmo para treinamento, são definidos quatro parâmetros do modelo *YOLOv8*:

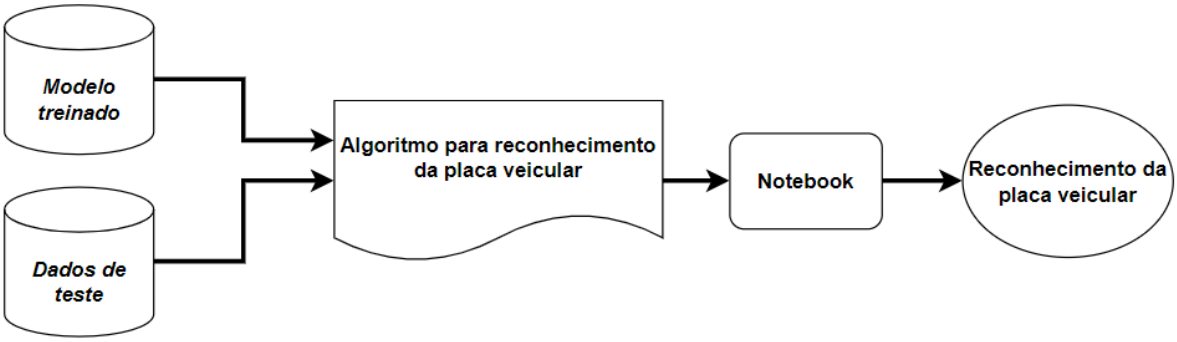
- A quantidade de épocas em 250. Uma época é uma interação completa do conjunto de dados de treinamento por meio da rede neural durante o treinamento;
- Define-se como 50 a quantidade de épocas para esperar por nenhuma melhoria observável, para que seja feita a interrupção antecipada do treinamento;
- O número de imagens que será percorrido em paralelo em 64;
- Habilita-se o ponto de verificação a cada vez que os dados são percorridos.

O processo de treinamento do *YOLOv8* para identificar o objeto placa veicular se dá por meio de uma máquina virtual do Google Colab, pois este fornece uma potente unidade de processamento gráfico, que é eficiente para as operações matriciais intensivas envolvidas no treinamento de redes neurais, permitindo o processamento de várias tarefas simultaneamente, acelerando significativamente o treinamento. Com o treinamento finalizado, tem-se como resultado o arquivo "*best.pt*" que possui os parâmetros ajustáveis ou coeficientes que a rede aprendeu durante o processo de treinamento, sendo essencial para o funcionamento da rede.

3.2.2 Teste do modelo *YOLO*v8

Na Figura 15 é apresentado o fluxograma com o procedimento para teste do modelo *YOLOv8*.

Figura 15 – Procedimento para teste do modelo YOLOv8.



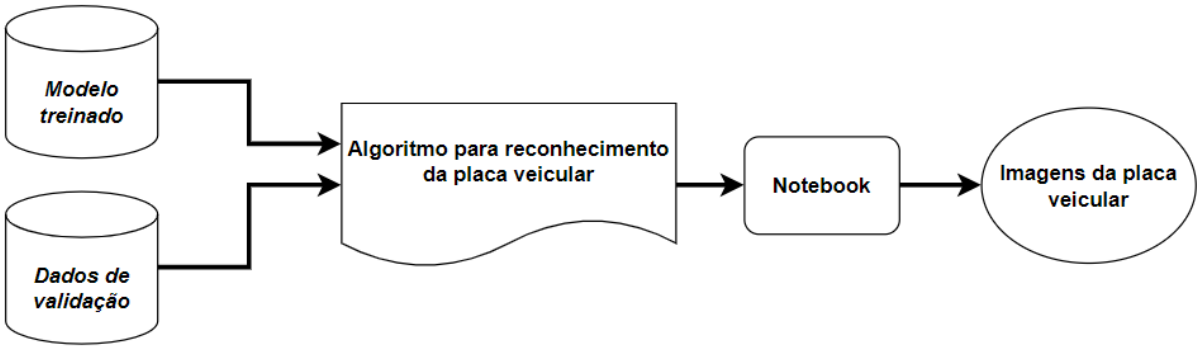
Fonte: Autor.

O Modelo treinado corresponde ao resultado do treinamento realizado na Seção 3.2.1, que é usado pelo Algoritmo para reconhecimento da placa veicular, no processo de análise dos Dados de teste apresentados na Seção 3.1.2. Optou-se por realizar o processamento do algoritmo no Notebook, uma vez que nesta etapa não é necessária uma potente unidade de processamento gráfico. No algoritmo para reconhecimento da placa veicular tem-se o parâmetro de limite de confiança que irá determinar o valor de corte para se ter um reconhecimento da placa veicular. Com o procedimento definido inicia-se o processo de testar como o modelo treinado se comporta ao identificar as placas veiculares dos Dados de teste e como o limite de confiança modifica os números de placas reconhecidas.

3.2.3 Validação do modelo *YOLOv8*

O processo de validação do modelo *YOLOv8* e semelhante ao realizado na Seção 3.2.2, com mudança no tipo de dado que será analisado, agora recebe os dados de validação da Seção 3.1.3. Para exemplificar o funcionamento do sistema de validação e apresentado a Figura 18 com o procedimento para validação do modelo *YOLOv8*.

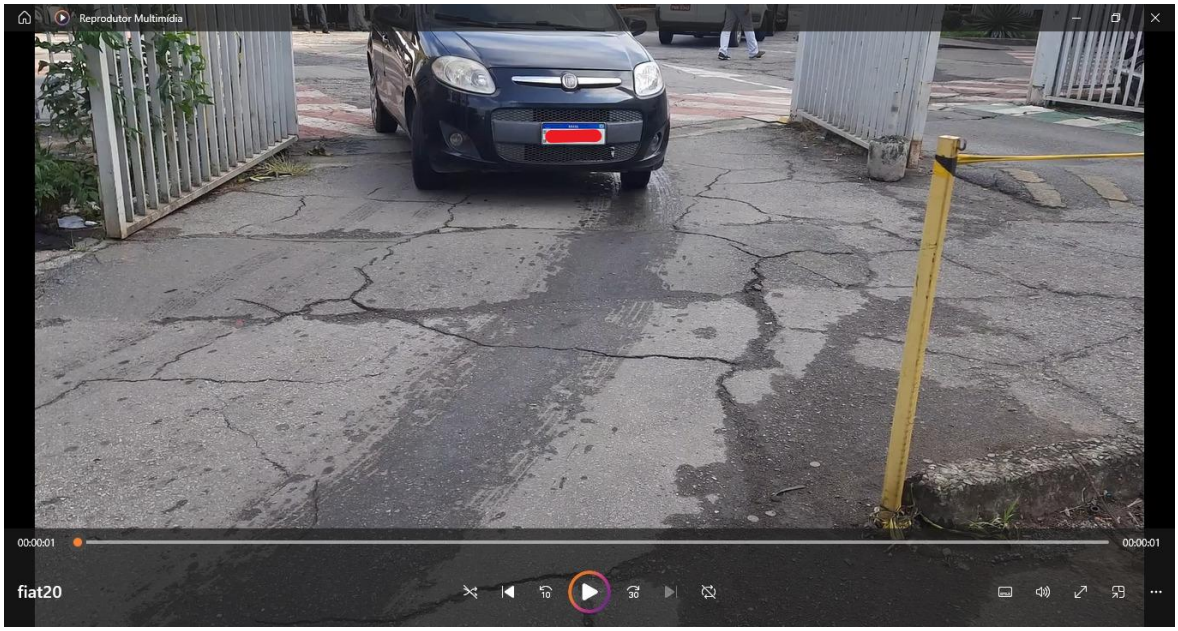
Figura 16 – Procedimento para validação do modelo YOLOv8.



