ALGORITMO DE IDENTIFICAÇÃO DE OBSTÁCULOS NA VIA JUNTO A SINALIZAÇÃO DE TRÂNSITO

ALGORITHM FOR IDENTIFYING OBSTACLES ON THE ROAD NEXT TO TRAFFIC SIGNS

Luiz Filipe da Silva Groff[1], Lucas Kenji Sasaki Yashima[2], Jefferson Antonio Ribeiro Passerini[3]

[1]Fundação Educacional de Fernandópolis, luizgroff@fef.edu.br
[2] Fundação Educacional de Fernandópolis, lucasyashima@fef.edu.br
[3] Fundação Educacional de Fernandópolis, jefferson.passerini@fef.edu.br

Área: Informação e Identificação Visual
Subárea: Matemática e Inteligência Computacional

RESUMO

O objetivo principal deste trabalho foi desenvolver uma solução inovadora baseada em inteligência artificial e visão computacional para detectar buracos em vias públicas, visando melhorar a segurança e a mobilidade das pessoas com deficiência visual. A pesquisa foca na criação de um sistema que é capaz de identificar e classificar esses obstáculos de forma eficaz e em tempo real, permitindo que as pessoas com deficiência visual possam se locomover com mais segurança em ambientes urbanos. Esse sistema foi projetado para ser acessível, robusto e capaz de operar em diferentes tipos de condições, como variações nas superfícies das vias e nas condições de iluminação. A detecção precisa dos buracos e a entrega de alertas em tempo real foram elementos essenciais, pois permitem que o usuário seja informado antecipadamente sobre os perigos, evitando assim acidentes e melhorando a qualidade de vida dos indivíduos que dependem de uma navegação segura.

Palavras-chave: Inteligência Artificial, Visão Computacional, Detecção de Buracos, Deficiência Visual, Segurança

ABSTRACT

The main objective of this work was to develop an innovative solution based on artificial intelligence and computer vision to detect potholes on public roads, aiming to improve the safety and mobility of people with visual impairments. The research focus on creating a system that is effectively and in real time identify and classify these obstacles, allowing people with visual impairments to move more safely in urban environments. This system was designed to be accessible, robust, and capable of operating under various conditions, such as variations in road surfaces and lighting conditions. Accurate pothole detection and the delivery of real-time alerts were essential elements, as they enable users to be warned in advance of dangers, thereby preventing accidents and improving the quality of life for individuals who depend on safe navigation.

Keywords: Artificial Intelligence, Computer Vision, Pothole Detection, Visual Impairments, Safety.

1. INTRODUÇÃO

A inteligência artificial é uma ferramenta, com o intuito de simular o processamento de dados assim como a mente humana, priorizando um desempenho que se assemelhe ou até se torne mais elevado, foi desenvolvida com auxílio do aprendizado de máquina (machine learning) que envolve a inserção de dados ao algoritmo e a partir destes, realize o processamento necessário e retorne o resultado desejado com base na necessidade do usuário.

Como é citado por Russell e Norvig (2016), "inteligência artificial é a área da ciência da computação que se dedica a criar sistemas capazes de realizar tarefas que, se fossem realizadas por humanos, exigiria inteligência".

Esta ferramenta está em constante aprimoramento com o passar dos anos, de tal forma que hoje é possível resolver diversos problemas enfrentados pela sociedade, podendo ser citados casos de relevância menor, como utilização para preparação de alimentos baseado nos recursos que encontram-se disponíveis, e alcançando o auxílio em casos que determinam uma relevância maior como a detecção de anomalias neurológicas através do uso de imagens fornecendo os devidos diagnósticos para identificação da doença. Segundo Goodfellow, Bengio e Courville (2016), "o aprendizado profundo permite a resolução de problemas complexos, como reconhecimento de padrões e a detecção de anomalias, que são extremamente valiosos em áreas como a medicina".

Porém para a utilização dela é necessário o conhecimento e afirmação de alguns conceitos, como por exemplo a rede neural, que tem como exemplo a funcionalidade de vários cérebros que colaboram e trabalham junto a fim de determinar um resultado que tenha mais assertividade. Como LeCun, Bengio e Hinton (2015) descrevem, "as redes neurais são compostas por múltiplas camadas de processamento que operam em paralelo, permitindo que um sistema aprenda a identificar padrões de forma muito mais eficiente". Por meio das redes neurais convolucionais que serão utilizadas para dar o aprendizado de máquina ao nosso algoritmo, conseguiremos determinar nas imagens encontradas, o que é considerado uma obstrução e a partir deste mecanismo será informado ao usuário determinado alerta.

De acordo com uma pesquisa do Datafolha realizada em 2024, 84% dos moradores da cidade de São Paulo reclamam de buracos no asfalto. Ou seja, a maioria da população da capital paulista reconhece que passam por problemas diariamente envolvendo obstruções nas ruas, principalmente pessoas cegas que acabam tendo dificuldades para realizar travessias em ruas ou avenidas que possuem problemas em sua estrutura.

Segundo o IBGE de 2023, o Brasil tem 6.5 milhões de pessoas com deficiência visual, dos quais 582 mil são cegas, com estes dados em mente e estudando a vasta entrega de soluções que a I.A tem a oferecer, surgiu a necessidade de entregar o auxílio na locomoção às pessoas que possuem uma incapacidade visual. Com isso, o nosso projeto visa a implementação de um algoritmo que por meio da inteligência artificial possibilita a detecção de obstruções nas sinalizações de trânsito, sendo possível auxiliar as pessoas que possuem limitações visuais. De tal forma, ao detectar essa obstrução, a pessoa que porta a deficiência consiga ter uma consciência maior dos obstáculos que se encontram em seu caminho.

2. REFERENCIAL TEÓRICO

Esta seção tem como objetivo realizar a introdução dos conceitos essenciais para um melhor esclarecimento da metodologia proposta, conectando a informações sobre o processamento de linguagem natural, assim como os modelos e algoritmos empregados neste trabalho.

2.1 APRENDIZADO DE MÁQUINA

A sociedade vem passando por constante evolução e as máquinas, assim como os humanos, também estão passando por essa transformação, e, um dos meios de evolução das máquinas é por meio do aprendizado, entrando principalmente no aprendizado de máquina.

Aprendizado de máquina tem como significado ser um segundo campo da inteligência artificial, onde é possível aprender dados, e por meio desses dados, identificar padrões para estar mais aptos às tarefas que lhe foram programadas. Segundo Mueller e Massaron (2018), o aprendizado de máquina é um conjunto de algoritmos capazes de analisar um número elevado de dados, também capaz de realizar as análises preditivas em velocidades superiores aos humanos.

O objetivo do Aprendizado de Máquina (AM) é a construção de programas que melhorem seu desempenho por meio de exemplos (Mitchell, 1997). E por meio do aprendizado de máquina, foi possível avançar ao nível da inteligência artificial nos dias de hoje. O aumento da capacidade dos computadores atuais é parcialmente em razão das técnicas de Aprendizado de Máquina. Entretanto, não é de hoje que se deseja fazer que o computador aprenda. Por exemplo, Alan Turing, o pai da computação, desenvolveu um teste, conhecido como teste de Turing, para saber se os computadores eram capazes de aprender. (Ludermir, 2021)

Paralelamente, à medida que os estudos para o aprendizado de máquina continuavam a crescer, e a possibilidade da inteligência artificial se tornar cada vez mais próxima, houve um aumento significante de pessoas com deficiências, segundo Leite e Silva (2006) deficiência é a limitação ou perda da capacidade de interagir com a sociedade de forma igualitária. conforme dados do IBGE na cartilha do censo demográfico de 2010 a deficiência visual apresentou-se com a maior ocorrência, afetando cerca de 18,6% da população brasileira.

