INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA – CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

ENZZO COMASSETTO DOS SANTOS

DETECÇÃO DE PLACAS DE VELOCIDADE EM AMBIENTES VIÁRIOS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS YOLO E RECONHECIMENTO DE CARACTERES OCR

FLORIANÓPOLIS, 2024.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA – CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

ENZZO COMASSETTO DOS SANTOS

DETECÇÃO DE PLACAS DE VELOCIDADE EM AMBIENTES VIÁRIOS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS YOLO E RECONHECIMENTO DE CARACTERES OCR

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Engenheiro em Eletrônica.

Orientador:
Prof. Robinson Pizzio, D.Eng.

FLORIANÓPOLIS, 2024.

Ficha de identificação da obra elaborada pelo autor.

Santos, Enzzo
DETECÇÃO DE PLACAS DE VELOCIDADE EM AMBIENTES VIÁRIOS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS YOLO E RECONHECIMENTO DE CARACTERES OCR / Enzzo Santos; orientação de Robinson Pizzio. - Florianópolis, SC, 2024.
61 p.

Trabalho de Conclusão de Curso (TCC) - Instituto Federal de Santa Catarina, Câmpus Florianópolis. Bacharelado em Engenharia Eletrônica. Departamento Acadêmico de Eletrônica.
Inclui Referências.

1. Reconhecimento de Placas de Velocidade de Trânsito. 2. YOLO (You Only Look Once). 3. Assistência ao Motorista. 4. Sistemas de Visão Computacional. 5. Mobilidade Urbana. I. Pizzio, Robinson. II. Instituto Federal de Santa Catarina. III. DETECÇÃO DE PLACAS DE VELOCIDADE EM AMBIENTES VIÁRIOS UTILIZANDO REDES NEURAIS

DETECÇÃO DE PLACAS DE VELOCIDADE EM AMBIENTES VIÁRIOS UTILIZANDO REDES NEURAIS CONVOLUCIONAIS YOLO E RECONHECIMENTO DE CARACTERES OCR

ENZZO COMASSETTO DOS SANTOS

Este trabalho foi julgado adequado para obtenção do título de Engenheiro em Eletrônica e aprovado na sua forma final pela banca examinadora do Curso Engenharia Eletrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 22 de Fevereiro, 2024.

Banca Examinadora:

gov.br Documento assinado digitalmente
ROBINSON PIZZIO
Data: 12/03/2024 17:15:18-0300
Verifique em https://validar.iti.gov.br

Robinson Pizzio, D.Eng.

gov.br Documento assinado digitalmente
MARIA CLAUDIA DE ALMEIDA CASTRO
Data: 13/03/2024 10:31:32-0300
Verifique em https://validar.iti.gov.br

Maria Cláudia de Almeida Castro, D.Eng.
IFSC

gov.br Documento assinado digitalmente
DANIEL LOHMANN
Data: 13/03/2024 12:55:13-0300
Verifique em https://validar.iti.gov.br

Daniel Lohmann, M.Eng.
IFSC

(Dedico este trabalho aos meus pais Marcio e Carina.)

AGRADECIMENTOS

Gostaria de expressar minha sincera gratidão a todas as pessoas que contribuíram para a realização deste trabalho. Primeiramente, agradeço ao meu orientador, Robinson Pizzio, por sua orientação, apoio e por sempre se mostrar disponível e disposto a ajudar no desenvolvimento do trabalho. Agradeço também aos membros da banca examinadora por dedicarem seu tempo e expertise na avaliação deste trabalho.

Sou imensamente grato aos meus familiares, em especial aos meus pais, Marcio e Carina, pelo seu amor incondicional, incentivo e apoio inabalável ao longo de toda a minha jornada acadêmica. Seu constante encorajamento e sacrifício pessoal foram fundamentais para minha motivação e sucesso. Sem o apoio e a estrutura que eles me proporcionaram, eu não estaria onde estou hoje.

Também gostaria de agradecer aos meus amigos pelo seu constante incentivo, compreensão e apoio emocional durante todo o percurso acadêmico. Suas palavras de encorajamento foram fundamentais para minha motivação e perseverança. Gostaria de estender um agradecimento especial a eles por contribuírem diretamente para este trabalho, seja fornecendo assistência, oferecendo dicas valiosas ou ajudando com a coleta de dados. Um agradecimento especial aos que tiraram fotos de placas de trânsito para que eu pudesse testar meus treinamentos. Sua colaboração foi fundamental para o sucesso deste projeto.

Gostaria de expressar minha sincera gratidão aos meus amigos Jade e Gustavo. À Jade, por sempre ouvir meus problemas e me incentivar e ajudar a solucioná-los, sua presença e apoio foram essenciais. Ao Gustavo, que sempre esteve disponível para me dar suporte quando algum problema técnico acontecia, sua ajuda foi fundamental para superar os desafios técnicos que surgiram ao longo deste trabalho

Agradeço também ao IFSC pelo suporte estrutural oferecido ao longo da minha formação e deste projeto.

Por fim, gostaria de estender meu reconhecimento a todos aqueles cujo apoio e colaboração foram essenciais para a conclusão deste trabalho, mesmo que não mencionados nominalmente. Este trabalho não teria sido possível sem o auxílio de cada um de vocês.

Obrigado!

"O aspecto mais emocionante da ciência é a sua constante descoberta de novos conhecimentos, que nos leva a uma compreensão mais profunda do que somos e do mundo que nos rodeia." - Isaac Asimov

RESUMO

Este trabalho de conclusão de curso explora a aplicação da técnica YOLO (*You Only Look Once*) juntamente com reconhecimento de caracteres OCR no reconhecimento de placas de velocidade de trânsito, com o objetivo de aprimorar os sistemas de assistência ao motorista. A abordagem YOLO, conhecida por sua eficiência computacional e precisão na detecção de objetos em tempo real, oferece uma solução promissora para automatizar a identificação de placas de velocidade, fornecendo informações cruciais para os sistemas de navegação e alerta ao condutor.

Em um cenário onde a segurança viária e a eficiência no trânsito são prioridades crescentes, a integração do YOLO no reconhecimento de placas de trânsito representa um avanço significativo. A capacidade de reconhecer rapidamente placas de velocidade com precisão é essencial para o funcionamento eficaz dos sistemas de assistência ao motorista, contribuindo diretamente para a prevenção de acidentes e a tomada de decisões informadas durante a condução.

Ao longo do desenvolvimento deste estudo, foram exploradas diversas etapas, desde a configuração do ambiente de treinamento e técnicas para aumentar a eficiência do treinamento até a análise dos resultados obtidos. A integração do YOLO demonstrou-se eficaz na detecção de placas de velocidade em diferentes condições, proporcionando uma base sólida para aprimoramentos futuros.

Os resultados deste estudo apontam para o potencial do YOLO como uma ferramenta poderosa para melhorar a segurança viária e a eficiência do trânsito. Ao automatizar a detecção de placas de velocidade, este trabalho não apenas contribui para aprimorar os sistemas existentes, mas também abre portas para futuras inovações na área de segurança veicular e mobilidade urbana.

Palavras-chave: Reconhecimento de Placas de Velocidade de Trânsito. YOLO (*You Only Look Once*). Assistência ao Motorista. Sistemas de Visão Computacional. Mobilidade Urbana.

ABSTRACT

This thesis explores the application of the YOLO (You Only Look Once) technique along with Optical Character Recognition (OCR) in recognizing traffic speed signs, aiming to enhance driver assistance systems. The YOLO approach, renowned for its computational efficiency and real-time object detection precision, offers a promising solution for automating speed sign identification, providing crucial information for navigation and driver alert systems.

In a context where road safety and traffic efficiency are increasingly prioritized, integrating YOLO into traffic sign recognition represents a significant advancement. The ability to quickly and accurately recognize speed signs is essential for the effective operation of driver assistance systems, directly contributing to accident prevention and informed decision-making during driving.

Throughout the development of this study, various stages were explored, from setting up the training environment and techniques to enhance training efficiency to analyzing the obtained results. The integration of YOLO proved effective in detecting speed signs under different conditions, providing a solid foundation for future enhancements.

The findings of this study point to the potential of YOLO as a powerful tool to improve road safety and traffic efficiency. By automating speed sign detection, this work not only contributes to enhancing existing systems but also opens doors for future innovations in the field of vehicular safety and urban mobility.

Keywords: Speed Limit Signs Recognition. YOLO (You Only Look Once). Driver Assistance. Computer Vision Systems. Urban Mobility.

LISTA DE FIGURAS

Figura 1 – Detecção de objetos utilizando YOLO 23
Figura 2 – Detectores de objeto no conjunto de dados MS COCO 24
Figura 3 – Exemplo de imagens presentes no dataset do GTSRB 37
Figura 4 – Exemplo de imagem adequada para treinamento com YOLO 39
Figura 5 – Exemplo de imagem do novo dataset 40
Figura 6 – Exemplo de imagem após o processamento por scripts do data augmentation 42
Figura 7 – Imagem de entrada de uma placa 44
Figura 8 – Imagem após identificação da bounding box 44
Figura 9 – Reconhecimento de velocidade de 60km/h identificada pelo sistema 45
Figura 10 – Identificação de placas em ambientes urbanos 47
Figura 11 – Identificação de placas em ambientes urbanos 48
Figura 12 – Identificação de placas em ambientes urbanos 48
Figura 13 – Identificação de placas em ambientes de baixa nitidez 49
Figura 14 – Identificação de placas em ambientes de baixa nitidez 49
Figura 15 – Identificação de placas em ambientes de baixa nitidez 50
Figura 16 – Identificação de placas em ambiente noturno 50
Figura 17 – Identificação de placas em ambiente noturno 51
Figura 18 – Identificação de placas com coloração danificada 51
Figura 19 – Identificação de placa parcialmente coberta 52
Figura 20 – Falso positivo em placa de ciclovia 53

Lista de tabelas

Tabela 1 – Resultados do treinamento da rede de identificação de velocidade de placas de trânsito 46