Fonte: Autor.

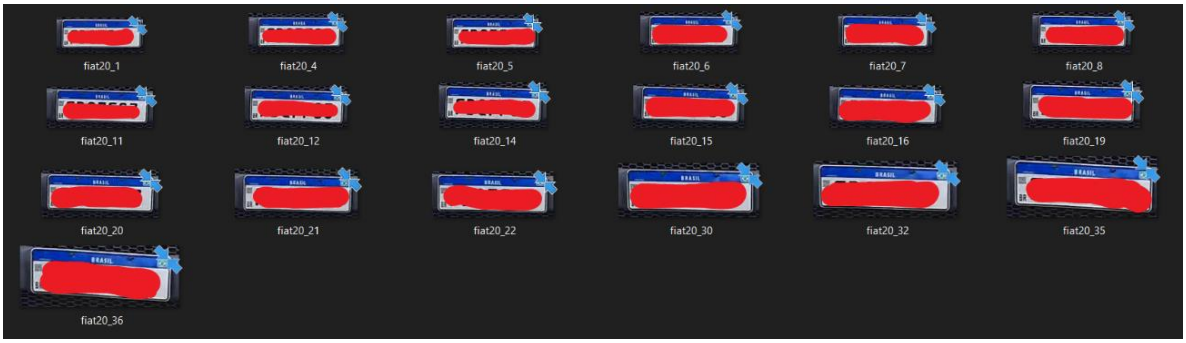
O vídeo apresentado na Figura 17 é um exemplo de dado de validação, que é analisado frame a frame pelo modelo *YOLOv8*, de forma a buscar o objeto placa veicular com limite de confiança igual ou superior a 0,85 em cada frame do vídeo. Quando encontrado o objeto placa veicular em algum dos frames, será salvo somente a região do frame em que modelo identificou a presença da placa veicular, como apresentado na Figura 18, que traz o resultado do modelo *YOLOv8* em reconhecer a placar veicular do vídeo fiat20.

Figura 17 – Captura do início do vídeo fiat20.



Fonte: Autor.

Figura 18 – Imagens da placa veicular do vídeo fiat20.



Fonte: Autor.

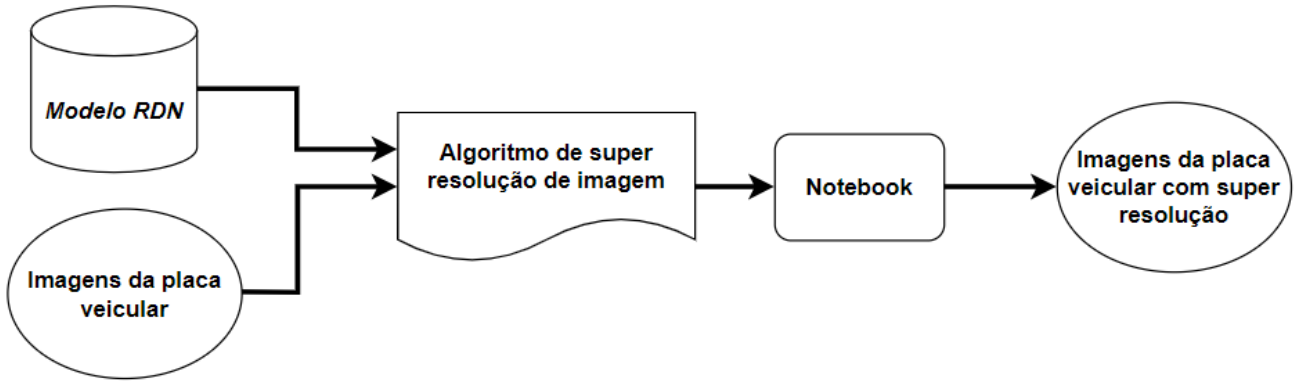
3.3 Pré-processamento da imagem

Nessa etapa são utilizadas algumas técnicas para o tratamento das imagens, para melhorar os dados da imagem, suprimindo distorções indesejadas ou aprimorando alguns recursos da imagem relevantes para tarefa de leitura da placa.

3.3.1 Super-resolução de imagem

Com uso do Modelo *RDN* e das imagens obtidas como resultado na Seção 3.2.3, executa-se o Algoritmo de super-resolução de imagem no Notebook, como resultado do procedimento tem-se as imagens da placa veicular com super-resolução. O procedimento descrito é apresentado na Figura 19.

Figura 19 – Procedimento da super-resolução de imagem.