2.2 REDE NEURAL CONVOLUCIONAL

Os classificadores são as técnicas de classificação ou rotulação de imagens, com base nas características das figuras apresentadas. A classificação de imagens é baseada em princípios de aprendizagem de máquina, de modo que os algoritmos são treinados para identificar os padrões específicos das imagens que estão disponíveis na base de dados. A classificação de imagens é extremamente relevante em diversos setores, como, por exemplo, no setor de segurança, onde a classificação de imagens é usada para identificar rostos de pessoas ou até mesmo para identificar sinalizações de trânsito nos carros. Existem diversos modelos de algoritmos para classificar imagens, sendo os mais utilizados *Random Florest*, *Support Vector Machines* (SVM) e Redes Neurais Convolucionais (CNN).

A Rede Neural Convolucional (CNN) é um modelo de aprendizado profundo (*Deep Learning*) que consegue reconhecer uma imagem que será lida e busca atribuir importância para

ela, através da diferenciação de alguns aspectos (brilho, cor, tamanho, entre outros) e objetos presentes na imagem e com isso realizar o reconhecimento do objeto identificado em questão (LI et al., 2021).

"O método de classificação deve ser selecionado de acordo com as características espectrais da imagem de sensoriamento remoto e os requisitos reais de trabalho" (FAGUNDES, Wendell S.; JÚNIOR, Mauro José A. 2022).

Abordando mais sobre as redes neurais, elas têm como princípio básico o recebimento de dados, o processamento e o resultado, a convolução é a análise dos dados separadamente como se fossem vistos parte por parte. Com o conceito básico marcado como concluído, é necessário ter o entendimento das camadas da CNN (*Convolutional Neural Network*), que são divididas em 3.

"Os blocos básicos de construção de CNNs envolvem operações de convolução, operadores de *pooling* (*downsampling*), funções de ativação e camadas totalmente conectadas" (PONTI, M. et al., 2017). A primeira camada, consiste em ser a lupa para o auxílio na detecção de texturas ou padrões, que são conhecidas como camadas convolucionais, a segunda é conhecida como camada de *pooling* que atua fazendo a identificação dos detalhes importantes descartando o que é irrelevante e por último, as camadas densas que consistem em tomar a decisão e dar o veredito final a respeito da classificação da imagem.

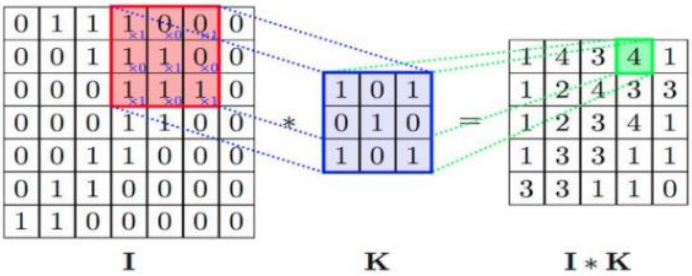
Conforme Sanvezzo (2022) Estas três camadas trazem o diferencial dos modelos CNN frente aos MLP, e são responsáveis por extrair características locais (ou local features) como bordas, contornos, formas e intensidade de cor.

Segundo Chenna (2023) as Redes Neurais Convolucionais (CNN) têm demonstrado desempenho de ponta em vários desafios e conjuntos de dados de processamento de computador e imagens. Isso levou as CNNs a serem amplamente utilizadas em aplicações como classificação de imagens, super resolução, segmentação e detecção de objetos.

Assim concluindo que o método de redes neurais se prova mais conciso para o estudo e interpretação de imagens e estudo de padrões.

Um exemplo de convolução ocorre quando um filtro (ou kernel) é aplicado a uma imagem para detectar características específicas, como suas bordas, contanto que uma imagem de 5x5 seja processada com um filtro de 3x3. A convolução envolve a sobreposição do filtro em cada posição possível da imagem, realizando uma multiplicação ponto a ponto entre o filtro e a parte correspondente da imagem, seguida por uma soma dos resultados. Esse processo é repetido em toda a imagem, gerando uma nova imagem (ou mapa de características) com os valores resultantes. O conceito foi amplamente discutido por LeCun et al. (1989) em seu trabalho seminal sobre redes neurais convolucionais para reconhecimento de dígitos, onde introduziram a ideia de aplicar convoluções para capturar informações espaciais de forma eficiente em imagens (LeCun, Y., et al, 1989).

Figura 1 – Exemplo de convolução.



Fonte: Pablo de Abreu Vieira.

2.3 *YOU ONLY LOOK ONCE* - YOLO

O Yolo (*You only look once*) é uma ferramenta de visão computacional de detecção precisa de objetos em imagens, que foi inicialmente lançado em 2015 e desenvolvido por Joseph Redmon e Ali Farhadi, ficou muito conhecido por sua velocidade e precisão que na época se mostrava superior às demais ferramentas, enquanto a maioria das técnicas populares da época levavam cerca de 0.5 segundos ou mais para processar a imagem, o Yolo conseguia fazer o mesmo em 50 milissegundos, permitindo o processamento de imagens em tempo real.

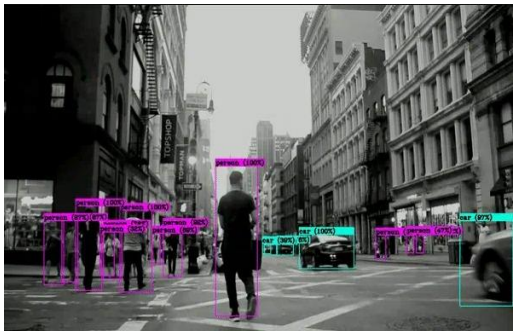
Como afirmam Redmon e Farhadi (2015), "a arquitetura do YOLO foi projetada para ser extremamente rápida, processando imagens em tempo real, o que a torna ideal para aplicações como vigilância e direção autônoma". Além desses aspectos, a ferramenta possui código aberto, tornando o seu acesso mais simples e acessível para qualquer pessoa.

A YOLO utiliza uma rede única, onde ela recebe uma imagem de entrada e ela é treinada totalmente de forma que as caixas delimitadoras são previstas em conjunto com as possíveis classes para cada uma delas. Portanto, por fazer o uso de apenas uma rede para realizar o processo, a YOLO consegue obter uma maior velocidade no seu processo e assim sendo preferido para detecção de objetos em tempo real (HUANG; PEDOEEM; CHEN, 2018).

A YOLO divide a imagem em uma rede de grade SxS, onde em cada uma dessas redes são retiradas caixas delimitadoras (REDMON et al., 2016). Para cada caixa delimitadora é feita a análise da probabilidade da existência de determinado objeto dentro dela ou não

As camadas convolucionais da CNN do modelo YOLO foram pré-treinadas na arquitetura de classificação de imagens *ImageNet*, onde nesse pré-treinamento as imagens de entrada foram treinadas com a metade da sua resolução (REDMON et al., 2016).