Lista de abreviaturas e siglas

ADAS	Sistemas Avançados de Assistência ao Condutor
AP	*Average Precision*
CNN	*Convolutional Neural Network*
CUDA	*Compute Unified Device Architecture*
DAE	Departamento Acadêmico de Eletrotécnica
DNN	Rede Neural Profunda
FPS	*Frames* por segundo
GAN	Rede Adversária Generativa
GPU	Unidade de processamento gráfico
GTSRB	*German Traffic Sign Recognition Benchmark*
HP	Hewlett-Packard
IFSC	Instituto Federal de Santa Catarina
JPEG	*Joint Photographic Experts Group*
LIDAR	*Light Detection and Ranging*
LSTM	*Long short-term memory*
MS COCO	*Microsoft Common Objects in Context*
OCR	Reconhecimento ótico de caracteres
PAN	*Path Aggregation Network*
PNG	*Portable Network Graphics*
R-CNN	*Region Based Convolutional Neural Networks*
SENATRAN	Secretaria Nacional de Trânsito
TIFF	*Tagged Image File Format*
WHO	*World Health Organization*
YOLO	*You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO..... 14
1.1 Justificativa..... 14
1.2 Definição do Problema..... 16
1.3 Objetivo Geral..... 17
1.4 Objetivos Específicos..... 17
1.5 Estrutura do Trabalho..... 18
2 FUNDAMENTAÇÃO TEÓRICA..... 18
2.1 Sistemas Avançados de Assistência ao Condutor (ADAS)..... 19
2.2 YOLO (You Only Look Once)..... 20
2.2.1 Fundamentos e princípios do YOLOv4..... 23
2.2.1.1 Estrutura..... 23
2.2.1.2 darknet..... 24
2.2.1.3 CNN..... 25
2.2.1.4 CUDA..... 25
2.2.1.5 Aumento de dados em imagens..... 26
2.2.1.6 Vantagens e Limitações..... 27
2.3 Tesseract OCR..... 28
2.4 Visão Computacional em Veículos Autônomos..... 28
2.5 Google Colab..... 30
3 METODOLOGIA..... 31
3.1 Métodos aplicados..... 32
3.1.1 Treinamento de uma rede neural com YOLO..... 32
3.1.2 Configurando o darknet para treinamento de objetos no YOLOv4..... 33
3.1.3 Considerações sobre o uso do Google Colab para realizar o treinamento..... 35
3.1.4 Escolha do dataset e primeiros treinamentos..... 36
3.1.5 Dataset definitivo e ajustes necessários para obtenção dos resultados desejados..... 39
3.1.5.1 Escolha do dataset e ajustes..... 40
3.1.5.2 Utilização do data augmentation..... 41
3.1.5.3 Treinamento final..... 43
4 APRESENTAÇÃO DOS RESULTADOS..... 46
5 CONSIDERAÇÕES FINAIS..... 54
5.1 Sugestões para trabalhos futuros..... 54
ANEXOS..... 58
ANEXO A – Fórmulas utilizadas..... 59

1 INTRODUÇÃO

A engenharia eletrônica tem presenciado avanços significativos em tecnologias de visão computacional nos últimos anos, impulsionando inovações cruciais no campo da segurança viária. Neste contexto, este trabalho se dedica à investigação do reconhecimento eficiente de placas de velocidade de trânsito, um aspecto fundamental na melhoria dos sistemas de assistência ao motorista. Com o aumento dos desafios relacionados à segurança nas estradas, a automação veicular e os dispositivos de auxílio ao condutor se tornaram cada vez mais relevantes (Brookhius, 2001).

A importância desses sistemas é inegável na sociedade contemporânea, onde a segurança rodoviária e a eficiência do tráfego são prioridades essenciais (Dixit, 2021). Este estudo explora uma abordagem para o reconhecimento de placas de velocidade de trânsito, visando não apenas aprimorar a segurança veicular, mas também proporcionar uma experiência de condução mais segura e informada. Ao capacitar os condutores com informações cruciais e facilitar a tomada de decisões, espera-se prevenir acidentes e melhorar significativamente a segurança nas estradas.

Assim, este trabalho almeja contribuir para o avanço da engenharia eletrônica aplicada à mobilidade, ao abordar uma necessidade crescente por soluções tecnológicas eficientes e seguras, espera-se fornecer uma contribuição significativa para a otimização do tráfego rodoviário.

1.1 Justificativa

A crescente complexidade do cenário do trânsito, aliada ao aumento do número de veículos em circulação, tem levado a uma preocupante escalada nos índices de acidentes e infrações de trânsito em todo o mundo. A gravidade desses eventos não apenas resulta em perdas humanas significativas, mas também impõe um ônus substancial às estruturas de saúde e aos recursos sociais.

Analisando o panorama das infrações de trânsito, torna-se evidente que o excesso de velocidade se destaca como uma das principais causas de incidentes. Segundo o Anuário do Departamento Nacional de Trânsito (DENATRAN) de 2021, a infração por excesso de velocidade foi a mais recorrente, totalizando impressionantes 1.182.603 ocorrências. Este dado reflete não apenas a frequência dessa infração, mas também a necessidade urgente de estratégias e tecnologias que contribuam para mitigar os riscos associados a velocidades inadequadas nas vias.

Segundo dados da World Health Organization (WHO, 2023), a cada ano, os incidentes no trânsito resultam em cerca de 1,19 milhão de mortes globais e causam lesões não fatais em uma faixa de 20 a 50 milhões de pessoas. Mais da metade desses óbitos ocorrem entre usuários considerados mais vulneráveis nas vias, como pedestres, ciclistas e motociclistas.

> As lesões por acidentes de trânsito são a principal causa de morte para crianças e jovens adultos com idades entre 5 e 29 anos. No entanto, dois terços das fatalidades no trânsito ocorrem entre pessoas em idade de trabalho (18 a 59 anos). Nove em cada 10 mortes nas estradas ocorrem em países de baixa e média renda, embora esses países tenham cerca de 60% dos veículos do mundo. (WHO, 2023).

A Secretaria Nacional de Trânsito (SENATRAN, 2023) informa que o Brasil possui uma frota ativa de 80.572.525 veículos, e que no período entre 2018 e 2023 ocorreram 5.066.466 acidente totalizando 7.960.213 feridos/ilesos envolvidos nos acidentes e 122.792 óbitos, tendo uma taxa de 2,42% de óbitos por acidente.

Para reduzir essas estatísticas de acidentes, a OMS atua como secretariado para a Década de Ação das Nações Unidas para a Segurança Viária 2021–2030, que tem como objetivo reduzir as mortes e lesões no trânsito em pelo menos 50% até 2030.

> Veículos seguros desempenham um papel essencial na prevenção de acidentes e na redução da probabilidade de lesões graves. Há uma série de regulamentos das Nações Unidas sobre segurança veicular que, se aplicados aos padrões de produção dos países, potencialmente salvariam muitas vidas. Isso inclui exigir que os fabricantes de veículos cumpram as regulamentações de impacto dianteiro e lateral, incluindo controle eletrônico de estabilidade, airbags e cintos de segurança em todos os veículos. Sem esses padrões básicos, o risco de lesões no trânsito – tanto para os que estão nos veículos quanto para os que estão fora deles – aumenta consideravelmente.(OPAS, 2023)

Diante desse contexto, a presente pesquisa busca abordar diretamente os desafios impostos pelos acidentes de trânsito e pelas infrações, com um foco

particular na infração de excesso de velocidade. Ao integrar o reconhecimento de placas de velocidade de trânsito aos Sistemas Avançados de Assistência ao Condutor (ADAS - *Advanced driver-assistance system*), visa-se não apenas combater a incidência dessa infração, mas também contribuir para a prevenção global de acidentes, aprimorando a segurança nas estradas e promovendo uma condução mais responsável e consciente (Brookhius, 2001).

Neste contexto, a seleção de técnicas específicas de visão computacional assume uma importância crucial para garantir o sucesso e a eficácia dos sistemas ADAS. Destaca-se, nesse sentido, a relevância do YOLO (*You Only Look Once*), uma abordagem reconhecida pela sua velocidade na detecção de objetos em tempo real. Essa característica é especialmente fundamental para detecções precisas em ambientes dinâmicos, como o tráfego viário.

1.2 Definição do Problema

O reconhecimento eficaz de placas de velocidade é fundamental para o sucesso dos ADAS, pois fornece informações cruciais para a tomada de decisões rápidas e seguras durante a condução. O desafio reside não apenas na precisão da detecção, mas também na capacidade de processamento em tempo real, considerando a dinâmica do ambiente rodoviário.

Portanto, o problema central deste trabalho consiste em desenvolver e aprimorar um sistema de reconhecimento de placas de velocidade de trânsito utilizando a abordagem YOLO. Isso envolve a superação de desafios relacionados à velocidade de processamento, precisão na detecção e a capacidade de resposta do sistema em ambientes dinâmicos, contribuindo assim para a eficácia global dos sistemas de assistência ao condutor e, por conseguinte, para a segurança nas estradas.

1.3 Objetivo Geral

O objetivo principal deste trabalho é configurar e treinar um sistema de reconhecimento de placas de velocidade de trânsito, para que o mesmo possa ser aplicado no futuro de maneira eficaz aos Sistemas Avançados de Assistência ao Condutor. A pesquisa busca, assim, aprimorar a integração de tecnologias de visão computacional nos veículos, destacando a importância da velocidade de detecção na dinâmica do tráfego rodoviário. Considerando estes fatores a abordagem YOLO, foi escolhida pois possui um tempo de resposta rápido e já é comprovado seu funcionamento em câmeras de 30 fps (Redmon, 2016). Já o Tesseract OCR não possui um atraso considerável na detecção de caracteres por ser uma ferramenta dedicada a detecção de caracteres em condições específicas (Poorani, 2022).

1.4 Objetivos Específicos

Configurar e treinar um modelo YOLO específico para o reconhecimento de placas de velocidade de trânsito, considerando a diversidade de formatos e condições de iluminação:

a) Treinar uma Rede YOLO para Detecção de Placas de Velocidade: Realizar o treinamento de uma rede YOLO utilizando um *dataset* específico para a detecção de placas de velocidade em imagens de trânsito;

b) Avaliar a Eficiência do Treinamento: Analisar a eficiência do treinamento da rede YOLO em detectar placas de velocidade, considerando diferentes condições de iluminação, ângulos de visão e ambientes;

c) Realizar Ajustes e Refinamentos: Identificar possíveis melhorias no desempenho da rede YOLO e realizar ajustes, refinamentos ou implementação de técnicas adicionais para mitigar desafios específicos, como falsos positivos;

d) Avaliar a necessidade de implementar outras técnicas de detecção junto com o YOLO;

1.5 Estrutura do Trabalho

Este trabalho é composto por seis seções principais, proporcionando uma abordagem abrangente da pesquisa. A seção de Resumo oferece uma sinopse dos objetivos, metodologia e conclusões essenciais do trabalho.

Na seção de Introdução, define-se o contexto, a justificativa, a definição do problema e os objetivos específicos que orientam a pesquisa.

Explora-se a Fundamentação Teórica na terceira seção, abordando conceitos teóricos e técnicas fundamentais, como segurança no trânsito, sistemas avançados de assistência ao condutor, *framework* YOLO, Tesseract OCR e visão computacional em veículos autônomos.

A Metodologia, quarta seção, detalha os métodos empregados, incluindo o treinamento de uma rede neural com YOLO, configurações específicas do *darknet* para o YOLOv4, seleção e ajustes do *dataset* para alcançar os resultados desejados.