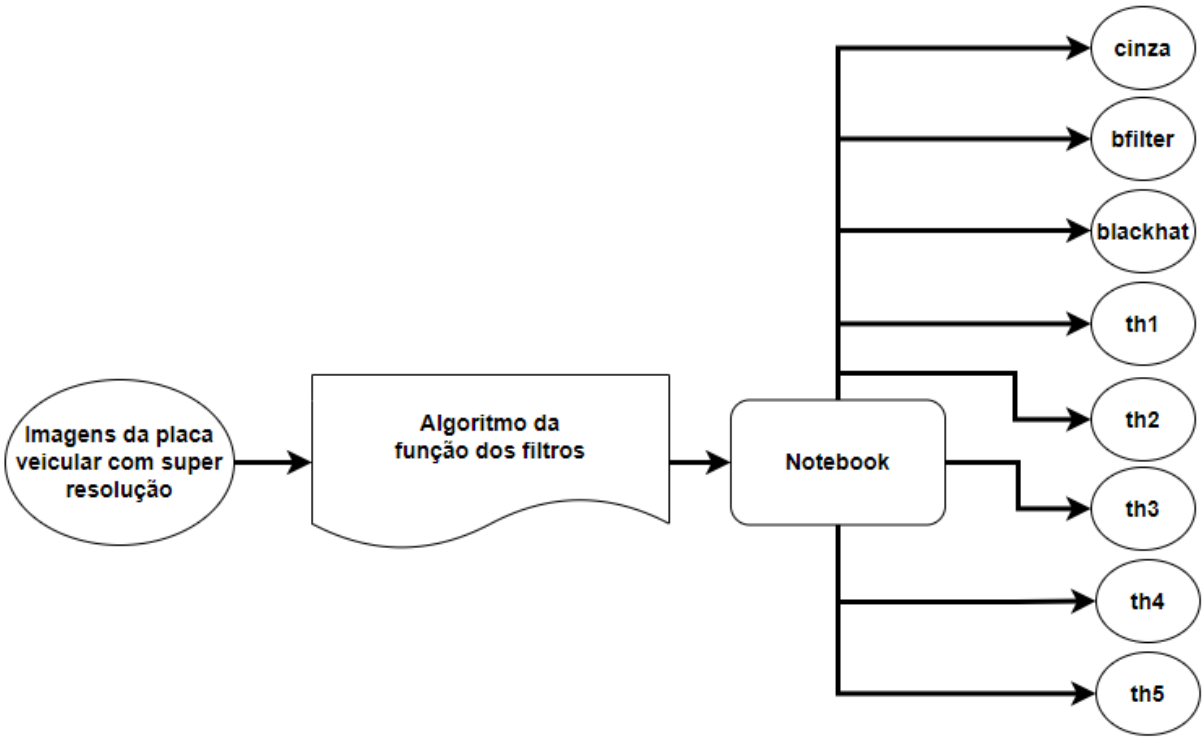


Fonte: Autor.

3.3.2 Filtros do OpenCV utilizados

O procedimento que realiza a filtragem das Imagens da placa veicular com super-resolução da Seção 3.3.1 é apresentado na Figura 20.

Figura 20 – Procedimento do algoritmo da função dos filtros.



Fonte: Autor.

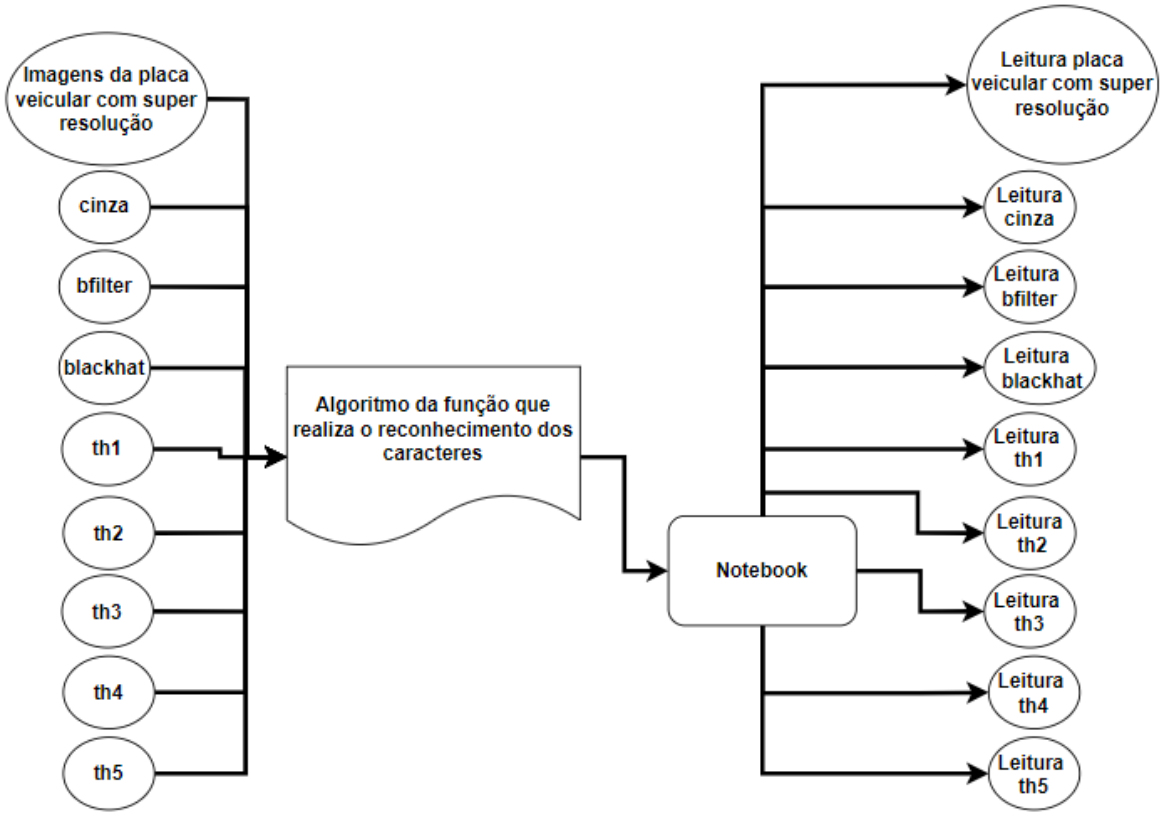
Executa-se o Algoritmo da função dos filtros no Notebook, tendo como resultado do procedimento oito novas imagens, nomeadas de forma correspondente ao filtro que as gerou, filtro cinza, bilateral, *blackhat*, th1, th2, th3, th4 e th5.

3.4 Leitura da placa veicular

3.4.1 Reconhecimento dos caracteres

Na Figura 21 é apresentado o procedimento para realizar o reconhecimento dos caracteres, que tem como entrada a Imagem da placa veicular com super-resolução, cinza, bfilter, blackhat, th1, th2, th3, th4 e th5.

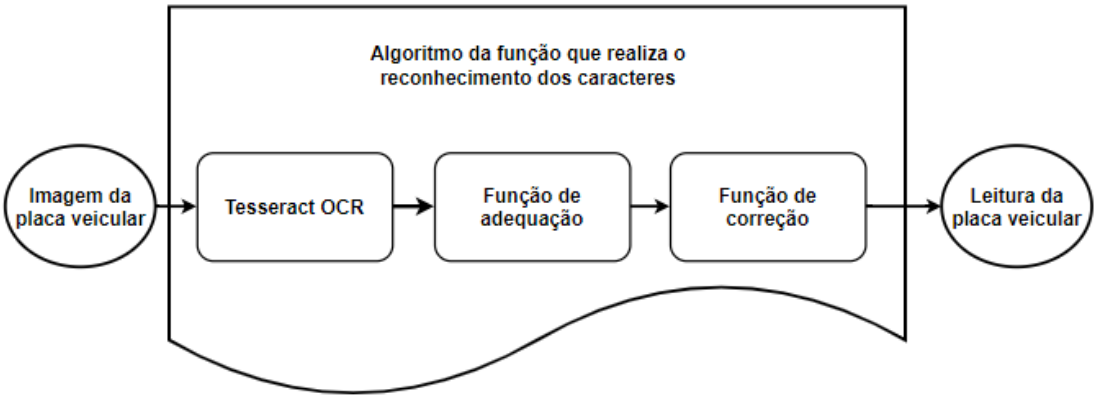
Figura 21 – Procedimento para realizar o reconhecimento dos caracteres.