Figura 2 – Exemplo de detecção do algoritmo



Fonte: IA Expert academy.

O YOLO trata a detecção de forma unificada e simultânea. Como observa Redmon et al. (2016), “ao contrário dos métodos convencionais que geram propostas de região e as classificam posteriormente, o YOLO transforma o problema de detecção em um problema de regressão” (p. 4), onde a rede aprende a prever diretamente as caixas delimitadoras e suas respectivas classes em uma única passada pela imagem. Esse design de arquitetura é o que permite ao YOLO ser tão rápido e eficiente.

Além disso, o YOLO é um modelo altamente escalável e capaz de ser adaptado a diferentes necessidades de processamento, devido à sua flexibilidade em relação ao tamanho da entrada da imagem e à quantidade de classes que pode detectar. O modelo foi testado em diversos cenários e mostrou-se eficaz em ambientes com várias condições desafiadoras, como imagens com pouca iluminação ou em situações em que os objetos aparecem parcialmente, como em cenas de trânsito com obstruções.

O YOLOv8 foi utilizado devido aos seus fundamentos significativos em relação à estabilidade e eficiência. Ele possui uma melhor detecção de objetos menores e aprimora o desempenho em tempo real, sendo mais robusto em condições de baixa luminosidade e ângulos complexos. Além disso, essa versão é mais flexível, permitindo personalizações para diferentes cenários, o que a torna uma escolha ideal para o trabalho em questão.

3. METODOLOGIA

A detecção de buracos em vias públicas é uma demanda identificada a partir de levantamentos de dados realizados junto a pessoas com deficiência visual. Esses indivíduos frequentemente enfrentam dificuldades ao transitar por ruas ou estradas em condições precárias, marcadas pela presença de buracos, o que compromete sua segurança e mobilidade. Nesse contexto, o presente trabalho tem como principal objetivo o desenvolvimento de um algoritmo robusto para a identificação de buracos em imagens de estradas.

A proposta baseia-se na utilização da arquitetura YOLOv8L (*You Only Look Once, versão 8 - Large*), amplamente reconhecida pela sua capacidade de realizar tarefas de detecção de objetos em tempo real. Essa arquitetura combina alta precisão e velocidade, características essenciais para problemas complexos como a identificação de buracos em imagens. O uso dessa tecnologia visa oferecer uma solução eficiente e acessível, contribuindo para a melhoria da mobilidade e segurança das pessoas, especialmente aquelas com deficiência visual, em ambientes urbanos e rurais. Como afirmam os desenvolvedores da **Ultralytics**, "YOLOv8

fornece uma combinação sem precedentes de precisão e desempenho em tempo real, atendendo às necessidades de aplicações em tempo real com alta demanda computacional" (Ultralytics, 2023).

Para o treinamento do modelo, utilizou-se a base de dados disponibilizada pelo Kaggle, que contém imagens rotuladas com a posição exata dos buracos identificados por meio de caixas delimitadoras (*bounding boxes*). A base de dados selecionada, é composta por 12.868 imagens capturadas em condições diversas, abrangendo variações meteorológicas, resoluções e tipos de estrada. Essas características tornaram o treinamento mais robusto e abrangente, aumentando a capacidade do modelo de generalizar os resultados.

“Imagens em alta definição projetadas especificamente para o avançado sistema de detecção de objetos YOLOv8. Abrangendo diversos terrenos, condições climáticas e cenários de iluminação, este conjunto de dados oferece uma profundidade incomparável de anomalias urbanas – buracos nas ruas”. (Dutta, 2023).

Todos os modelos desenvolvidos para esta pesquisa foram desenvolvidos na linguagem Python e de suas bibliotecas, sendo elas a OS, Ultralytics, Cv2 e a Matplotlib.

O treinamento foi realizado em três cenários distintos, utilizando 300, 80 e 20 épocas, com o objetivo de avaliar o impacto do número de épocas no desempenho final do modelo. Essa abordagem permitiu uma análise mais detalhada sobre a relação entre o tempo de treinamento e a eficiência do algoritmo.

Um aspecto relevante dessa base de dados é a rotulagem das imagens. O modelo YOLOv8L requer que os rótulos estejam em um formato específico, com coordenadas normalizadas. Assim, as caixas delimitadoras originais foram convertidas para um formato compatível, contendo as coordenadas normalizadas do centro da caixa, seguidas pela largura e altura, em proporção ao tamanho da imagem. Esse pré-processamento foi essencial para garantir a compatibilidade e eficácia do modelo durante as etapas de treinamento e validação.

O pré-processamento das imagens constitui uma etapa essencial no desenvolvimento do modelo, pois prepara os dados para que o aprendizado ocorra de maneira eficiente. Uma das etapas mais importantes desse processo foi o redimensionamento das imagens, ajustando todas para o formato 640x640 pixels. Esse tamanho foi escolhido por ser o ideal para o YOLOv8L, uma vez que esse modelo foi projetado para processar imagens em dimensões fixas, otimizando seu desempenho.

O YOLOv8L destaca-se como uma das versões mais estáveis da arquitetura YOLO atualmente, oferecendo alta precisão e rapidez. Essas características são cruciais para tarefas como a detecção de buracos ou obstruções em vias, nas quais é indispensável um modelo que combine precisão com a capacidade de realizar predições em tempo real.

A arquitetura do YOLOv8L é baseada em uma rede neural convolucional (CNN) que divide a imagem em uma grade de células. Para cada célula, o modelo realiza predições sobre a presença de objetos (neste caso, buracos), suas localizações (caixas delimitadoras) e as respectivas classes, conforme definido durante o treinamento.

Para a detecção de buraco o modelo foi ajustado para identificar exclusivamente uma classe ("Hole"), simplificando o processo de treinamento e avaliação. Um dos principais desafios enfrentados foi garantir a identificação precisa dos buracos mesmo em condições adversas, como baixa iluminação, ângulos incomuns ou buracos parcialmente preenchidos com água. Esses ajustes foram fundamentais para aprimorar a robustez e confiabilidade do modelo.

3.1 Métricas de Treinamento do YOLOv8L

O desempenho do modelo foi avaliado utilizando um conjunto abrangente de métricas que incluem train/box_loss, train/cls_loss, train/dfl_loss, precisão, recall, val/box_loss, val/cls_loss, val/dfl_loss, metrics/mAP50(B) e metrics/mAP50-95(B). Essas métricas têm como objetivo medir a eficácia do modelo na detecção e classificação de objetos, atribuindo maior relevância às predições com pontuações mais altas, que indicam uma melhor correspondência com os objetos reais.

As métricas train/box_loss e val/box_loss representam a perda associada à previsão das caixas delimitadoras (bounding boxes) durante o treinamento (train/box_loss) e a validação (val/box_loss). Estas métricas avaliam quão bem o modelo está aprendendo a posicionar as caixas delimitadoras em torno dos objetos. Perdas menores indicam que o modelo está ajustando com precisão as caixas em relação aos objetos reais.

As métricas train/cls_loss e val/cls_loss medem a perda relacionada à classificação dos objetos dentro das caixas delimitadoras, durante o treinamento (train/cls_loss) e a validação (val/cls_loss). Avaliam a capacidade do modelo de atribuir a classe correta a um objeto detectado, como "buraco". Pontuações mais baixas indicam um modelo mais preciso na classificação.