Os Resultados são apresentados e discutidos na quinta seção, incluindo a análise da taxa de acerto na detecção de placas de velocidade sob diversas condições, como iluminação variada e diferentes ângulos de visão.

Por fim, nas Considerações Finais, destacamos as principais conclusões do trabalho, acompanhadas por sugestões para possíveis pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Sistemas Avançados de Assistência ao Condutor (ADAS)

Segundo Brookhuis (2001) ADAS são sistemas eletrônicos desenvolvidos para auxiliar o motorista durante a condução de um veículo. Esses sistemas têm como objetivo reduzir erros humanos, melhorar a segurança no trânsito, aumentar a eficiência do transporte e proporcionar mais conforto ao motorista. Alguns exemplos de tecnologias ADAS incluem:

a. Sensores de ponto cego: esses sensores alertam o motorista quando há um veículo em seu ponto cego, ajudando a evitar colisões durante as mudanças de faixa;
b. Controle de cruzeiro adaptativo: esse sistema ajusta automaticamente a velocidade do veículo para manter uma distância segura do veículo à frente, ajudando a evitar colisões por trás;
c. Assistência de estacionamento: essa tecnologia ajuda o motorista a estacionar o veículo, fornecendo orientações e controlando a direção e a velocidade do veículo;
d. Alertas de colisão: esses alertas avisam o motorista quando há um risco iminente de colisão, permitindo que o motorista tome medidas para evitar a colisão;
e. Sistemas de navegação: esses sistemas fornecem orientações de direção e informações sobre o tráfego, ajudando o motorista a chegar ao seu destino com segurança e eficiência.

Os ADAS oferecem uma série de benefícios. Eles têm o potencial de reduzir erros dos condutores e aumentar a eficiência no tráfego e transporte, o que pode resultar em uma considerável diminuição do sofrimento humano, custos econômicos e poluição. Além disso, os sistemas ADAS podem melhorar significativamente a segurança no trânsito, permitindo acomodar mais veículos nas estradas e vias dedicadas, e possibilitando a condução de alto desempenho sem se preocupar com condições ambientais e climáticas. Também é destacado que os sistemas ADAS podem tornar os condutores mais seguros e eficientes, especialmente para condutores idosos e inexperientes. Ademais, os sistemas ADAS

facilitam a realização da tarefa de condução, fornecendo conselhos, instruções e alertas em tempo real para os condutores (BROOKHUIS, 2001).

O resultado dos estudos apresentados por So (2020) propõe uma metodologia para estimar a efetividade prática dos ADAS em diferentes tipos de vias. Os dados apresentados pelo estudo mostram que a efetividade prática dos sistemas ADAS varia entre diferentes tipos de vias, com uma efetividade mais consistente em vias urbanas do que em vias rurais. Essas informações são fundamentais para compreender a eficácia real dos sistemas ADAS e para orientar a formulação de políticas e estratégias relacionadas a esses sistemas.

Os resultados do estudo também destacam a importância de ajustar a efetividade inicial dos sistemas ADAS para refletir a efetividade prática em condições reais de operação. Os ajustes levam em conta fatores como a taxa de adoção do sistema, a taxa de uso do sistema e a exigência mínima de ativação do sistema. Essas considerações são essenciais para garantir que as estimativas de efetividade dos sistemas ADAS sejam precisas e confiáveis.

2.2 YOLO (You Only Look Once)

Joseph Redmon, o desenvolvedor do YOLO, descreve a abordagem deste modelo de detecção em seu artigo da seguinte maneira:

> Sistemas de detecção anteriores frequentemente adaptam classificadores ou localizadores para realizar a detecção, aplicando o modelo a uma imagem em várias localizações e escalas. Regiões da imagem com pontuações elevadas são consideradas detecções. No entanto, nosso método difere substancialmente. Aplicamos uma única rede neural à imagem completa, que, por sua vez, a divide em regiões, prevendo caixas delimitadoras e probabilidades para cada uma. Essas caixas delimitadoras são ponderadas pelas probabilidades previstas. (Redmon, 2016)

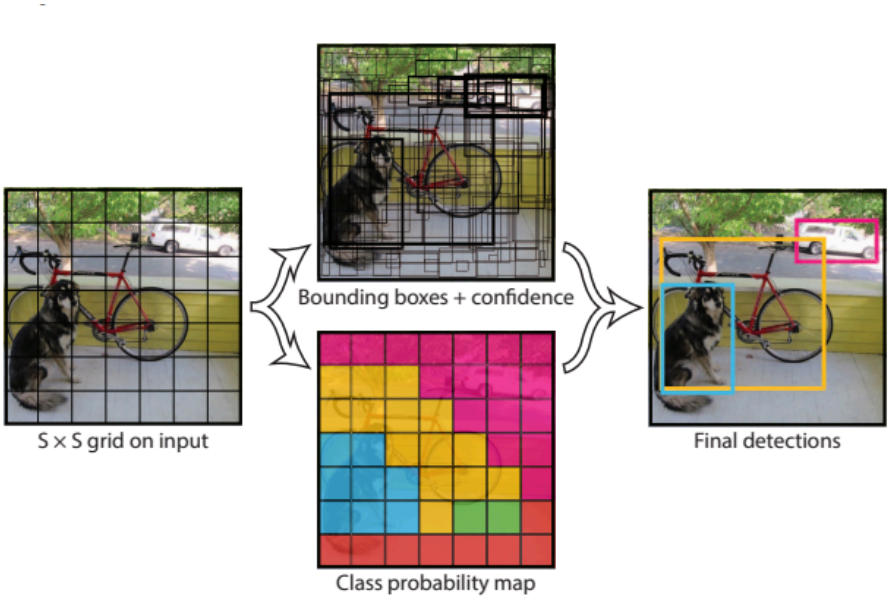
Redmon destaca as vantagens do modelo em relação aos sistemas baseados em classificadores, ressaltando que o YOLO analisa a imagem globalmente no momento do teste, permitindo previsões informadas pelo contexto global da imagem. Além disso, destaca a eficiência do YOLO ao realizar previsões com uma única avaliação de rede, proporcionando uma velocidade

significativamente maior em comparação com sistemas como R-CNN (*Region Based Convolutional Neural Networks*) e *Fast* R-CNN.

Segundo Redmon (2016) As abordagens recentes, exemplificadas pelo R-CNN, empregam métodos de proposição de regiões para gerar inicialmente caixas delimitadoras potenciais em uma imagem, seguido por um processo classificador aplicado a essas caixas propostas. Após a classificação, um pós-processamento é utilizado para aprimorar as caixas delimitadoras, remover detecções duplicadas e reatribuir pontuações com base em outros objetos na cena. Essas *pipelines* complexas são caracterizadas por baixa velocidade e dificuldade de otimização, uma vez que cada componente individual precisa ser treinado separadamente.

Em contraste, a abordagem do YOLO reformula a detecção de objetos como um único problema de regressão, indo diretamente nos *pixels* da imagem para as coordenadas das caixas delimitadoras e probabilidades de classe. Essa técnica inovadora permite que, com apenas uma análise da imagem, o YOLO preveja quais objetos estão presentes e onde estão. Esse método simplificado se destaca pela eficiência e supera a necessidade de treinamentos e otimizações independentes para cada componente. A Figura 1 demonstra a técnica de detecção utilizada pelo YOLO. Como descrito por Redmon (2016), o YOLO utiliza uma técnica chamada “detecção de objetos em grade”, que divide a imagem em uma grade e realiza a detecção de objetos diretamente nessa grade, pois a partir da grade inicial o yolo detecta a probabilidade da mesma conter um objeto a ser identificado, e a partir dos filtros presentes na rede YOLO ele consegue identificar os objetos de interesse em apenas uma única análise, não sendo necessário várias iterações para reconhecer um objeto.

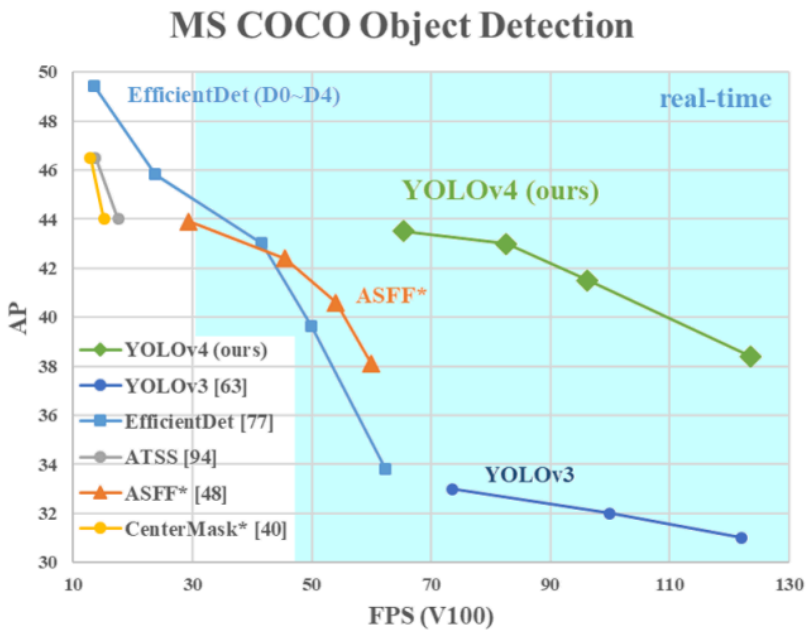
Figura 1 – Detecção de objetos utilizando YOLO



Fonte: Redmon (2016).

A Figura 2 retirada como explicado por Bochkovskiy (2020), compara a precisão média (*average precision,* AP) com os *frames* por segundo (FPS) de vários detectores de objetos, como visto no gráfico, o YOLOv4, utilizado neste trabalho, é o detector que possui uma melhor relação entre precisão e eficiência computacional, rodando duas vezes mais rápido que o EfficientDet e tendo uma precisão maior do que seu antecessor YOLOv3, apesar de ambos possuírem uma velocidade de detecção semelhante.

Figura 2 – Detectores de objetos no conjunto de dados MS COCO



Fonte: Bochkovskiy (2020).

MS COCO (*Microsoft Common Objects in Context*) é um conjunto de dados amplamente utilizado na área de visão computacional, especialmente em tarefas de detecção e segmentação de objetos. Criado pela Microsoft, o MS COCO foi projetado para fornecer um conjunto diversificado e desafiador de imagens do mundo real para treinamento e avaliação de algoritmos de visão computacional.