Fonte: Autor.

O Algoritmo da função que realiza o reconhecimento dos caracteres é executado no Notebook, tendo como resultado até nove leituras. Dentro do Algoritmo da função que realiza o reconhecimento dos caracteres, têm-se três procedimentos que são apresentados na Figura 22.

Figura 22 – Procedimentos realizados dentro da função que realiza o reconhecimento dos caracteres.



Fonte: Autor.

No Tesseract OCR ocorre o reconhecimento óptico de caracteres, define-se os parâmetros da ferramenta em segmentação como “psm 9”, modo motor como “oem 1” e uma lista de caracteres permitidos. Com o reconhecimento óptico de caracteres feito, tem-se em seguida a Função de adequação, responsável por conferir se foi extraído exatamente 7 caracteres e definir que tipo de informação se espera em cada posição. A posição de cada caractere é estabelecida na estrutura do modelo da placa MERCOSUL:

1. Letra
2. Letra
3. Letra
4. Número
5. Letra
6. Número
7. Número

Por fim, a Função de correção que atua modificando valores de um caractere de número para letra ou letra para número, com base no que foi definido na função de adequação. A condição para que isso aconteça é que o caractere esteja presente na Tabela 1.

Tabela 1 – Mapeamento para conversão de caracteres.

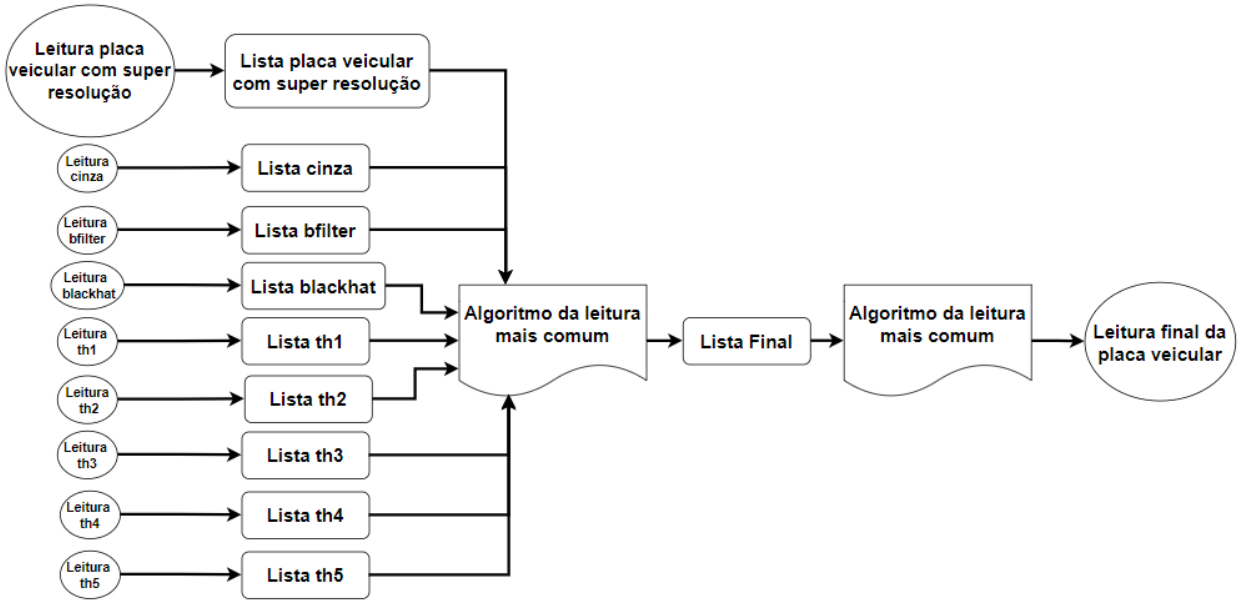
Letra	Número
O	**0**
I	**1**
J	**3**
A	**4**
G	**6**
S	**5**

Fonte: Autor.

3.4.2 Definição da leitura da placa

Aa leituras da Seção 3.4.1 são armazenadas cada uma em uma lista correspondente ao seu nome, como pode ser observado na Figura 23.

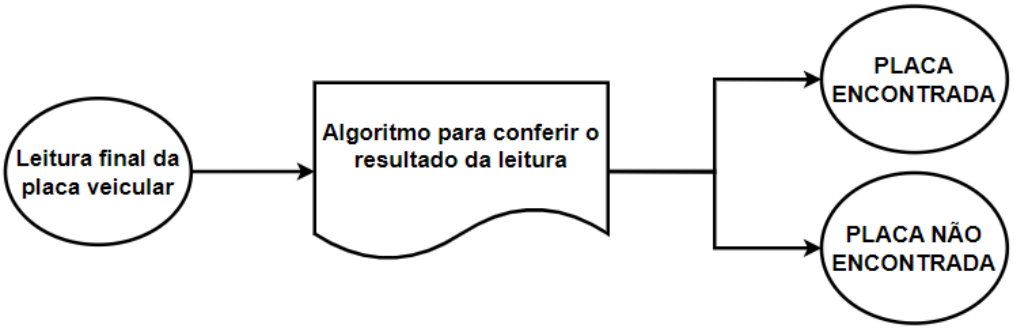
Figura 23 – Procedimento para se obter a leitura final da placa veicular.



Fonte: Autor.

Após feito o reconhecimento de caracteres de todas as imagens é preciso definir uma leitura que irá representar a leitura da placa para cada lista apresentada na Figura 23, para isso é usado o Algoritmo da leitura mais comum. Com uso do Algoritmo da leitura mais comum, se encontra a leitura mais comum em cada uma das listas da Figura 23, essa leitura é adicionada na Lista Final. Novamente o algoritmo da leitura mais comum é usada, porém na Lista Final para se obter a Leitura final da placa veicular. Para conferir o resultado da Leitura final da placa veicular tem-se o procedimento apresentado na Figura 24. Onde por meio do algoritmo para conferir o resultado da leitura, tem-se dois possíveis resultados PLACA ENCONTRADA ou PLACA NÃO ENCONTRADA.

Figura 24 – Procedimento para conferir o resultado da leitura.



Fonte: Autor.