Por exemplo, em um estudo de detecção de comportamento animal, métricas como precisão e recall foram usadas para avaliar a eficiência, alcançando resultados elevados, como 99,9% de precisão e 99,2% de recall, o que destaca a capacidade do modelo em identificar corretamente os objetos em diferentes condições (Subedi et al., 2023).

As métricas *Train/dfl_loss* e *val/dfl_loss* referem-se à perda na previsão de distribuições de pontos finos (*fine-grained localization*), que impactam a qualidade da localização do objeto dentro da caixa, onde valores baixos indicam uma detecção mais precisa das bordas do objeto.

A precisão foi calculada como a proporção de verdadeiros positivos (detecções corretas do objeto alvo, neste caso, buracos) em relação ao total de detecções positivas realizadas pelo modelo. Já a *recall* avalia a proporção de verdadeiros positivos em relação ao número total de objetos presentes nas imagens, fornecendo uma visão sobre a sensibilidade do modelo.

A métrica metrics/mAP50(B) avalia a precisão considerando uma sobreposição mínima de 50% (IoU) entre a predição do modelo e a localização real do objeto, enquanto metrics/mAP50-95(B) oferece uma média ponderada das precisões em diferentes limiares de sobreposição, variando de 0,5 a 0,95, proporcionando uma análise mais detalhada e robusta do desempenho geral do modelo.

A métrica *Metrics*/mAP50(B) representa a média de precisão (*mean Average Precision*, ou mAP) considerando uma sobreposição mínima de 50% (IoU[1]) entre a predição do modelo e o objeto real. Essa métrica avalia se o modelo está identificando corretamente a localização dos objetos com uma margem de erro de até 50%.

A métrica IoU (*Intersection over Union*), ou seja, a intersecção sobre união, é utilizada para avaliar a precisão das previsões de modelos de detecção de objetos. Ela mede o grau de

[1] IoU: do inglês *Intersection over Union*

sobreposição entre a caixa delimitadora prevista pelo modelo e a caixa real (verdadeiro valor ou *ground truth*) do objeto. A métrica IoU varia entre 0 e 1, onde 1 indica a previsão perfeita entre a caixa prevista e a real e 0 indica que não existe sobreposição.

A *Metrics*/mAP50-95(B) expande a avaliação do mAP considerando uma média em diferentes limiares de sobreposição (IoU), variando de 0,5 a 0,95 em incrementos de 0,05, fornecendo uma análise mais detalhada do desempenho do modelo em identificar objetos com diferentes níveis de precisão exigidos.

Essas métricas juntas oferecem uma visão abrangente do desempenho do modelo, cobrindo desde a localização dos objetos (caixas delimitadoras) até a classificação correta e a sensibilidade em cenários variados.

3.2 Treinamento do Modelo

A primeira tentativa de treinamento do modelo foi realizada localmente, utilizando o Visual Studio Code em um notebook pessoal, um processador core i5 8265U, com 16 gigabytes de memória RAM, placa de vídeo integrada da Intel, 463 gigabytes de armazenamento SSD, 1 terabyte de armazenamento HD. com um sistema operacional Windows 10. No entanto, essa abordagem revelou-se inviável devido ao longo tempo de processamento necessário. Estimou-se cerca de 15 horas para treinar apenas 100 imagens ao longo de 100 épocas, evidenciando a limitação do hardware disponível.

Diante desse obstáculo, optou-se pela utilização do Google Colab, uma ferramenta que oferece suporte gratuito e limitado ao uso de GPUs, permitindo um treinamento mais rápido e eficiente. Contudo, devido à complexidade e à demanda do projeto, foi necessário migrar para o Google Colab Pro, que disponibiliza GPUs de maior desempenho, atendendo às necessidades específicas do treinamento.

A GPU utilizada, é a melhor que o plano Pro do google colab, sendo a A100 que fornece 40 gigabytes de memória a unidade de processamento gráfico permitindo um funcionamento mais rápido ao treinamento, assim como 83.5 gigabytes de memória RAM e 235.7 gigabytes de armazenamento para o notebook de testes.

Os notebooks do Colab permitem combinar código executável e texto rico em um único documento, além de imagens (Google Research, 2017).

Além do ambiente computacional, foram empregadas bibliotecas especializadas, destacando-se a Ultralytics, que forneceu o modelo pré-configurado do YOLO, facilitando sua instalação, execução e treinamento. Como afirmam os desenvolvedores da Ultralytics, "YOLO revolucionou a detecção de objetos ao combinar precisão, velocidade e simplicidade em uma única arquitetura, oferecendo resultados excepcionais em tempo real para uma ampla gama de aplicações" (Ultralytics, 2023). Para avaliar o impacto do tempo de treinamento no desempenho do modelo, foram criados três conjuntos de treinamento distintos, variando o número de épocas. E realizou-se o treinamento com as seguintes configurações: 12868 imagens com 300 épocas, 12.868 Imagens com 80 Épocas e 12.868 Imagens com 20 Épocas.

A época (ou epoch) é um conceito essencial no aprendizado de máquina, representando uma passagem completa por todo o conjunto de dados de treinamento durante o processo de treinamento de um modelo. Durante cada época, o modelo faz previsões com base nos pesos atuais, calcula os erros em relação aos rótulos reais e ajusta seus pesos utilizando algoritmos de

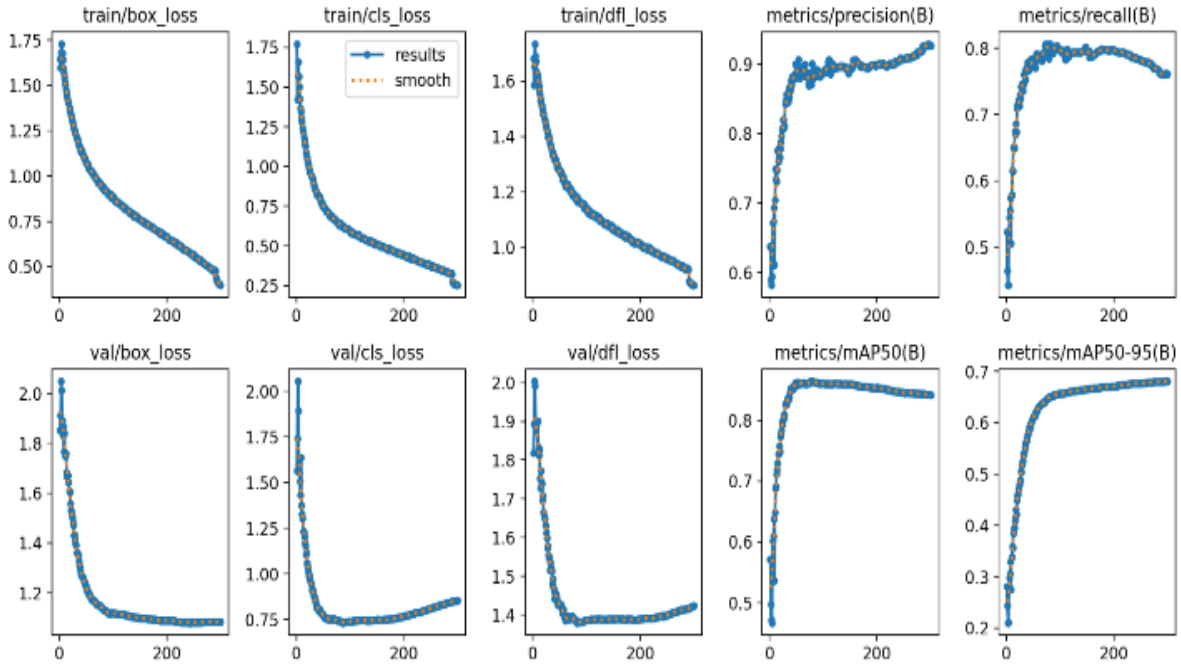
otimização, como o Gradiente Descendente, para refinar sua capacidade de generalização e melhorar seu desempenho.