2.2.1 Fundamentos e princípios do YOLOv4

2.2.1.1 Estrutura

A estrutura do YOLOv4 descrita por Bochkovskiy (2020) é composta por três partes principais: a espinha dorsal (*backbone*), o pescoço (*neck*) e a cabeça (*head*). A espinha dorsal é responsável por extrair características significativas das imagens de entrada, enquanto o pescoço é responsável por agregar informações de diferentes níveis da espinha dorsal para melhorar a precisão da detecção de objetos. A cabeça é responsável por gerar as caixas delimitadoras e as pontuações de confiança para cada objeto detectado.

No YOLOv4, a espinha dorsal escolhida é a *CSPdarknet53*, que é uma versão modificada da *darknet53* que utiliza conexões parciais em cascata (CSP) para melhorar a eficiência e a precisão da detecção de objetos. O pescoço é composto pelo *Path Aggregation Network* (PAN), que é uma técnica de agregação de recursos que combina informações de diferentes níveis da espinha dorsal para melhorar a precisão da detecção de objetos. A cabeça é baseada no YOLOv3, que utiliza uma abordagem de detecção de objetos baseada em ancoragem (*anchor-based*) para gerar as caixas delimitadoras e as pontuações de confiança.

2.2.1.2 darknet

De acordo com Bochkovskiy (2020) *darknet* é uma estrutura de rede neural convolucional (CNN) que desempenha um papel crucial no contexto do YOLO, um popular modelo de detecção de objetos em imagens. A *darknet* é uma arquitetura de CNN altamente eficiente, conhecida por sua capacidade de processamento rápido e sua capacidade de manter um alto nível de precisão na detecção de objetos. No contexto do YOLOv4, a *darknet53* é a espinha dorsal (*backbone*) escolhida, fornecendo a base para a detecção de objetos em tempo real com alta precisão.

O Darknet-53 é uma arquitetura de rede neural convolucional (CNN) desenvolvida para tarefas de detecção de objetos e classificação de imagens. O Darknet-53 é composto por 53 camadas convolucionais e é projetado para extrair características complexas de imagens de entrada, o que o torna adequado para tarefas de visão computacional, como detecção de objetos em tempo real (Bochkovskiy, 2020).

A importância da *darknet* para o YOLO reside em sua capacidade de extrair características significativas das imagens de entrada, permitindo que o YOLO detecte objetos com eficiência e precisão. Além disso, a *darknet* desempenha um papel fundamental na habilidade do YOLO de processar rapidamente as informações das imagens, tornando-o adequado para aplicações em tempo real.

2.2.1.3 CNN

Segundo Saha (2018), uma rede neural convolucional é definida do seguinte modo:

> Uma Rede Neural Convolucional (ConvNet/CNN) é um algoritmo de Aprendizado Profundo que pode receber uma imagem de entrada, atribuir importância a vários aspectos/objetos na imagem e ser capaz de diferenciá-los uns dos outros. O pré-processamento necessário em uma CNN é significativamente menor em comparação com outros algoritmos de classificação. Enquanto em métodos primitivos os filtros são projetados manualmente, com treinamento suficiente, as CNN têm a capacidade de aprender esses filtros/características.

Saha ainda destaca que uma Rede Neural Convolucional opera de maneira semelhante à organização da conectividade dos neurônios no cérebro humano, sendo influenciada pela estrutura do córtex visual. Nessa analogia, os neurônios individuais de uma *ConvNet* respondem a estímulos em regiões restritas do campo visual, denominadas Campos Receptivos, que, quando sobrepostos, abrangem a totalidade da área visual. Esse *design* espelhado na biologia permite à *ConvNet* aprender e discernir características complexas em uma imagem, simulando a capacidade humana de distinguir diferentes aspectos ou objetos.

2.2.1.4 CUDA

De acordo com o *site* da Nvidia (NVIDIA CORPORATION) para desenvolvedores CUDA (*Compute Unified Device Architecture*) é uma arquitetura de computação paralela desenvolvida pela NVIDIA para seus processadores gráficos (GPUs). Introduzida pela primeira vez em 2006, a tecnologia CUDA tem desempenhado um papel fundamental na evolução do processamento paralelo e na aceleração de cargas de trabalho computacionais intensivas.

A principal característica do CUDA é sua capacidade de permitir que os desenvolvedores utilizem a potência de processamento massivamente paralela das GPUs NVIDIA para tarefas além da simples renderização gráfica. Com o CUDA, as GPUs podem ser aproveitadas para realizar cálculos complexos e simultâneos, aumentando significativamente o desempenho de algoritmos e aplicações que requerem grande poder computacional.

Essa arquitetura de programação paralela utiliza uma extensão da linguagem de programação C, tornando-a acessível aos desenvolvedores familiarizados com programação convencional, o CUDA tambem pode ser utilzado em ambientes de execução Python. Ao dividir tarefas em *threads*, que são unidades de execução individuais, o CUDA permite que milhares de *threads* sejam executados simultaneamente, acelerando operações matemáticas e científicas.

Para o treinamento de modelos de aprendizado profundo, como o YOLOv4, o suporte ao CUDA é crucial. Ele possibilita a execução rápida e eficiente das operações matriciais intensivas envolvidas nos cálculos das redes neurais, aproveitando a natureza altamente paralela das GPUs.

2.2.1.5 Aumento de dados em imagens

Segundo o *site datacamp* em seu artigo "*A Complete Guide to Data Augmentation*" *Data augmentation* (aumento de dados) é uma técnica crucial empregada em aprendizado de máquina para expandir artificialmente conjuntos de treinamento, criando versões modificadas de dados existentes. Esse processo envolve introduzir alterações mínimas no conjunto de dados ou utilizar métodos de aprendizado profundo para gerar novos pontos de dados. Existem dois tipos de aumento de dados:

Dado Aumentado: Este tipo é derivado do conjunto de dados original com pequenas modificações. No contexto da ampliação de um conjunto de imagens, são aplicadas transformações geométricas e de espaço de cores, como inversão, redimensionamento, recorte, brilho e ajustes de contraste, para aumentar o tamanho e a diversidade do conjunto de treinamento.

Dado Sintético: Em contraste, o dado sintético é gerado artificialmente sem utilizar o conjunto de dados original. Isso frequentemente envolve o uso de Redes Neurais Profundas (DNNs) e Rede Adversária Generativa (GANs) para criar dados sintéticos.

Abaixo estão listadas as técnicas mais utilizadas em imagens para ampliar um conjunto de imagens já existentes:

a) Injeção de Ruído: Adição de ruído aleatório ao conjunto de dados da imagem para aprimorar o desempenho do modelo;

b) Transformações Geométricas: Inverter, cortar, girar, esticar e ampliar imagens aleatoriamente. Cuidado deve ser tomado ao aplicar múltiplas transformações nas mesmas imagens para evitar redução no desempenho do modelo;

c) Transformações de Espaço de Cores: Mudar aleatoriamente os canais de cor RGB, contraste e brilho;

d) Filtros de *Kernel*: Mudar aleatoriamente a nitidez ou desfoque da imagem;

e) Apagamento Aleatório: Excluir parte da imagem inicial;

f) Mistura de Imagens: Misturar e combinar várias imagens.

Essa ampla gama de técnicas de *data augmentation* demonstra a versatilidade e adaptabilidade dessa abordagem em vários tipos de dados, contribuindo significativamente para a robustez e eficácia de modelos de aprendizado de máquina. É importante ressaltar que se deve avaliar quais técnicas podem ser adequadas ao conjunto de dados utilizados, identificar uma abordagem efetiva de aumento de dados pode ser uma tarefa desafiadora.

2.2.1.6 Vantagens e Limitações

Como demonstrado no trabalho de Bochkovskiy (2020) o YOLOv4 possui capacidade de operar em tempo real em uma GPU convencional, tornando-o adequado para uma ampla gama de aplicações práticas. Além disso, o YOLOv4 demonstra melhorias significativas em termos de precisão e velocidade em comparação com outros detectores de objetos de última geração.

Uma desvantagem, e o principal desafio enfrentado neste trabalho na utilização do YOLOv4, é observada também no artigo de Poorani (2022) "*Number Plate Detection Using YOLOV4 and Tesseract OCR*" que descreve a utilização do

YOLO para identificação de placas veiculares. Segundo Poorani (2022) o YOLOV4 é um modelo de detecção de objetos utilizado para localizar a placa de licença em uma imagem. No entanto, o mesmo não é capaz de reconhecer os caracteres na placa de licença. Para realizar o reconhecimento dos caracteres, o Tesseract OCR, um *software* de reconhecimento óptico de caracteres, é empregado após a detecção da placa de licença pelo YOLOV4. Essa abordagem permite que o modelo seja capaz de reconhecer os caracteres na placa de licença, possibilitando a identificação completa da placa de licença.

2.3 Tesseract OCR

De acordo com o GitHub do Tesseract (2023), este foi originalmente desenvolvido nos Laboratórios Hewlett-Packard no Reino Unido e nos Estados Unidos entre 1985 e 1994, com posterior código aberto pela HP em 2005, passou a ser mantido pelo Google de 2006 a novembro de 2018. Atualmente, sob a liderança de Stefan Weil, a versão estável 5 (lançada em 30 de novembro de 2021) está disponível no GitHub.

Esta ferramenta oferece uma *engine* OCR (libtesseract) e um programa de linha de comando (tesseract). A versão 4 introduziu um novo mecanismo OCR baseado em redes neurais (LSTM), focado em reconhecimento de linha, mas mantendo suporte ao mecanismo legado do Tesseract 3 para reconhecimento de padrões de caracteres.

Compatível com mais de 100 idiomas, o Tesseract suporta formatos de imagem como PNG, JPEG e TIFF. É relevante mencionar que, para obter melhores resultados de OCR, a qualidade da imagem fornecida ao Tesseract é crucial.

2.4 Visão Computacional em Veículos Autônomos

> “Um Veículo Autônomo (VA) é capaz de realizar funções parciais ou completas, incluindo, entre outras coisas, condução, estacionamento e manutenção na faixa, com supervisão indireta do motorista ou nenhuma supervisão. “ (Dixit, 2021)

Segundo o artigo "*Safety and Risk Analysis of Autonomous Vehicles Using Computer Vision and Neural Networks*" (DIXIT, 2021), a visão computacional em veículos autônomos refere-se à capacidade dos veículos de interpretar e compreender o ambiente ao seu redor por meio de sistemas de câmeras e sensores. Essa tecnologia permite que os veículos autônomos identifiquem e reconheçam objetos, pedestres, sinais de trânsito, faixas de rodagem e outros elementos importantes para a navegação segura.

Os sistemas de visão computacional utilizam câmeras de alta resolução e LIDARs de alta velocidade para capturar informações detalhadas sobre o ambiente, permitindo que os veículos autônomos tomem decisões em tempo real com base nessas informações visuais. Isso é fundamental para a segurança e a capacidade dos veículos autônomos de operar de forma independente e prever possíveis situações de risco.