3.6 Recursos

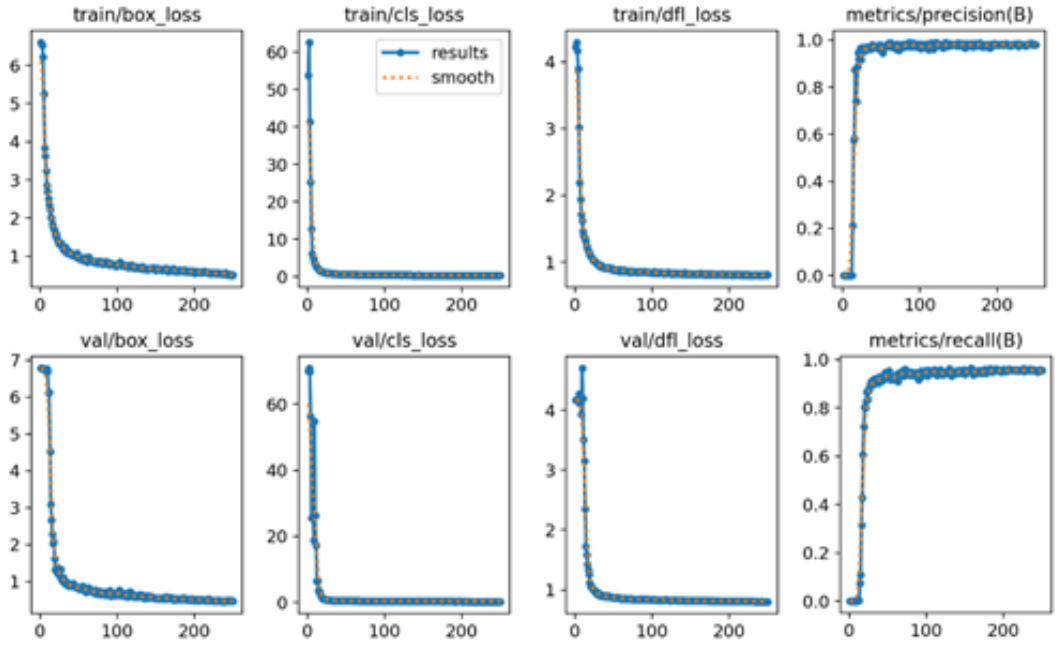
Para o desenvolvimento do projeto, foi utilizado um notebook com processador Intel (R). Core (TM) i5-3230M CPU @ 2.60GHz, 8.00 GB de memória RAM, com uma placa de vídeo NVIDIA GeForce GT 650M e o Google Colab. Como meio de adquirir o vídeo foi utilizado a câmera de um smartphone, modelo Samsung Galaxy M31, com uma câmera de 64MP e com taxa de aquisição de até 30 frames por segundos.

4. RESULTADOS

4.1 Resultados do treinamento personalizado do modelo *YOLO*v8

Os resultados obtidos com o procedimento da Seção 3.2.1 é observado respectivamente na Figuras 25. O eixo horizontal do gráfico indica o número de épocas, enquanto o eixo vertical representa o valor da perda, com exceção dos gráficos *metrics/precision* e *metrics/recall*.

Figura 25 – Resultados do treinamento do modelo *YOLO*v8.



Fonte: Autor.

Avaliando os gráficos de métrica de perda *train/box_loss, train/cls_loss* e *train/dfl_loss* fica evidente que não ocorre *underfitting*, uma vez que o modelo apresentou valores próximo a 0, também não ocorre *overfitting*, pois o mesmo resultado ocorre nos gráficos de *val/box_loss*, *val/cls_loss* e *val/dfl_loss*. Com isso temos o modelo se adaptando bem tanto aos dados de treinamento quanto de validação.

Observando os gráficos *metrics/precision* e *metrics/recall* podemos constatar a qualidade do modelo, as métricas no eixo vertical variam no intervalo entre 0 e 1, ou seja, entre 0% e 100%. Dessa forma, verifica-se que o modelo obteve valores acima da faixa de 90%, indicando uma boa confiabilidade do modelo em identificar placas de automóveis. Mais informações sobre os gráficos da Figura 25 estão disponíveis na Seção 2.4.

Figura 26 – Resultado da rotulação inicial e rotulação final.

Fonte: Autor.

Durante o processo de treinamento do modelo YOLOv8, foi observado como a rotulação dos dados da Seção 3.1.1, utilizados para treinamento da rede neural afetam os resultados das detecções da placa veicular. Como apresentado na Figura 26, a Rotulação 1 feita inicialmente gerou um modelo que não consegue identificar a placa veicular próxima, como visto no Resultado da Rotulação 1, isso ocorre por ter sido feito a rotulação de um maior número de placas mais distantes, o que fez com que o modelo fosse melhor em identificar o objeto placar veicular nestas circunstâncias.

Na Rotulação 2 o processo de rotulação foi feito focando somente nas placas onde é possível visualizar os caracteres, com isso tem se um modelo que identifica corretamente placas próximas e distantes como visto no Resultado da Rotulação 2. Com isso em mente faz se necessário a rotulação precisa e consistente para o treinamento eficaz do modelo de detecção de objetos.

4.2 Resultado do teste do modelo *YOLO*v8

Os resultados ao utilizar os dados de teste da Seção 3.1.2 no modelo treinado é apresentado na Tabela 2 e Tabela 3, cada tabela possui um limite de confiança, que irá determinar o valor de corte para se ter um reconhecimento de placa veicular válido. Chegou-se aos valores 0,40 e 0,85 por meio de tentativa e erro, ao se busca

por valores que apresentassem um bom equilíbrio entre os parâmetros placas reconhecidas, falso negativo e falso positivo.

Tabela 2 – Dados de teste com limite de confiança em 0,40.

Número de placas	Placas reconhecidas	Falso negativo	Falso positivo
69	52	17	5

Fonte: Autor.

Tabela 3 – Dados de teste com limite de confiança em 0,85.

Número de placas	Placas reconhecidas	Falso negativo	Falso positivo
69	24	45	0

Fonte: Autor.

Observa-se que com o parâmetro de limite de confiança em 0,40, se consegue um maior número de placas reconhecidas e por consequência menor número de falso negativo, que é quando existe uma placa veicular e o modelo não consegue identificar, mas em paralelo ocasiona falso positivo, de forma a classificar outros objetos na imagem como a placa veicular. Diante desse resultado optou-se por trabalhar com um limite de confiança de 0,85 para o procedimento da Seção 3.2.3, uma vez que o foco nesse processo não é ter um grande número de placas reconhecidas, mas placas com uma boa visualização dos caracteres, que ocorre geralmente com um limite de confiança próximo a 0,85 e sem falso positivo que iria gerar dados desnecessários para serem processados.