O treinamento com 12.868 imagens ao longo de 300 épocas foi planejado para oferecer à rede neural um tempo significativo de aprendizado, permitindo a extração detalhada das características das imagens e aprimorando sua capacidade de identificar buracos com alta precisão. Essa abordagem visou maximizar o desempenho do modelo, garantindo sua robustez mesmo em cenários desafiadores, como condições adversas de iluminação e variações nas características das estradas. Ao explorar os limites do modelo, buscou-se desenvolver uma solução eficiente e confiável para detecção de buracos.

O modelo treinado ao longo de 300 épocas demonstrou um desempenho significativo na tarefa de detecção de buracos. Para avaliar sua eficiência, foram analisadas métricas de treinamento e validação, cada qual com propósitos distintos. As métricas de treinamento refletem o ajuste do modelo aos dados apresentados durante o aprendizado, enquanto as de validação avaliam sua capacidade de generalização ao lidar com dados inéditos, sendo cruciais para identificar possíveis problemas de sobreajuste.

A análise comparativa entre as métricas de treinamento e validação destaca a capacidade do modelo de generalizar bem os dados, o que é evidenciado pela proximidade entre as perdas de ambos os conjuntos. Essa proximidade demonstra que o modelo não sofreu sobreajuste, ou seja, não se limitou a memorizar os dados de treinamento, mas aprendeu padrões úteis para realizar previsões em dados não vistos. Com perdas reduzidas tanto no treinamento quanto na validação, e com métricas globais de alto desempenho, o modelo treinado por 300 épocas demonstrou ser robusto, eficiente e adequado para a detecção de buracos em diversos cenários.

Figura 3 – Gráficos do treinamento 1º de detecção

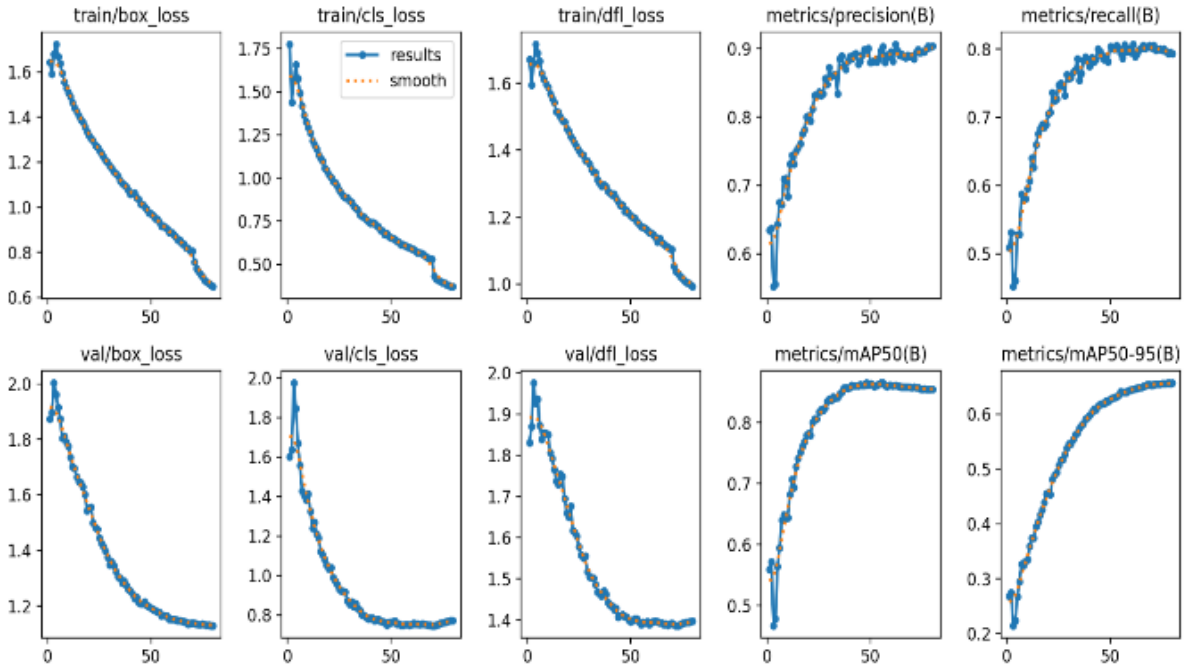


Fonte: dos autores

O treinamento com 12.868 imagens ao longo de 80 épocas foi configurado como uma abordagem intermediária, buscando avaliar o desempenho do modelo com um tempo de treinamento reduzido em comparação a configurações mais extensas. O objetivo principal foi encontrar um equilíbrio entre a precisão do modelo e a eficiência temporal do processo, permitindo uma análise do impacto da redução de épocas no aprendizado da rede neural sem comprometer significativamente sua capacidade de detectar buracos.

O modelo de 80 épocas apresenta uma redução significativa das perdas de treinamento e validação, com valores de mAP50 e mAP50-95 próximos de 0,8 e 0,6, respectivamente. Esses resultados indicam que o modelo ainda está em fase de evolução, mas já demonstra um desempenho consistente e confiável para a detecção de buracos. Ele é capaz de identificar características essenciais nas imagens, mostrando-se adequado para cenários menos complexos ou com menor variabilidade, embora ainda tenha espaço para melhorias adicionais com mais épocas de treinamento.