Dixit também destaca as aplicações de visão computacional em sistemas avançados de assistência ao motorista (ADAS) em veículos autônomos. A visão computacional desempenha um papel fundamental na capacidade dos veículos autônomos de compreender e interpretar o ambiente ao seu redor. Por meio de algoritmos e técnicas de processamento de imagens, a visão computacional permite que os veículos autônomos identifiquem e reconheçam objetos, pedestres, sinais de trânsito, faixas de tráfego, obstáculos e demais elementos do cenário rodoviário.

Essa capacidade de percepção visual é essencial para a tomada de decisões autônomas, como desvio de obstáculos, manutenção da trajetória, reconhecimento de semáforos e sinais de trânsito, e detecção de potenciais situações de risco. Além disso, a integração da visão computacional com sistemas de redes neurais permite o desenvolvimento de modelos de aprendizado de máquina capazes de aprimorar a capacidade preditiva e de resposta dos veículos autônomos diante de diferentes cenários e condições de tráfego (DIXIT, 2021).

2.5 Google Colab

Conforme descrito na página do Google Colab (Google, 2024) Google Colab é uma plataforma de computação em nuvem que oferece um ambiente de desenvolvimento integrado (IDE) baseado em *notebooks* Jupyter. Ele permite aos usuários escrever e executar código Python diretamente no navegador da *web*, sem a necessidade de configurar um ambiente de desenvolvimento local. O Colab é especialmente útil para desenvolvimento de projetos de aprendizado de máquina e ciência de dados, pois oferece acesso gratuito a recursos de computação poderosos, incluindo GPUs e TPUs, permitindo a execução de tarefas intensivas em processamento, como treinamento de modelos de deep learning (Google, 2024).

Além disso, o Colab facilita o compartilhamento de notebooks e colaboração em tempo real entre usuários. Sua integração com o Google Drive permite o armazenamento e o acesso fácil aos notebooks e conjuntos de dados, tornando-o uma ferramenta versátil e acessível para desenvolvedores e pesquisadores.

Os ambientes de execução dos Google Colab são compatíveis com o CUDA pois utilizam GPUs Nvidia (Google, 2024).

3 METODOLOGIA

A metodologia adotada para desenvolver este trabalho de conclusão de curso envolveu uma série de etapas cuidadosamente planejadas para alcançar resultados precisos e confiáveis no reconhecimento de placas de velocidade utilizando o *framework* YOLO.

Após a definição clara do problema em questão, o primeiro passo consistiu em buscar exemplos de treinamentos utilizando o YOLO. Nesse sentido, foram consultados tutoriais relevantes, destacando-se o tutorial de Joseph Redmon (2016).

O treinamento do YOLO foi executado de maneira integral no ambiente Google Drive/Colab, escolhido estrategicamente para oferecer uma abordagem acessível e gratuita, permitindo que outros pesquisadores e entusiastas pudessem replicar o treinamento de maneira facilitada, sem a necessidade de um *hardware* robusto próprio para que o treinamento seja realizado, os passos para a reprodução do treinamento assim como todos os códigos utilizados durante o projeto podem ser encontrados no repositório do GitHub (Santos, 2023).

A seleção do *dataset* desempenhou um papel crucial na qualidade do treinamento. Foi empreendido um esforço para encontrar um *dataset* considerado adequado, caracterizado por um número substancial de imagens de placas de velocidade capturadas em diferentes ângulos.

Com o *dataset* definido, o treinamento do YOLO foi iniciado, seguido por uma verificação criteriosa dos resultados. A métrica adotada foi a obtenção de uma taxa de acerto superior a 90% para a base de teste. Caso esse critério não fosse atendido, ajustes no *dataset* eram realizados ou, em casos mais extremos, a troca por um *dataset* mais apropriado.

Para simular situações do mundo real, imagens de placas de velocidade foram obtidas, algumas de autoria própria e outras compartilhadas por conhecidos. A eficácia da rede pode então ser avaliada para essas imagens fora do conjunto de treinamento. Caso fosse necessário, o treinamento era refeito, ajustando o *dataset* até alcançar uma taxa de acerto igual ou superior a 80% para essas novas imagens, garantindo uma abordagem robusta e generalizável.

3.1 Métodos aplicados

3.1.1 Treinamento de uma rede neural com YOLO

Para iniciar o treinamento de detecção de objetos em uma rede YOLO, é crucial contar com um *dataset* substancial, composto por um conjunto significativo de imagens que apresentem o objeto de interesse. Cada objeto é descrito através de arquivos de texto, que contêm informações sobre as *bounding boxes* (caixas delimitadoras dos objetos de interesse) associadas a cada imagem, além de conter a classe da imagem, um número que refere a imagem a um conjunto específico, como por exemplo “placa de 20 km/h” ou “placa de 100 km/h”.

Esse *dataset* é então dividido em dois grupos distintos: o grupo de treinamento e o grupo de teste. Essa divisão, realizada de maneira aleatória a partir de um *script* em *python* disponível no repositório do GitHub (Santos, 2023), destina 70% das imagens para o grupo de treinamento, enquanto reserva os 30% restantes para o grupo de teste.

Durante o treinamento, a rede YOLO utiliza o grupo de treinamento para ajustar seus parâmetros internos, buscando aprender padrões e características associados aos objetos presentes nas imagens. As *bounding boxes* fornecidas neste estágio inicial são essenciais para guiar a rede durante o processo de aprendizado.

Após a fase de treinamento, a rede é avaliada quanto à sua capacidade de generalização no grupo de teste. Neste estágio, a YOLO é desafiada a identificar corretamente os objetos nas imagens de teste, utilizando as *bounding boxes* fornecidas anteriormente. Essa etapa é crucial para verificar se a rede consegue aplicar eficazmente o conhecimento adquirido durante o treinamento a novas instâncias.

Essa abordagem iterativa de treinamento e teste é fundamental para aprimorar o desempenho da YOLO na detecção de objetos, permitindo que a rede se adapte a uma variedade de situações e contextos.

O treinamento é encerrado quando o número máximo de iterações definido pelo usuário é alcançado, ou quando a precisão mínima do treinamento é atingida.

3.1.2 Configurando o *darknet* para treinamento de objetos no YOLOv4

O treinamento de objetos com o YOLOv4 no *darknet* envolve a configuração de diversos parâmetros essenciais para o sucesso do processo. Essas configurações, em conjunto, definem a arquitetura da rede, as classes a serem detectadas e os caminhos para os dados de treinamento e teste.

O arquivo central nesse processo é o yolov4.cfg, que descreve a arquitetura específica da rede, incluindo o número de camadas, filtros, e outras características. Algumas alterações foram feitas no arquivo de configurações, seguindo o tutorial de treinamento utilizado como base e as indicações de Joseph Redmon (2016).

O valor da linha *batches* foi alterado para 64, A linha *batches* nas configurações do YOLO especifica o número de lotes de dados que serão processados antes de fazer uma atualização de peso durante o treinamento da rede neural. Um lote (*batch*) é um conjunto de exemplos de treinamento que são processados em conjunto para calcular o gradiente e atualizar os pesos da rede (Redmon, 2016). Aumentar o valor de *batches* para 64 significa que a rede neural irá processar 64 exemplos de treinamento antes de atualizar os pesos, Isso pode ajudar no treinamento, pois permite que a rede veja mais exemplos de uma só vez, o que pode levar a uma convergência mais rápida e a uma melhor generalização do modelo, porém exige uma maior utilização de memória.

A linha *subdivision* teve seu valor modificado para 16, segundo Redmon (2016) *subdivisions* especifica quantas subdivisões serão feitas dentro de cada lote durante o treinamento. Essas subdivisões permitem que o processamento dentro de cada lote seja distribuído ao longo de várias etapas, o que pode ajudar a reduzir a quantidade de memória necessária durante o treinamento.

o valor de *max_batches* foi calculado como sendo 26000, limitando o número máximo de iterações da rede. *Steps* tem seus valores de 80% e 90% de *max_batches, steps* referem-se aos pontos durante o treinamento nos quais ocorrem ajustes na taxa de aprendizado (learning rate). A taxa de aprendizado controla o quão rápido o modelo de rede neural ajusta seus pesos durante o treinamento. Esses valores são escolhidos para garantir que os ajustes na taxa de aprendizado ocorram em estágios específicos do treinamento, o que pode ajudar a estabilizar o processo de otimização e melhorar a convergência do modelo (Redmon, 2016).

O tamanho *width e height* foi definido para 416, definir o tamanho de entrada para 416x416 pixels é uma escolha prática que oferece um bom equilíbrio entre desempenho, precisão e compatibilidade com hardware e software (Redmon, 2016).

A linha *filters* foi alterada de acordo com a fórmula *filters*=((número de classes) + 5)x3 em todas as 3 camadas de convolução (Redmon, 2016). A linha *filters* refere-se ao número de filtros utilizados em cada camada convolucional. Os filtros são os neurônios em uma camada convolucional que são responsáveis por detectar padrões específicos nos dados de entrada, como bordas, texturas ou formas.

Paralelamente, o arquivo “obj.names” lista os nomes das classes a serem detectadas, fornecendo ao modelo a compreensão das categorias que deve identificar.

A preparação dos dados é realizada através dos arquivos “train.txt” e “test.txt”, que enumeram os caminhos para as imagens de treinamento e teste, respectivamente. Esses arquivos orientam o *darknet* sobre quais dados usar em cada fase do treinamento. Adicionalmente, o arquivo “obj.data” contém informações cruciais sobre o conjunto de dados, como o caminho para o arquivo de nomes de classes, caminhos para os conjuntos de treinamento e teste, entre outros.

No início do treinamento, o *darknet* pôde inicializar a rede com pesos pré-treinados, proporcionando uma breve inicialização. O arquivo de pesos pré-treinados, usualmente denominado “yolov4.weights”, é vital para esse propósito.

Durante o processo de treinamento, um arquivo de configuração específico, como o “yolov4_training.cfg”, baseado no arquivo principal “yolov4.cfg”, é ajustado para atender aos requisitos específicos do treinamento, incluindo o número de classes. O *script* de treinamento, implementado pelo comando *darknet detector train*, incorpora todas essas configurações e inicia o treinamento iterativo, onde os pesos da rede são refinados para otimizar a precisão das detecções.

Essa abordagem abrangente das configurações no *darknet* fornece uma base sólida para o treinamento eficaz do YOLOv4, garantindo que o modelo esteja alinhado com as características do conjunto de dados específico, resultando em detecções precisas e generalizáveis.