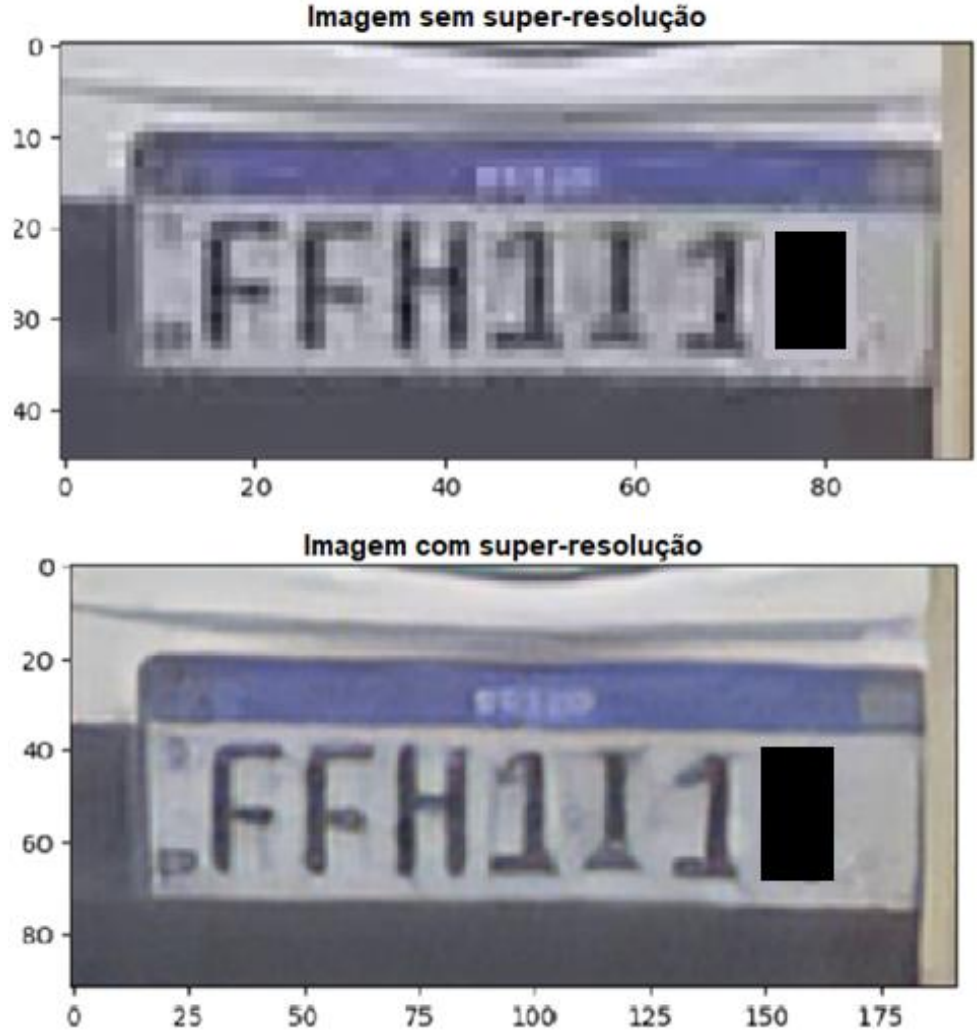
4.3 Resultado da validação do modelo *YOLO*v8

O resultado do procedimento da Seção 3.2.3 obteve êxito em encontrar a placa veicular em todos os dados de validação da Seção 3.1.3, produzindo com isso 21 pastas com as imagens do reconhecimento da placa veicular correspondente a cada vídeo, gerando um total de 893 detecções do objeto placa veicular. Observa-se que o limite de confiança em 0,85 para os dados de validação foi um acerto, visto que se consegue com ele diminuir os números de detecções do objeto placa veicular e não ter a situação de falso negativo, que tende a ocorrer ao usar um limite de confiança mais alto, situação que pode ser observada na Tabela 3 da Seção 4.2.

4.4 Resultado do pré-processamento da imagem

O resultado do procedimento de super-resolução de imagem é demostrado por meio do comparativo das imagens da Figura 27, onde nota-se a melhora na visibilidade e a legibilidade da placa. O procedimento de super-resolução de imagem permite uma ampliação sem perda significativa de qualidade e ajuda a reduzir impacto de ruídos.

Figura 27 – Imagem sem super-resolução x Imagem com super-resolução.

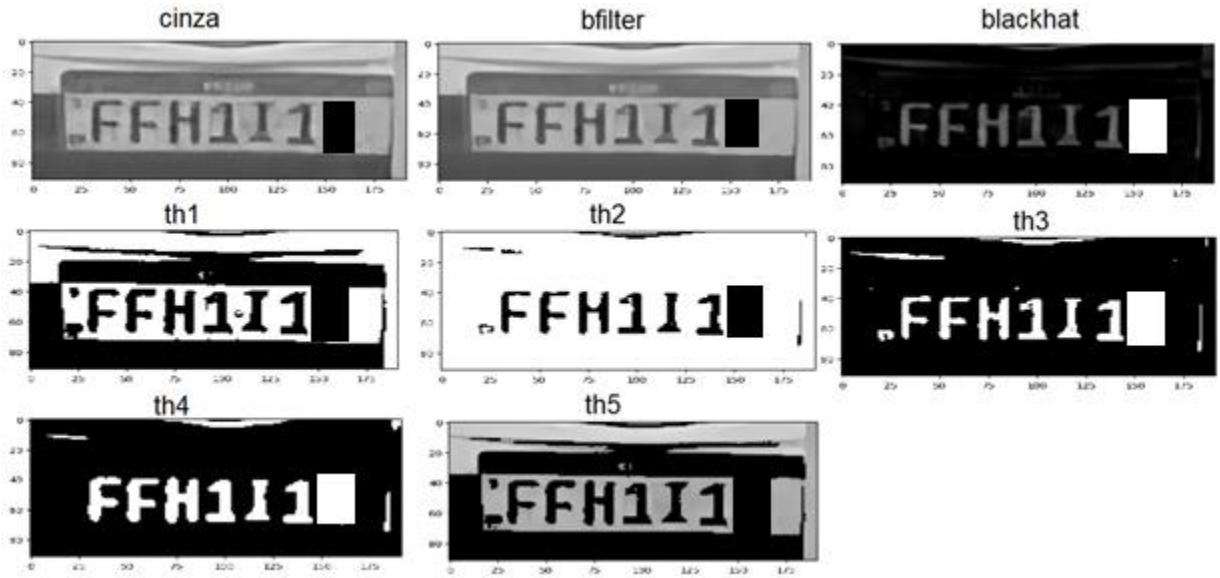


Fonte: Autor.

O uso de super-resolução de imagem se apresenta como uma solução para melhorar as imagens capturadas em baixa resolução pelo autor, porém o uso de uma câmera mais apropriada para a captura dos vídeos pode se apresentar como uma melhor solução, sendo necessário um maior estudo sobre o assunto para se chegar a uma resposta.

A Figura 28 apresenta um exemplo do resultado do uso dos filtros da Seção 3.3.2 no pré-processamento da imagem.

Figura 28 – Exemplo de resultado da Seção 3.3.2.



Fonte: Autor.