Figura 4 - Gráficos do 2º treinamento de detecção

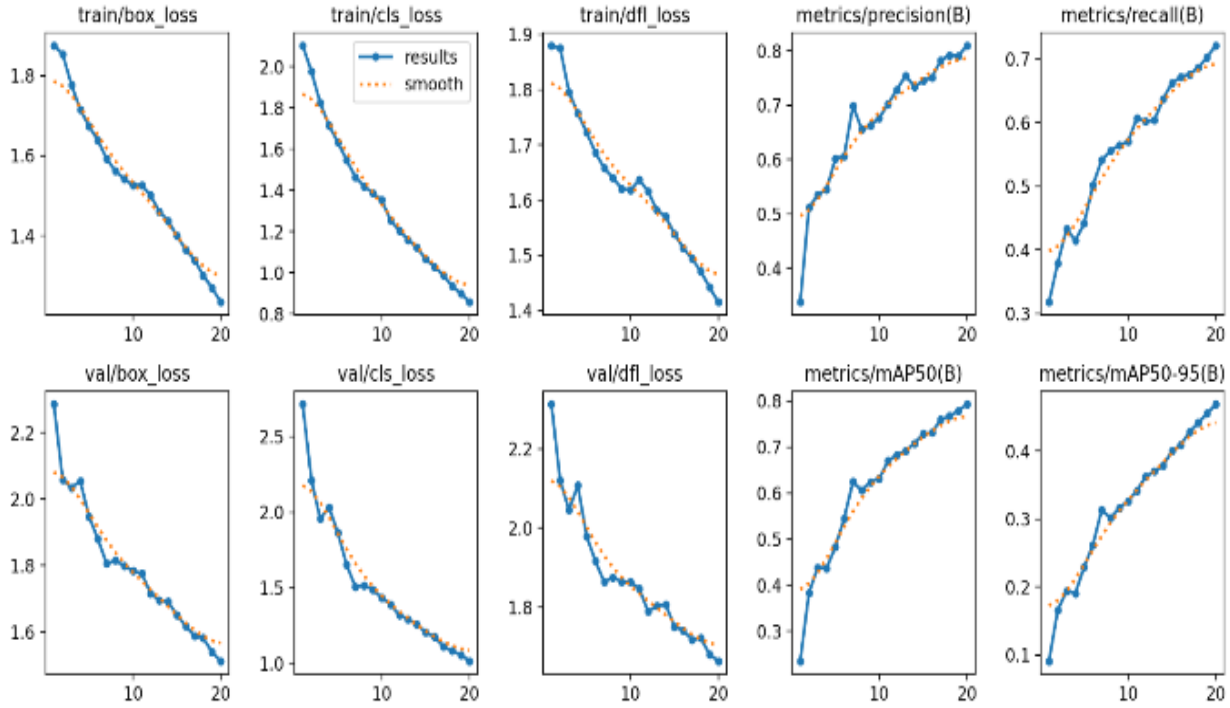


Fonte: dos autores

O treinamento com 12.868 imagens ao longo de 20 épocas foi projetado para simular um cenário de tempo de treinamento mais restrito, buscando avaliar a capacidade do modelo de alcançar um desempenho aceitável mesmo com um número reduzido de épocas. Essa configuração permitiu analisar o impacto de um treinamento mais curto no aprendizado da rede neural, verificando se ela seria capaz de identificar buracos de maneira razoável, mesmo em condições de treinamento limitadas.

O modelo treinado por 20 épocas demonstra um início promissor, com perdas em declínio e métricas como mAP50-95 atingindo valores intermediários. Embora ainda esteja em estágio inicial de aprendizado, o modelo já consegue realizar detecções básicas de buracos, indicando que a rede neural está aprendendo a extrair padrões relevantes das imagens. No entanto, seu desempenho limitado em comparação com os modelos de 80 e 300 épocas mostra que ele não é ideal para situações complexas ou variações significativas nos dados.

Figura 5 - Gráficos do 3º treinamento de detecção



Fonte: dos autores

3.7 Métricas de Desempenho

O desempenho de um classificador pode ser verificado por meio de métricas de desempenho como a acurácia, F-Score ou AUC (*area under curve*) (HOSSIN; SULAIMAN, 2015). Durante o processo de avaliação de um classificador, é importante conhecer a capacidade de generalização do teste a ser realizado, obtida com a validação cruzada dos dados analisados. A validação cruzada é aplicada para estimar o quão preciso é o modelo para um novo conjunto de dados, sendo sua abordagem principal a separação do conjunto de dados em subconjuntos mutuamente exclusivos, alguns dos quais são utilizados para treinamento e os demais para validação ou teste (SATO et al., 2013; LAUREANO; CAETANO; CORTEZ, 2014).

Após todas as passagens pelos grupos de teste e treinamento, pode-se calcular a taxa média de acerto do classificador utilizando alguma métrica de desempenho como a acurácia, técnica que pode ser aplicada a diferentes classificadores. As métricas de avaliação adotadas nas tarefas de aprendizado profundo desempenham um papel fundamental na obtenção dos resultados da otimização de um classificador. Assim, uma seleção de métricas de avaliação adequadas é uma chave importante para discriminar e obter o classificador ideal (HOSSIN; SULAIMAN, 2015). As métricas são aplicadas para confirmar uma hipótese. Na área de processamento de imagens ou reconhecimento de padrões, essas métricas são extraídas a partir de uma rotulagem dos dados investigados.

Para problemas de classificação binária, a avaliação de uma melhor (ótima) solução para uma determinada classificação pode ser definida com base na matriz de confusão. A partir desta matriz, rótulos são atribuídos: verdadeiro positivo (vp), falso positivo (fp), verdadeiro negativo (vn) e falso negativo (fn), onde: Verdadeiro positivo (vp): se uma instância é positiva e é classificada como positiva; Verdadeiro negativo (vn): se uma instância é negativa e é

classificada como negativa; Falso positivo (fp): se uma instância é negativa e é classificada como positiva; Falso negativo (fn): se uma instância é positiva e é classificada como negativa.

A partir da matriz de confusão, várias métricas comumente usadas podem ser geradas, conforme descrito na Tabela 1 (HOSSIN; SULAIMAN, 2015).

Tabela 1: Métricas (matriz de confusão)

Métrica	Fórmula	Foco de Avaliação
Acurácia (acc)	$\frac{vp + vn}{vp + fp + vn + fn}$	Em geral, a métrica de acurácia mede a proporção de previsões corretas sobre o total de instâncias avaliadas.
Taxa de Erro (err)	$\frac{fp + fn}{vp + fp + vn + fn}$	Mede a proporção de previsões incorretas sobre o número de instâncias avaliadas.
Sensibilidade (sn)	$\frac{vp}{vp + fn}$	Utilizada para medir a fração de padrões positivos que são classificados corretamente.
Especificidade (sp)	$\frac{vn}{vn + fp}$	Utilizada para medir a fração de padrões negativos que são classificados corretamente.
Precisão (p)	$\frac{vp}{vp + fp}$	Utilizada para medir a relação entre as previsões positivas realizadas corretamente e todas as previsões positivas (incluindo as falsas).
Recall (r)	$\frac{vp}{vp + vn}$	Utilizado para medir a fração de padrões positivos corretamente classificados.
F_1 *Score* (fm)	$2 * \frac{p * r}{p + r}$	Representa a média harmônica entre os valores de precisão e *recall*.

Fonte: Adaptado de Hossin e Sulaiman (2015)

Hossin e Sulaiman (2015) afirmam que a precisão é a métrica de avaliação mais utilizada para problemas de classificação binária ou de múltiplas classes. Uma métrica complementar à acurácia é a taxa de erro que avalia a solução produzida por sua porcentagem de previsões corretas, sendo que uma das grandes vantagens de ambas é a facilidade de serem calculadas, entendidas e aplicáveis a diversos problemas diferentes. Os autores ainda afirmam que a métrica F1 Score é um bom discriminador e obteve melhor desempenho do que a acurácia na otimização de problemas de classificação binária.

4. RESULTADOS E DISCUSSÃO

Após o treinamento do modelo, os arquivos com treinamento foram baixados do Google Colab, abrangendo os três cenários testados: 300 épocas, 80 épocas e 20 épocas. O desenvolvimento final foi conduzido no ambiente Visual Studio Code, utilizado para testar e depurar o código.