3.1.3 Considerações sobre o uso do Google Colab para realizar o treinamento

O treinamento da rede YOLOv4 no Google Colab foi uma escolha estratégica para este projeto, fundamentada na gratuidade do ambiente e na disponibilidade de uma GPU potente, que acelera significativamente o processo de aprendizado da rede neural, quando comparada aos outros computadores disponíveis para realizar o treinamento. No entanto, ao longo do desenvolvimento, algumas considerações relevantes emergiram, impactando a eficiência geral do uso do Google Colab para o treinamento de modelos complexos.

Um dos pontos positivos do Google Colab foi sua eficiência geral no treinamento, proporcionando uma experiência satisfatória no que diz respeito à rapidez e capacidade de processamento, os treinamentos em sua versão gratuita demoraram de duas a três semanas dependendo do tamanho do *dataset* utilizado. A utilização de uma GPU poderosa, mesmo em sua versão gratuita, permitiu realizar treinamentos de forma acessível.

Contudo, a limitação de uso diário do Google Colab tornou-se uma preocupação à medida que foram realizados diversos treinamentos para aprimorar os resultados. Em algumas etapas, tornou-se necessário contratar o serviço para obter acesso a mais unidades computacionais, acelerando significativamente o processo de aprendizado da rede. A ausência de uma clareza quanto ao consumo

de unidades computacionais durante a execução do código gerou uma incerteza em relação ao término do plano contratado.

Outro ponto positivo do Colab Pro é a possibilidade do treinamento continuar funcionando mesmo se o ambiente de execução do Colab for desconectado, um problema que ocorreu frequentemente.

Outro desafio enfrentado foi a atualização periódica das bibliotecas no Google Colab. Ao longo dos meses de utilização, observou-se que as versões das bibliotecas eram atualizadas regularmente. Isso gerou alguns contratempos, como por exemplo quando a versão do CUDA foi atualizada, deixando de ser compatível com a versão específica do *darknet* que estava sendo utilizada para o treinamento.

Para contornar esse desafio, foram implementadas verificações no código para garantir a instalação das versões corretas das bibliotecas utilizadas, toda vez que o ambiente de desenvolvimento do Colab é iniciado. Essa abordagem permitiu manter a compatibilidade entre as diversas partes do ambiente de desenvolvimento, minimizando os impactos negativos causados por atualizações não planejadas.

Em resumo, o Google Colab se mostrou uma ferramenta valiosa para o treinamento do YOLOv4, proporcionando acesso a recursos computacionais robustos de maneira gratuita. No entanto, é importante estar ciente das limitações diárias e das potenciais questões de compatibilidade que podem surgir devido às atualizações regulares das bibliotecas. Adotar estratégias proativas, como verificação de versões e monitoramento do consumo de unidades computacionais, é essencial para otimizar a eficiência e garantir uma experiência de treinamento fluida e bem-sucedida.

3.1.4 Escolha do *dataset* e primeiros treinamentos

Para a construção deste trabalho de conclusão de curso, inicialmente, devido a falta de um *dataset* de placas de trânsito brasileiras, foram utilizados os arquivos disponíveis no *German Traffic Sign Recognition Benchmark* (GTSRB). Dado que este *dataset* organiza as imagens em pastas, em que cada representa uma classe específica de sinal de trânsito, facilitou o descarte das classes que não

correspondiam às placas de velocidade, sobrando desta forma apenas as classes de interesse deste trabalho.

Figura 3 – Exemplo de imagens presentes no *dataset* do GTSRB



Fonte: GTSRB (2019).

Após essa seleção, foram desenvolvidos códigos em *python*, disponíveis no repositório do GitHub (Santos, 2023), para excluir as *bounding boxes* dos arquivos csv, das classes que foram retiradas do treinamento, as *bounding boxes* ou caixas delimitadoras são responsáveis por marcar os objetos de interesse em uma imagem, para que o YOLO a partir desta indicação seja capaz de identificar os objetos que ele deve reconhecer, pós treinamento a rede deve ser capaz de gerear suas próprias *bounding boxes* quando identificar um objeto de interesse em uma imagem. Essa etapa foi crucial para refinar o conjunto de dados, concentrando-se apenas nas imagens de interesse.

Os arquivos de *bounding boxes* restantes foram então adaptados para atender ao padrão do YOLO, resultando na geração de um arquivo de texto para cada imagem. Esses arquivos foram nomeados conforme a imagem à qual se referiam, estabelecendo uma correspondência eficiente e necessária para o treinamento do YOLO.

O treinamento foi conduzido para oito classes de placas de velocidade sendo estas:

- 20km/h
- 30km/h
- 50km/h
- 60km/h
- 70km/h
- 80km/h
- 100km/h
- 120km/h

Vale ressaltar que, devido à restrição de uso diário na versão gratuita do Google Colab, o processo de treinamento demandou aproximadamente três semanas.

Nos testes realizados após o treinamento, observou-se que a rede apresentava dificuldades em detectar placas de velocidade em imagens de situações reais, não detectando quase nenhuma imagem que foi apresentada à rede. Após pesquisas adicionais, compreendeu-se que, apesar das imagens do *dataset* escolhido apresentarem, como visto na Figura 3, boas variações de ângulo, brilho, nitidez e de ser um *dataset* com bastante dados o YOLO requer imagens com um contexto amplo, indo além do objeto específico a ser detectado. Consequentemente, verificou-se que as imagens desse conjunto de dados não eram ideais para o treinamento do YOLO.

Figura 4 – Exemplo de imagem adequada para treinamento com YOLO



Fonte: Acervo próprio (2024).

A figura 4 mostra como seria uma imagem adequada para o treinamento da rede YOLO, mostrando também em destaque a *bounding box*. Diante disso, iniciou-se uma nova fase na pesquisa, buscando por um banco de dados mais adequado às exigências do YOLO.

3.1.5 *Dataset* definitivo e ajustes necessários para obtenção dos resultados desejados

O processo de aprimoramento do treinamento do modelo YOLOv4 envolveu a obtenção e refinamento de um *dataset* definitivo, bem como ajustes cruciais para otimizar o reconhecimento de placas de velocidade. Este estágio crítico do projeto abrange diversas etapas fundamentais.

3.1.5.1 Escolha do *dataset* e ajustes

Após a aquisição de um *dataset* contendo imagens de placas em contextos variados e com ampla variação de iluminação e posições diferentes, deu-se início a um novo treinamento. Assim como na etapa anterior, esse treinamento foi concebido para diversas classes de placas de velocidade, compreendendo limites como 30 km/h, 40 km/h, 50 km/h, 60 km/h, 70 km/h, 80 km/h, 90 km/h, 100 km/h, 110 km/h e 120 km/h. O valor das classes mudou do primeiro treinamento para este pois foi necessário adaptar-se às características do *dataset* utilizado

Figura 5 – Exemplo de imagem do novo *dataset*



Fonte: *Dataset*.

Uma medida essencial adotada neste estágio foi a exclusão de imagens que não continham placas de trânsito, além da remoção das *bounding boxes* correspondentes a carros e motos presentes no *dataset* original. Esse processo de limpeza foi vital para concentrar o treinamento na detecção específica das placas de velocidade, os códigos em *python* utilizados para realizar estes processos encontram-se em repositório no GitHub (Santos, 2016).

Embora este *dataset* tenha sido fundamental para o treinamento da rede YOLO, seu desempenho inicial ficou abaixo das expectativas. Durante os testes, foi

observado que algumas placas de velocidade eram reconhecidas, mas a precisão na identificação da velocidade em si era inconsistentemente baixa, variando entre 5% e 15%.

Diante deste cenário, conscientes de que o *dataset* utilizado era apropriado para o treinamento de uma rede YOLO, buscaram-se soluções para elevar a performance do modelo. A primeira estratégia adotada foi a implementação do *data augmentation*, uma técnica que visa diversificar o conjunto de treinamento através de modificações nas imagens existentes. Essa abordagem visa enriquecer a base de dados, proporcionando à rede uma maior capacidade de generalização diante de diferentes condições de iluminação, perspectivas e contextos.

3.1.5.2 Utilização do *data augmentation*

O *data augmentation* representa um passo importante na otimização do treinamento, permitindo que a rede neural se torne mais robusta e capaz de reconhecer placas de velocidade com maior precisão, independentemente das variações presentes nas imagens. Essa adaptação estratégica é fundamental para enfrentar os desafios reais de detecção em ambientes dinâmicos de tráfego.

Após análise sobre a aplicação de técnicas de *data augmentation*, concluiu-se que as alterações mais relevantes para o *dataset* de placas de velocidade seriam a adição de ruído na imagem, a alteração do tamanho e a modificação do brilho.

Para implementar essas alterações de maneira eficiente, foram desenvolvidos códigos em *python*, disponíveis no repositório do GitHub (Santos, 2023), capazes de gerar variações aleatórias nesses parâmetros. Ao combinar essas modificações ao *dataset* original, o tamanho do conjunto de dados foi quadruplicado. O *dataset* após a adição das novas imagens contém 5556 imagens.

Figura 6 – Exemplo de imagem após o processamento por *scripts* do *data augmentation*



Fonte: Acervo próprio.

Optou-se por não utilizar alterações de cor ou transformações geométricas, uma vez que foi considerado que essas mudanças seriam pouco úteis para placas em situações reais. Isso se deve ao fato de que as placas geralmente possuem um formato circular, com exterior vermelho e interior contendo um número sobre fundo branco.

Com a conclusão desse novo treinamento, foram realizados testes adicionais no banco de imagens, buscando validar a eficácia das modificações implementadas.

Apesar do novo treinamento, a taxa de acerto nas velocidades específicas permaneceu abaixo de 20%. Contudo, é importante ressaltar uma melhoria considerável na capacidade da rede em discernir se um objeto era ou não uma placa de velocidade. Nesse aspecto, alcançou-se uma taxa de acerto de 55%, demonstrando desta forma o ponto levantado anteriormente neste artigo, o qual foi descoberto neste ponto do trabalho, de que o YOLO é melhor reconhecendo objetos do que caracteres ou números.

Ao perceber que o reconhecimento das placas estava ocorrendo em 50% das imagens mostradas para a rede, identificou-se que a principal limitação residia

no reconhecimento dos números. Diante dessa constatação, optou-se por uma abordagem híbrida.

Para as placas em que o YOLO conseguia identificar o *bounding box*, este foi extraído, e o Tesseract OCR foi utilizado para reconhecer o número dentro do círculo da placa. Essa abordagem combinada resultou em um significativo avanço, com uma taxa de acerto para a velocidade de 76,38% para o conjunto de placas detectadas. Essa integração bem-sucedida evidencia a sinergia entre técnicas de visão computacional e reconhecimento óptico de caracteres, destacando-se como uma estratégia eficaz para superar desafios específicos encontrados durante o processo de desenvolvimento do modelo.