A conversão da imagem para as escalas em cinza ocorre para possibilitar uma maior velocidade no processamento da imagem. O bfilter é aplicado para reduzir o ruído da imagem enquanto preserva as bordas dos caracteres presente na placa. O *blackhat* é utilizado para realçar a região dos caracteres que é uma região mais escura em relação ao resto da imagem. Os filtros th1, th2, th3, th4 e th5 é usado para simplificar a representação da imagem, onde pixels com valores abaixo de um limiar são considerados pretos e pixels com valores acima do limiar são considerados brancos, com o propósito destacar os caracteres.

4.5 Resultado da leitura da placa veicular

Na Tabela 4 é apresentado os resultados em realizar a leitura da placa dos dados de validação da Seção 3.2.3, o resultado PLACA ENCONTRADA implica que a leitura correspondente àquela placa foi identificada e para os casos onde não foi possível identificar se tem o resultado PLACA NÃO ENCONTRADA.

Tabela 4 – Resultados da leitura dos dados de validação.

Pasta	Placa veicular	Resultado da leitura
Caso-1	FFH1I1[redacted]	PLACA ENCONTRADA
Caso-2	EPE7F9[redacted]	PLACA ENCONTRADA
Caso-3	[redacted]HE5G21	PLACA ENCONTRADA
Caso-4	NYP1E2[redacted]	PLACA ENCONTRADA
Caso-5	SHX6C1[redacted]	PLACA ENCONTRADA
Caso-6	NWE414[redacted]	PLACA ENCONTRADA
Caso-7	OSQ0E0[redacted]	PLACA ENCONTRADA
Caso-8	MUV2E1[redacted]	PLACA ENCONTRADA
Caso-9	OXI2G0[redacted]	PLACA ENCONTRADA
Caso-10	SIU7C8[redacted]	PLACA ENCONTRADA
Caso-11	FAS3A3[redacted]	PLACA ENCONTRADA
Caso-12	SHC0B8[redacted]	PLACA ENCONTRADA
Caso-13	OMA8C9[redacted]	PLACA ENCONTRADA
Caso-14	RUM4F4[redacted]	PLACA ENCONTRADA
Caso-15	FBQ7F0[redacted]	PLACA ENCONTRADA
Caso-16	SHQ0A7[redacted]	PLACA ENCONTRADA
Caso-17	QNO5J8[redacted]	PLACA ENCONTRADA
Caso-18	PUC0G3[redacted]	PLACA ENCONTRADA
Caso-19	QAA1H7[redacted]	PLACA ENCONTRADA
Caso-20	GAC0H8[redacted]	PLACA NÃO ENCONTRADA
Caso-21	RMQ3I9[redacted]	PLACA NÃO ENCONTRADA

Fonte: Autor.

Com base na Tabela 4 temos um percentual de acerto de 90,47% em realizar a leitura correta da placa veicular, só não sendo possível no Caso-20 e Caso-21 em que os vídeos correspondentes foram realizados durante o período noturno, se levanta a hipótese de que em período noturno, por não haver uma boa iluminação na captura do vídeo, tal situação ocorra, sendo necessário uma maior investigação. Como exemplo de demonstração do resultado do reconhecimento de caractere de uma imagem, tem-se as Figuras 29 e 30.

Figura 29 – Exemplo do resultado do reconhecimento de caractere de uma imagem do Caso-3.

Fonte: Autor.

Figura 30 – Exemplo do resultado do reconhecimento de caractere de uma imagem do Caso-21.

Fonte: Autor.

Na Figura 29 é apresentado uma das imagens do Caso-3 e o reconhecimento de caracteres em cada filtro. Tem-se como resultado cinco leituras corretas, três onde não foi possível realizar uma leitura dentro do padrão estabelecido e uma que reconheceu o último caractere de forma errada, trocando um por quatro. Mesmo com os três filtros sem resposta válida e uma leitura errada pode se considerar o procedimento um sucesso, visto que ao buscar a leitura mais comum se terá a leitura correta no final.

Já na Figura 30 é apresentado uma das imagens do Caso-2, que tem como resultado nenhuma leitura correta, com sete leituras fora do padrão estabelecido e duas no padrão, mas com caracteres errados. A falha em questão pode ser facilmente identificada comparando as Figuras 29 e 30, onde em uma existe uma maior clareza dos caracteres da placa veicular do que na outra, o que afeta consideravelmente o Tesseract OCR em realizar o reconhecimento óptico de caracteres.

5. CONCLUSÃO

Este trabalho descreveu a implementação da identificação de placas veiculares, em particular no modelo MERCOSUL, utilizando de técnicas de aprendizado profundo, processamento de imagens e reconhecimento óptico de caracteres. Em realizar a identificação da placa veicular o modelo treinado teve sucesso em todos os dados de validação e no reconhecimento dos caracteres obteve 90,47% de acerto.

Dessa maneira, o objetivo principal do trabalho foi cumprido, quanto aos objetivos específicos, foi alcançado adequadamente todos os objetivos. Contudo, o método apresentou limitações com relação ao reconhecimento óptico de caracteres das imagens no período noturno.

5.1 Recomendações de futuras melhorias

No âmbito deste trabalho, há uma série de testes e modificações para serem realizadas, visando a evolução da solução proposta e aprimoramento dos resultados no reconhecimento da placa veicular e extração de caracteres.

Para trabalhos futuros, é pertinente realizar validação do algoritmo em imagens provenientes de fontes diversas, dada a variação nas características das imagens produzidas por dispositivos distintos, de forma que pode ser necessário realizar alguma adaptação para que o algoritmo seja indiferente à fonte dos dados. O uso de câmaras que possam operar durante o período noturno. Realizar um estudo sobre quais filtros tem melhor resultado para o procedimento de reconhecimento de caracteres. Também é válido expandir a base de dados de treinamento para incluir placas de motocicletas.

Outra perspectiva relevante envolve o desenvolvimento de ferramentas de *tracking* que permitam tratar o mesmo objeto como único, possibilitando a identificação da direção do veículo no vídeo.

Essas melhorias propostas representam oportunidades valiosas para aprimorar a eficácia e a versatilidade da solução em questão, ampliando suas aplicações potenciais e contribuindo para o avanço contínuo na área de reconhecimento de placas veiculares.

REFERÊNCIAS BIBLIOGRÁFICAS

ALESSANDRO CAUDURO, Deep Learning: o motor dos negócios na era da inteligência artificial, Disponível em: https://alessandrocauduro.medium.com/inteligência-artificial-uma-corrida-desleal-80bfa53075ed. Acesso em: 12 out. 2023.

ARAÚJO, Flávio HD et al. Redes Neurais Convolucionais com Tensorflow: Teoria e Prática. SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. III Escola Regional de Informática do Piauí. Livro Anais-Artigos e Minicursos, v. 1, p. 382-406, 2017.