Durante essa etapa, foram empregadas diversas bibliotecas para viabilizar a implementação e funcionalidade do modelo. A biblioteca OS foi utilizada para interagir com o sistema operacional no qual o modelo YOLO seria configurado. A biblioteca Ultralytics foi essencial para a instalação e integração do modelo YOLO, enquanto a OpenCV foi empregada para carregar as imagens, visualizar os resultados e desenhar as caixas delimitadoras (bounding boxes).

Adicionalmente, foi configurado um critério de confiança para o modelo. As caixas delimitadoras foram desenhadas apenas em objetos que apresentassem uma probabilidade de detecção superior a 40%, garantindo maior precisão e confiabilidade nos resultados. Essa

abordagem foi fundamental para minimizar falsas detecções e aprimorar a eficácia do modelo na identificação de objetos relevantes.

Foi implementada a funcionalidade de realizar a detecção de buracos em múltiplas imagens de forma automatizada. O processo inicia com a abertura de uma pasta específica que contém as imagens a serem analisadas. Em seguida, o código percorre todos os arquivos dessa pasta, verificando se possuem formatos compatíveis, como .jpg, .jpeg ou .png.

Cada imagem identificada é carregada e processada pelo modelo de detecção. Durante o processamento, o modelo realiza a identificação dos objetos presentes na imagem (neste caso, buracos) e desenha caixas delimitadoras (*bounding boxes*) ao redor deles, desde que a confiança da detecção seja superior a um limite previamente definido.

Após o processamento, a imagem com as detecções é salva em uma pasta de destino previamente configurada, mantendo o nome do arquivo original. Esse fluxo de trabalho garante que todas as imagens processadas sejam armazenadas de forma organizada, facilitando a análise e visualização dos resultados de maneira centralizada e eficiente. Essa abordagem proporciona praticidade e escalabilidade, permitindo que várias imagens sejam analisadas em sequência sem necessidade de intervenção manual.

O modelo treinado foi carregado e submetido a testes utilizando um conjunto de imagens de validação. Durante esse processo, a biblioteca OpenCV foi utilizada para processar as imagens e desenhar as caixas delimitadoras (bounding boxes) nos locais que o modelo identificou como buracos. A avaliação das detecções foi feita com base na análise visual e na sobreposição das bounding boxes geradas pelo modelo em relação aos buracos reais presentes nas imagens.Para que o modelo fosse considerado como tendo realizado uma detecção correta, a bounding box deveria delimitar de maneira precisa a região ao redor do buraco, com uma margem de erro mínima. Essa precisão foi avaliada por meio de uma inspeção visual detalhada, onde as caixas geradas pelo modelo foram comparadas com os buracos visíveis nas imagens originais. O critério para o acerto levou em consideração não apenas a localização correta, mas também o tamanho e o alinhamento da bounding box em relação à borda do buraco.

Figura 6 - Detecção do buraco em uma imagem real

Fonte: dos autores

Nesta etapa, foi realizada a coleta e análise dos dados obtidos a partir dos três modelos treinados. Cada modelo foi avaliado com um conjunto de 100 imagens, sendo 50 delas contendo buracos em diversas formas e condições, e as outras 50 sem a presença de buracos. Essa divisão permitiu uma análise detalhada dos falsos negativos (FN), ou seja, casos em que o modelo falhou em detectar buracos presentes nas imagens.

Com base nos resultados obtidos, foi conduzida uma discussão comparativa entre os modelos, considerando o desempenho de cada um em relação à tarefa de detecção. Essa análise visou identificar diferenças de desempenho associadas às configurações de treinamento e determinar qual modelo apresentou os melhores resultados, alinhados aos objetivos propostos pelo trabalho. Essa abordagem permitiu avaliar de forma criteriosa a eficácia dos modelos e fornecer insights relevantes para futuras otimizações.

A análise dos dados da Tabela 2 apresenta o número de buracos detectados por cada modelo treinado com diferentes quantidades de épocas. Inicialmente, foi realizada uma contagem manual dos buracos presentes nas imagens, estabelecendo o total de 126 buracos como referência. Após o treinamento, os modelos foram avaliados individualmente, analisando imagem por imagem para verificar os buracos corretamente detectados e identificar os falsos positivos. O modelo com 20 épocas detectou 101 buracos, mas apresentou 8 falsos positivos. Já o modelo com 80 épocas detectou 95 buracos, com 5 falsos positivos, enquanto o modelo com 300 épocas localizou 94 buracos com apenas 1 falso positivo.

Tabela 2: Número de Buracos detectados para cada um dos modelos estudados

Número de Épocas	Quantidade de Buracos	Buracos Detectados	Falso Positivo
20 Épocas	126	101	8
80 Épocas	126	95	5
300 Épocas	126	94	1

Fonte: Elaborado pelos autores.

A análise dos modelos treinados (Tabela 3) evidenciou que o modelo com 20 épocas alcançou a maior acurácia (81,76%), destacando-se também por sua elevada sensibilidade (80,15%), o que reflete sua capacidade de identificar buracos com eficácia. Contudo, devido ao número limitado de épocas de treinamento, esse modelo apresentou a menor especificidade (85,45%), resultando em um elevado número de falsos positivos, indicando que frequentemente identificou objetos inexistentes como buracos.

Tabela 3: Métricas de Desempenho dos modelos estudados com 30, 80 e 300 épocas e treinamento

NºÉpocas	Precisão	Sensibilidade	Especificidade	F1 Score	Acurácia
20 Épocas	92,66%	80,15%	85,45%	85,95%	81,76%
80 Épocas	95,00%	75,39%	90,56%	84,06%	79,88%
300 Épocas	98,94%	74,60%	98,03%	85,06%	81,35%

Fonte: Elaborado pelos autores.

O modelo treinado com 80 época, como pode ser visualizado na Tabela 3, apresentou desempenho intermediário, com acurácia de 79,88%, sensibilidade de 75,39% e especificidade de 90,56%. Apesar de uma redução nos falsos positivos em comparação ao modelo de 20 épocas, esse modelo não demonstrou vantagens significativas em nenhuma das métricas analisadas, mostrando-se inferior tanto em desempenho quanto em equilíbrio entre precisão e sensibilidade.

Por sua vez, o modelo treinado com 300 épocas destacou-se pela robustez de sua base de aprendizado, alcançando os melhores resultados em precisão (98,94%) e especificidade (98,03%). Esses valores indicam uma superior capacidade de evitar falsos positivos e identificar corretamente os buracos. Apesar de apresentar uma acurácia ligeiramente inferior à do modelo de 20 épocas (81,35%) e uma sensibilidade de 74,60%, o modelo de 300 épocas apresentou um F1 Score de 85,06%, demonstrando um bom equilíbrio entre os acertos e erros.

Conclui-se que, embora o modelo de 20 épocas tenha apresentado maior acurácia e sensibilidade, o modelo treinado com 300 épocas revelou-se mais eficiente em cenários que demandam alta precisão e especificidade, mostrando-se a alternativa mais adequada para aplicações que priorizam a redução de falsos positivos e a consistência na identificação de buracos.