3.1.5.3 Treinamento final

Diante dos resultados obtidos, concluiu-se que uma abordagem mais eficaz para aumentar a taxa de acerto das placas de velocidade seria realizar um novo treinamento da rede YOLO. Nessa etapa, o *dataset* permaneceria o mesmo, porém, seria utilizado para treinar uma única classe denominada "placas de velocidade de trânsito". Esta estratégia visava direcionar o treinamento para a identificação específica das placas, enquanto a tarefa de reconhecimento das velocidades seria delegada ao Tesseract OCR.

Esta abordagem oferece vantagens significativas, eliminando a dependência das velocidades presentes no *dataset* original para treinamento, uma vez que tais classes não seriam mais necessárias. Além disso, a separação de responsabilidades entre o YOLO e o Tesseract proporcionou um ajuste mais refinado e especializado para cada etapa do processo de identificação de placas de velocidade. As figuras abaixo mostram um exemplo de como a rede trabalha para identificar uma placa.

Na Figura 7 temos a imagem original do banco para testes.

Figura 7 – Imagem de entrada de uma placa



Fonte: Acervo próprio.

Após receber a imagem a rede YOLO irá identificar a localização da placa, se existir uma placa, e marcar a *bounding box* correspondente a seu local, como visto na Figura 8.

Figura 8 – Imagem após identificação da *bounding box*



Fonte: Acervo próprio.

Feito a identificação o *script* irá recortar a imagem e entregará para o Tesseract OCR apenas a imagem da placa, que será tratada e então feito a

identificação de velocidade, como visto na Figura 9, onde o número abaixo da imagem é a velocidade identificada pelo sistema.

Figura 9 – Reconhecimento de velocidade de 60km/h identificada pelo sistema



Fonte: Acervo próprio.

Os resultados dessa nova abordagem serão detalhadamente explorados e discutidos na seção de Apresentação dos Resultados deste trabalho.

4 APRESENTAÇÃO DOS RESULTADOS

Concluída a etapa de treinamento para a classe "placas de velocidade de trânsito" com 26 mil iterações, os pesos resultantes foram aplicados ao *dataset* de validação, composto por 116 imagens de acervo próprio capturadas em diversas localizações, contendo placas de velocidades variadas. Este *datase*t, assim como o utilizado para o treinamento estão disponíveis no meu repositório do GitHub.

Os resultados desta validação estão apresentados na Tabela 1, que oferece uma visão abrangente da performance do modelo YOLO alinhado com o Tesseract OCR na identificação de placas de velocidade em situações diversas. Essa análise permitirá uma avaliação criteriosa dos resultados e embasará as conclusões finais deste trabalho.

Tabela 1 – Resultados do treinamento da rede de identificação de velocidade de placas de trânsito

	Acertos	Erros	Taxa Acertos (%)
20km/h	4	1	80,00%
30km/h	14	1	93,33%
40km/h	40	3	93,02%
50km/h	3	0	100,00%
60km/h	41	6	87,23%
70km/h	2	0	100,00%
80km/h	9	2	81,82%
90km/h	0	1	0,00%
100km/h	2	2	50,00%
110km/h	1	0	100,00%
Total	116	16	87,88%

A Tabela 1 mostra o resultado dos acertos conjuntos entre a identificação da placa do YOLO e a identificação do número correto pelo Tesseract, apenas quando os 2 casos geraram acerto o mesmo foi considerado, das 132 imagens 7 placas não foram reconhecidas pelo YOLO, desta forma o acerto da rede YOLO foi de 125 imagens em 132 possíveis, ou 94,69% de acerto.

As 125 placas que foram reconhecidas pelo YOLO foram analisadas pelo Tesseract, e o mesmo acertou a velocidade em 119 delas, gerando uma taxa de acerto de 95,2%.

Os resultados conjuntos entre as duas técnicas na identificação de placas de velocidade, como apresentados na Tabela 1, revelaram uma taxa de acerto global de 87,88%. Esse índice supera as expectativas estabelecidas, indicando um desempenho satisfatório do modelo YOLO na tarefa proposta.

É crucial notar que, apesar do resultado global positivo, algumas velocidades específicas, como 90 km/h e 100 km/h, apresentaram taxas de acerto aparentemente inferiores. Contudo, é importante ressaltar que a amostragem dessas velocidades foi limitada, o que pode impactar a representatividade estatística dos resultados para essas categorias específicas.

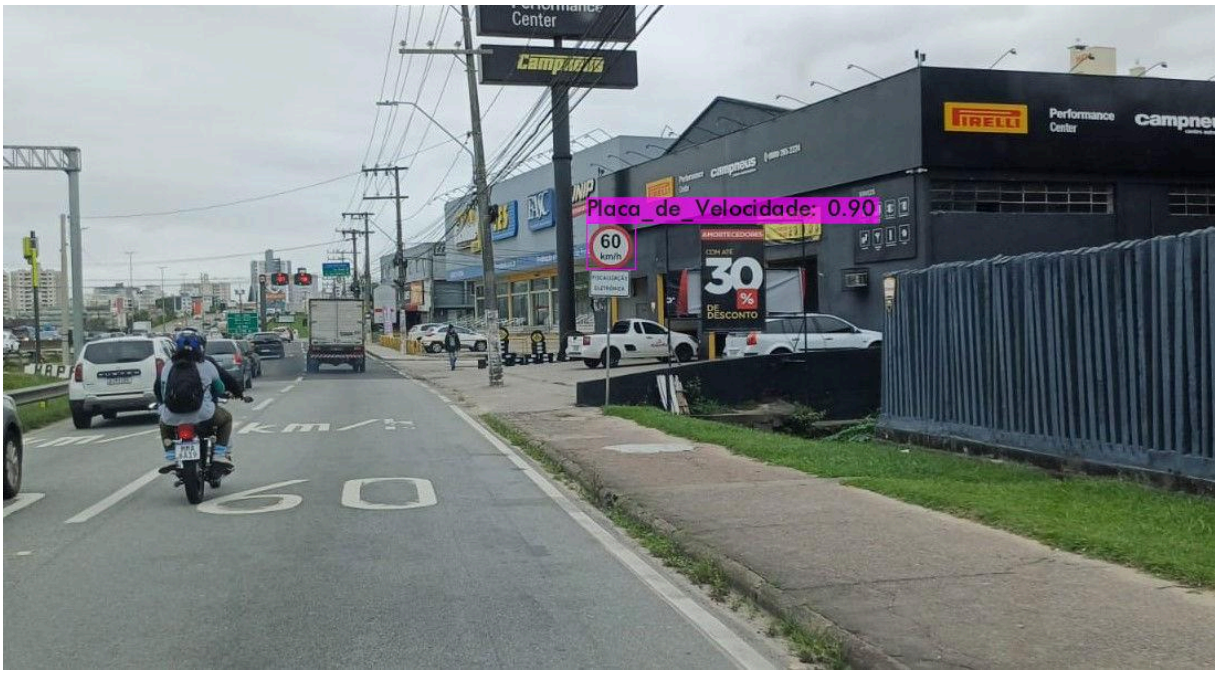
A fim de proporcionar uma análise mais detalhada e contextualizada, serão apresentados exemplos de identificação em situações mais desafiadoras, estes exemplos foram retirados do *dataset* de 132 imagens utilizadas para validar os resultados como visto na Tabela 1. Estes incluirão ambientes urbanos com alta densidade visual, imagens noturnas, condições climáticas adversas e capturas em alta velocidade, onde a nitidez da imagem pode ser comprometida. Esses casos extremos permitirão uma avaliação mais abrangente da robustez do modelo em diferentes cenários.

Figura 10 – Identificação de placas em ambientes urbanos



Fonte: Acervo próprio.

Figura 11 – Identificação de placas em ambientes urbanos



Fonte: Acervo próprio.

Figura 12 – Identificação de placas em ambientes urbanos



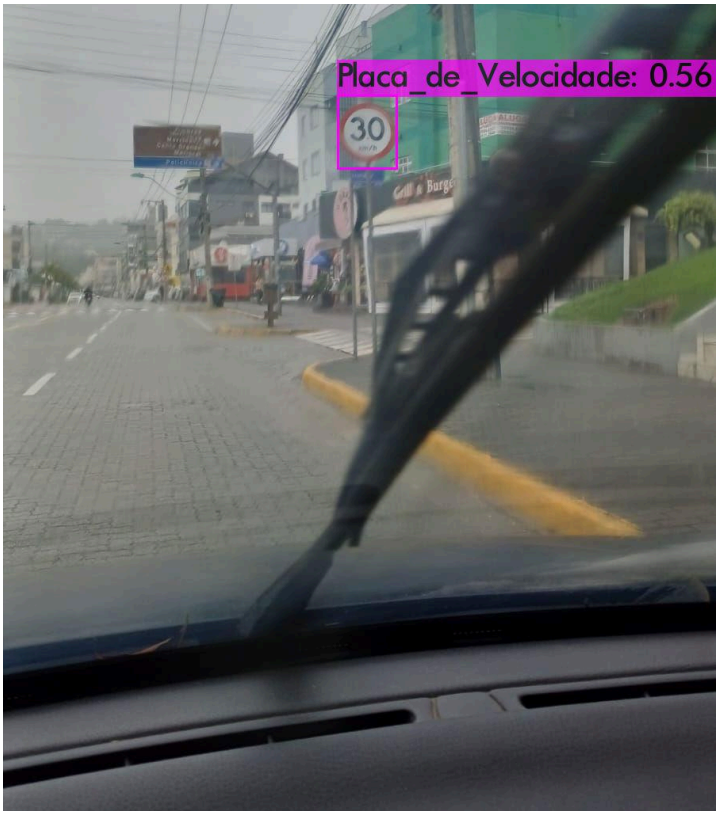
Fonte: Acervo próprio.

Figura 13 – Identificação de placas em ambientes de baixa nitidez



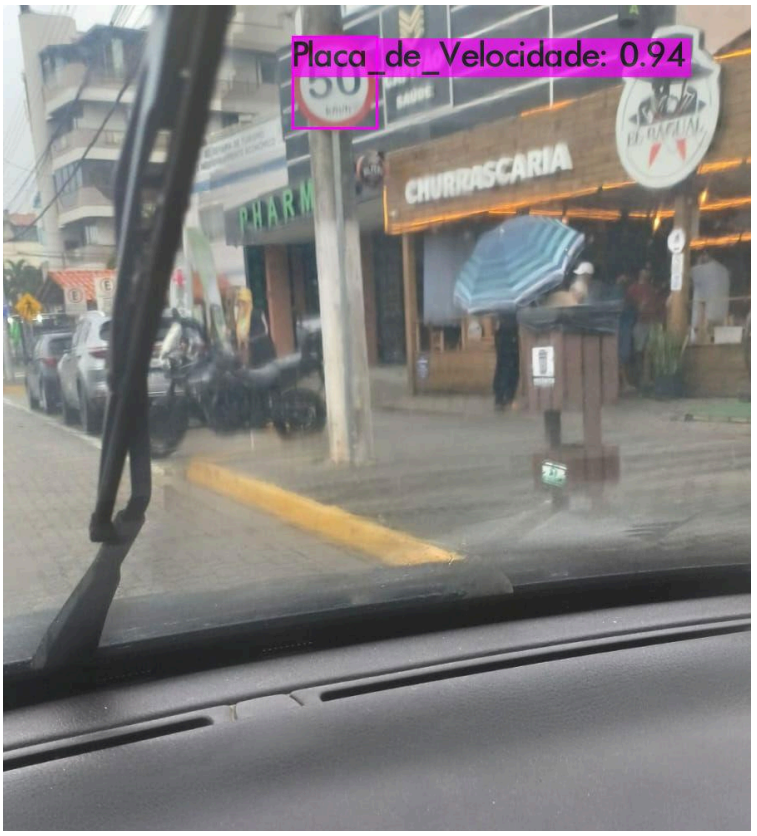
Fonte: Acervo próprio.