BITTENCOURT, C. D. R. Classificação Automática do Acabamento de Gordura em Imagens Digitais de Carcaças Bovinas. Dissertação (Mestrado) — Instituto de Ciências Exatas, Universidade de Brasília, 2009.

BOGGIONE, G. d. A. Restauração de Imagens do Satélite LandSat-7. Dissertação (Mestrado) — Ministério da Ciência e Tecnologia, Instituto Nacional de Pesquisas Espaciais, 2004.

CAVALCANTI, G. et al. Brazilian vehicle identification using a new embedded plate recognition system. Elsevier Measurement VOL. 70, 2015.

CVAT.ai. Disponível em: https://www.cvat.ai/. Acesso em: 9 de setembro de 2023.

CHEN, C. et al. Application of Image Processing to the Vehicle License Plate Recognition. 2013.

DAVIES, E. Roy. Machine Vision: Teoria, Algoritmos, Practicalities. Morgan Kaufmann, 2005.

DEEP Learning Book, 2022, Disponível em: https://www.deeplearningbook.com.br/o-que-sao-redes-neurais-artificiais-profundas/. Acesso em: 20 out. 2023.

DENG, L. et al. Deep Learning: Methods and Applications. p. 197–387, 2013.

FONSECA GALINDO, C. et al. Sistema automático para reconhecimento de placas de automóveis baseado em processamento digital de imagens e redes perceptron de múltiplas camadas. XIX ENMC e VII ECTM, 2016.

GABRIEL RESENDE GONÇALVES; Matheus Alves Diniz; Rayson Laroca; David Menotti; William Robson Schwartz: Real-time Automatic License Plate Recognition Through Deep Multi-Task Networks. Em: Conference on Graphic, Patterns and Images (SIBGRAPI), pp. 1-8, 2018.

GONÇALVES et al. License plate recognition based on temporal redundancy. International Conference on Intelligent Transportation Systems, 2016.

GONZALEZ RAFAEL C.; WOODS, R. E. Digital Image Processing. [S.l.]: Tom Robbins, 2002.

GONZALEZ, Rafael C.; WOODS, R. C. Processamento digital de imagens. tradução: Cristina Yamagami e Leonardo Piamonte. 2010.
GOODFELLOW, I., Bengio, Y. e Courville, A. (2016). Deep Learning. MIT Press.

HE, Kaiming et al. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770-778.

J. KIM, J. Kwon Lee, and K. Mu Lee. Accurate image superresolution using very deep convolutional networks. In CVPR, 2016.

J. REDMON, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” pp. 779–788, 06 2016.

J.-B. HUANG, A. Singh, and N. Ahuja. Single image superresolution from transformed self-exemplars. In CVPR, 2015.

JAIN, KASTURI, BRIAN G. Schunck. Machine Vision. McGraw Hill, 1 edition, 1995.

LAROCA, S. et al. A robust real-time automatic license plate recognition based on the yolo detector. International Joint Conference on Neural Networks, 2018.

LIU, A. et al. Ssd: Single shot multibox detector. Proc. Eur. Conf. Comput. Vis. (ECCV), 2016.

MONTAZZOLLI; JUNG. Real-time brazilian license plate detection and recognition using deep convolutional neural networks. SIBGRAPI, 2017.

N. A. V. BEÇA. “Construção de uma Câmara de Alta Sensibilidade e Baixa Resolução para Super-Resolução”. Master’s thesis, Faculdade de Ciências e Tecnologia, Universidade de Coimbra, Coimbra, Portugal, 2008.

R. TIMOFTE, V. De Smet, and L. Van Gool. A+: Adjusted anchored neighborhood regression for fast super-resolution. In ACCV, 2014.

R. TIMOFTE, V. De, and L. V. Gool. Anchored neighborhood regression for fast example-based super-resolution. In ICCV, 2013.

RAFAEL ENZIO. Reconhecimento-Automatico-de-Placas-de-Automoveis-no-Mercosul. GitHub plataforma de hospedagem de código-fonte. Disponível em: https://github.com/FaelEnzio/Reconhecimento-Automatico-de-Placas-de-Automoveis-no-Mercosul. Acesso em: 24 nov. 2023.

REN, H. et al. Faster r-cnn: Towards realtime object detection with region proposal networks. Proc. Adv. Neural Inf. Process. Syst., 2015.

S. CHAUDHURI. Super-Resolution Imaging. Kluwer Academic Publishers, 2001.

S. JYOTHI et al, Automatic license plate recognition using pre-processing methods. 2017.

S. SCHULTER, C. Leistner, and H. Bischof. Fast and accurate image upscaling with super-resolution forests. In CVPR, 2015.

SMITH, R. (2007). An Overview of the Tesseract OCR Engine. Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), 2, 629–633.

TECNOBLOG. Novo padrão de placas de carro começa a ser usado no Brasil.11/09/2018. Disponível em: https://tecnoblog.net/noticias/2018/09/11/placa-carros-MERCOSUL-brasil/. Acesso em: 05-11-2023.

TENSORFLOW. “Tensors”. Disponível em: <https://www.tensorflow.org/>. Acesso em 7 de Maio de 2021.

TONY YIU, Understanding Neural Networks. Towards Data Science,2 Jun. de 2019. Disponível em: <https://towardsdatascience.com/understanding-neural-networks-19020b758230 > Acesso em: 01/05/2021.

UTRALYTICS. YOLO: A Brief History, Disponivek em: https://docs.ultralytics.com/#yolo-a-brief-history. Acesso em: 05-11-2023.

W. SHI, J. Caballero, C. Ledig, X. Zhuang, W. Bai, K. Bhatia, A. M. S. M. de Marvao, T. Dawes, D. ORegan, and D. Rueckert. Cardiac image super-resolution with global correspondence using multi-atlas patchmatch. In MICCAI, 2013..

W. W. Zou and P. C. Yuen. Very low resolution face recognition problem. TIP, 2012.

WANG, J.; GAO, G.; YANG, H. The Method Research of Vehicle Image Preprocessing and License Plate Location. 2009.

WANGENHEIM, Aldo von. Visão computacional: seminário introdução à visão computacional. Florianópolis, [1998]. Disponível em: Acesso em: 11 mar. 2010.

XIE, L. et al. A new cnn-based method for multi-directional. IEEE Transactions on Intelligent Transportation Systems, 2018.

XIE, Lele; AHMAD, Tasweer; JIN, Lianwen; LIU, Yuliang; ZHANG, Sheng. A New CNN-Based Method for Multi-Directional Car License Plate Detection. IEEE Transactions on Intelligent Transportation Systems, v. 19, n. 2, p. 507-517, 2018.

YUAN, Z. et al. A robust and efficient approach to license plate detection. IEEE Transactions on Image Processing, 2016.

ZHANG, Yulun et al. Residual dense network for image super-resolution. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018. p. 2472-2481.