Figura 14 – Identificação de placas em ambientes de baixa nitidez



Fonte: Acervo próprio.

Figura 15 – Identificação de placas em ambientes de baixa nitidez



Fonte: Acervo próprio.

Figura 16 – Identificação de placas em ambiente noturno



Fonte: Acervo próprio.

Figura 17 – Identificação de placas em ambiente noturno



Fonte: Acervo próprio.

Figura 18 – Identificação de placas com coloração danificada



Fonte: Acervo próprio.

A análise visual dos resultados proporciona *insights* valiosos sobre a eficácia do modelo YOLO no reconhecimento de placas de velocidade em diversas condições desafiadoras. Os exemplos apresentados abrangem cenários urbanos complexos, imagens noturnas, condições climáticas adversas e situações de alta velocidade, demonstrando a capacidade do modelo de generalização em ambientes diversificados.

É importante destacar que todas as placas identificadas nas imagens foram corretamente reconhecidas em termos de velocidade pelo Tesseract OCR. Em alguns casos, como na Figura 19, embora o YOLO tenha identificado a placa, parte dela estava parcialmente coberta, impossibilitando a leitura da velocidade pelo Tesseract.

Figura 19 – Identificação de placa parcialmente coberta



Fonte: Acervo próprio.

Outro ponto relevante é que a qualidade mínima da imagem é crucial para garantir uma identificação precisa. A distância entre a câmera e a placa deve ser suficiente para que o Tesseract possa realizar a leitura do recorte da imagem, o que pode impactar a identificação em distâncias maiores.

Por fim, é válido mencionar a ocorrência de alguns falsos positivos para outros tipos de placas que apresentam o círculo vermelho no contorno, como no caso de placas de ciclovia. No entanto, esses casos são facilmente descartados, uma vez que o Tesseract não identificará um número de velocidade nesses contextos específicos.

Figura 20 – Falso positivo em placa de ciclovia



Fonte: Acervo próprio.

5 CONSIDERAÇÕES FINAIS

O presente trabalho buscou desenvolver e aprimorar um sistema de detecção e reconhecimento de placas de velocidade por meio da combinação do modelo YOLO (You Only Look Once) e do Tesseract OCR. A proposta envolveu um processo iterativo de treinamento, ajustes e validação, visando alcançar resultados de acerto da velocidade acima de 80% para condições e ambientes variados.

Considerando os resultados obtidos neste trabalho, pode-se concluir que o uso do *framework* YOLO para o reconhecimento de placas de velocidade em imagens de trânsito é uma abordagem promissora. Os resultados obtidos, representados pela taxa de acerto global de 87,88%, refletem o sucesso da abordagem adotada.

A análise visual demonstrou que o YOLO foi capaz de identificar placas de velocidade em diferentes contextos, desde ambientes urbanos complexos até condições noturnas desafiadoras. A integração do Tesseract OCR para o reconhecimento preciso das velocidades contribuiu para a robustez do sistema.

Ainda, algumas limitações foram identificadas, como a necessidade de imagens com qualidade mínima para garantir a leitura adequada pelo Tesseract e a ocorrência de falsos positivos em casos específicos de placas com círculo vermelho no contorno.

Diante dos resultados obtidos e das lições aprendidas durante o desenvolvimento, este trabalho fornece uma contribuição relevante para o campo de visão computacional em sistemas de assistência ao condutor. As considerações aqui apresentadas servem como base para futuras pesquisas e melhorias na detecção e reconhecimento de placas de velocidade, visando aprimorar a segurança e eficiência dos sistemas de transporte.

5.1 Sugestões para trabalhos futuros

Considerando o desenvolvimento deste estudo, algumas sugestões para trabalhos futuros podem ser definidos. Primeiramente, seria valioso implementar o

sistema em um veículo real para conduzir testes em tempo real. Essa abordagem permitiria avaliar a eficácia do modelo em condições dinâmicas de tráfego, proporcionando desdobramentos importantes sobre sua aplicabilidade prática.

Outra frente de aprimoramento seria o refinamento do treinamento da rede YOLO para mitigar falsos positivos. Investigar e ajustar parâmetros específicos, diversificar o *dataset* ou realizar modificações na arquitetura da rede são estratégias que podem contribuir para a redução desses falsos positivos.

Além disso, sugere-se explorar a implementação de outras técnicas de processamento de imagem. A incorporação de abordagens avançadas, como filtragem de ruído, aprimoramento de bordas e técnicas de pré-processamento, pode complementar as capacidades do YOLO, melhorando a qualidade e a precisão dos resultados.

Essas sugestões visam não apenas ampliar as capacidades do sistema, mas também abrir oportunidades para investigações mais aprofundadas e avanços na detecção de placas de velocidade em ambientes automotivos.

REFERÊNCIAS

Autor desconhecido. ***Dataset* imagens de placas de velocidade de trânsito definitivo.** Disponível em: <https://drive.google.com/drive/folders/1DPidfxzpmKuIqv1Wct9GgNSM4o82Npjc?usp=sharing>.

BOCHKOVSKIY, Alexey; WANG, Chien-Yao; LIAO Hong-Yuan Mark. ***YOLOv4: Optimal Speed and Accuracy of Object Detection***. Disponível em: <https://arxiv.org/pdf/2004.10934.pdf>.

Brasil. Ministério da Infraestrutura. **Registro Nacional de Acidentes e Estatísticas de Trânsito.** Governo Federal, 2023. Disponível em: <https://www.gov.br/transportes/pt-br/assuntos/transito/conteudo-Senatran/registro-nacional-de-acidentes-e-estatisticas-de-transito>. Acesso em: 20 de dezembro de 2023.

BROOKHUIS, Karel A.; DE WAARD, Dick; JANSSEN, Wiel H. ***Behavioural impacts of Advanced Driver Assistance Systems–an overview. European Journal of Transport and Infrastructure Research***, [S. l.], v. 1, n. 3, 2001. DOI: 10.18757/ejtir.2001.1.3.3667. Disponível em: <https://journals.open.tudelft.nl/ejtir/article/view/3667>.

CUDA *Toolkit*. Disponível em: <https://developer.nvidia.com/cuda-toolkit>.

DIXIT, A.; KUMAR CHIDAMBARAM, R.; ALLAM, Z. ***Safety and Risk Analysis of Autonomous Vehicles Using Computer Vision and Neural Networks***. v. 3, p. 595-617, 2021. Disponível em: <https://www.mdpi.com/2624-8921/3/3/36>. Acesso em: 05 de Janeiro de 2024.

DOS SANTOS, Enzzo Comassetto. ***Dataset* para testes**. Disponível em: <https://drive.google.com/drive/folders/1U-oWWy6heRA4tHpWJ6aHV60CE_AQhdRU?usp=sharing>.

DOS SANTOS, Enzzo Comassetto. **Repositório GitHub**. Disponível em: <https://github.com/enzzocs/TCC/tree/main>.

DOS SANTOS, Enzzo Comassetto. **Treinamento no Google Colab**. Disponível em: <https://colab.research.google.com/drive/10pQKv0Q8DJJuA6RmiXAPbl2af3ryR4zu#scrollTo=y0HfX7gsNP6_>.

Google. **Colaboratory. [Colab]**. Disponível em: https://colab.research.google.com/. Acesso em: 02 jan. 2024.

GTSRB - German Traffic Sign Recognition Benchmark. Disponível em: <https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign> Acesso em: 17 de abril de 2023.

Organização Pan-Americana da Saúde (OPAS). **Segurança no Trânsito**. OPAS, 2023. Disponível em: <https://www.paho.org/pt/topicos/seguranca-no-transito>. Acesso em: 21 de dezembro de 2023.

Poorani, G. et al. **Number Plate Detection Using YOLOV4 and Tesseract OCR**. *Journal of Pharmaceutical Negative Results*, v. 13, Special Issue 3, 2022. Disponível em: <https://pnrjournal.com/index.php/home/article/view/426/297>. Acesso em: 04 de novembro de 2023.

REDMON, Joseph; et al. ***YOLO: Real-Time Object Detection.*** 2016. Disponível em: <https://pjreddie.com/darknet/yolo/>.

REDMON, Joseph; et al. ***You Only Look Once:Unified, Real-Time Object Detection***. Disponível em: <https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf>.

SAHA, Sumit. ***A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way***. Disponível em: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53> Acesso em: 02 de janeiro de 2024.

SENATRAN. **Anuário SENATAN 2021**, 2021. Disponível em: <https://www.gov.br/transportes/pt-br/assuntos/transito/conteudo-Senatran/estatisticas-senatran/anuariosenatran2021.pdf/view>. Acesso em: 19 de dezembro de 2023.

SO, Jian et al. ***The Practical Effectiveness of Advanced Driver Assistance Systems at Different Roadway Facilities: System Limitation, Adoption, and Usage***. IEEE Transactions on Intelligent Transportation Systems, v. 21, n. 9, p. 3859-3871, set. 2020. DOI: 10.1109/TITS.2019.2959255.

Techzizou. ***Train a custom YOLOv4 detector using Google Colab [Tutorial for Beginners]***. Disponível em: <https://techzizou.com/train-a-custom-yolov4-detector-using-google-colab-tutorial-for-beginners/>. Acesso em: 16 de julho de 2023.

Train a custom YOLOv4 object detector. Disponível em: <https://colab.research.google.com/drive/1zqRb08ljHvIIMR4fgAXeNy1kUtjDU85B?usp=sharing>.

ANEXOS

ANEXO A – Fórmulas utilizadas

Taxa de acerto (%)	$\frac{Total_{acertos}}{Total_{amostras}}$
Total de amostras	$Total_{acertos} + Total_{erros}$
filters	$(Numero_{classes} + 5)\ x\ 3$