5. **CONCLUSÃO**

Para concluir, podemos afirmar que o algoritmo treinado por 300 épocas se destaca como o mais eficiente em cumprir o objetivo proposto, mesmo não apresentando a maior acurácia entre os três modelos analisados. Sua performance superior em termos de precisão e especificidade faz com que ele se sobressaia, sendo capaz de identificar os buracos de forma mais assertiva e consistente do que os demais algoritmos. Em contextos de aplicação prática, especialmente no suporte a pessoas com deficiências visuais, essa alta precisão e especificidade são fatores essenciais, pois reduzem a probabilidade de falsos alarmes e aumentam a confiabilidade do sistema em detectar buracos com exatidão.

Além disso, a capacidade do modelo de 300 épocas de minimizar falsos positivos indicados por sua especificidade elevada, significa que ele é mais eficaz em diferenciar verdadeiros buracos de outras irregularidades menos relevantes no solo. Isso é particularmente importante para pessoas com deficiência visual, pois uma detecção precisa evita distrações ou alertas desnecessários.

Dessa forma, podemos concluir que este modelo oferece um equilíbrio ideal entre sensibilidade e especificidade, garantindo uma eficácia aprimorada na detecção dos buracos, o que o torna a opção mais adequada para o uso prático no contexto proposto.

REFERÊNCIAS

CHENNA, dwith. (2023) *Evolution of Convolutional Neural Network (CNN): Compute vs Memory bandwidth for Edge AI*.

DATAFOLHA. Buraco no asfalto é problema mais comum de SP para 8 de cada 10 moradores, mostra Datafolha, Disponível em: https://www1.folha.uol.com.br/cotidiano/2024/03/buraco-no-asfalto-e-problema-mais-comum-de-sp-para-8-de-cada-10-moradores-diz-datafolha.shtml. Acesso em: 14 jun. 2024.

Dutta, G. (2023). *pothole dataset_yolov8_format*. Kaggle. Disponível em: https://www.kaggle.com/datasets/ryukijanoramunae/pothole-dataset

FAGUNDES, Wendell S. ; JÚNIOR, Mauro José A. (2022) Evolução das técnicas de classificação de imagens do sensoriamento remoto utilizadas na produção científica brasileira.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

Google Research, (2017), Disponível em: https://colab.research.google.com

HOSSIN, M; SULAIMAN, M. N. A Review on Evaluation Metrics for Data Classification Evaluations. **International Journal of Data Mining & Knowledge Management Process**, v. 5, n. 2, p. 1-11, March 2015.

HUANG, R.; PEDOEEM, J.; CHEN, C. *Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers*. In: IEEE. 2018 IEEE *International Conference on Big Data* (Big Data). [S.l.], 2018. p. 2503–2510.

IA Expert academy. Disponível em: https://iaexpert.academy/2020/10/13/deteccao-de-objetos-com-yolo-uma-abordagem-moderna/. Acesso em: 08 out. 2024.

IBGE. Censo Demográfico 2010: Características Gerais da População, Religião e Pessoas com Deficiência. Rio de Janeiro, 2012. Acompanha CD-ROM

Jones Granatyr. Disponível em: www.udemy.com/course/deteccao-de-objetos-com-yolo-darknet-opencv-python. Acesso em: 24 out. 2024

JUNIOR, Joel Savenazzo Redes Neurais Convolucionais Aplicadas à Detecção de Não-Conformidades em Equipamentos Industriais, 2022.

LAUREANO, R. M. S.; CAETANO, N.; CORTEZ, P. Previsão de tempos de internamento num hospital português: aplicação da metodologia CRISP-DM. **RISTI – Revista Ibérica de Sistemas e Tecnologias da Informação**. Associação Ibérica de Sistemas e Tecnologia da Informação (AISTI), n. 13, p. 581-591, jun. 2014.

LeCun, Y., Bengio, Y., & Hinton, G. (2015*). Deep learning. Nature*, 521(7553), 436-444.
Ng, Andrew. "*Machine Learning Yearning: Technical Strategy for AI Engineers, In the Era of Deep Learning.*" deeplearning.ai, 2018. Acesso em: 16 jun. 2024.

LEITE, Maria Ruth Siffert Diniz Teixeira; SILVA, Glicélio Ramos. Inclusão da pessoa com deficiência visual nas instituições de educação superior de belo horizonte. 2006. Disponível em: . Acesso em: 10 out. 2019.

LI, Zewen. et al. *A survey of convolutional neural networks: analysis, applications, and prospects*. IEEE *transactions on neural networks and learning systems*, IEEE, 2021.

LUDERMIR, Bernarda Teresa. (2021) Inteligência Artificial e Aprendizado de Máquina: estado atual e tendências.

Mario Filho | Machine Learning. Disponível em: https://mariofilho.com/precisao-recall-e-f1-score-em-machine-learning/. Acesso em: 25 out. 2024.

MITCHELL, T. Machine Learning. S. l.: McGraw Hill, 1997.

Pablo de Abreu Vieira. Disponível em: https://www.researchgate.net/figure/Figura-10-Figura-representando-o-processo-de-convolucao-2D-Temos-uma-matriz-como_fig3_375408014 Acesso em: 24 out 2024.

OPENCV. Disponível em: https://opencv.org/about/. Acesso em: 12 out. 2024.

PAUL, Mueller John; MASSARON, Luca. Aprendizado de Máquina: Para Leigos. 1ª. ed. atual. Rio de Janeiro: Alta Books, 2019. 432 p. v. 2. ISBN 9788575223451.

PONTI, M.; RIBEIRO, L.; NAZARE, T.; BUI T.; COLLOMOSSE, J.; "*Everything You Wanted to Know about Deep Learning for Computer Vision but Were Afraid to Ask*," 2017 30th SIBGRAPI *Conference on Graphics, Patterns and Images Tutorials* (SIBGRAPI-T), 2017, pp. 17-41.

PYTHON DOCS. Disponível em: https://docs.python.org/pt-br/3/faq/general.html. Acesso em: 16 out. 2024.

REDMON, J. et al. *You only look once: Unified, real-time object detection. In: Proceedings of the* IEEE *conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 779–788.

REDMON, J., & FARHADI, A. (2015). YOLO9000: Better, Faster, Stronger. https://arxiv.org/abs/1612.08242

RUSSEL, S., & NORVIG, P. (2016). Inteligência Artificial: Estruturas e Estratégias para a Solução Complexa de Problemas (3ª ed.). Pearson Education.

SATO, L. Y. et al. Análise comparativa de algoritmos de árvore de decisão do sistema WEKA para classificação do uso e cobertura da terra. In: XVI Simpósio Brasileiro de Sensoriamento Remoto - SBSR, Foz do Iguaçu, PR, 13 a 18 de abril de 2013, INPE. **Anais...** Foz do Iguaçu, Brasil, 2013. p. 2353-2360.

Subedi, S., R. B. Bist, X. Yang, and L. Chai. 2023. Tracking floor eggs with machine vision in cage-free hen houses. *Poultry Science*, 102:102637.

Tracker. (2024). O que é : Training Epoch. Disponível em: .https://iatracker.com.br/glossario/o-que-e-training-epoch/#:~:text=O%20Training%20Epoch%2C%20ou%20%C3%A9poca,durante%20o%20processo%20de%20treinamento. Acesso em: 28 nov. 2024.

Ultralytics. (2023). *YOLO: Real-time object detection with state-of-the-art performance* [Repositório]. GitHub. Disponível em: https://github.com/ultralytics/yolov5. Acesso em: 28 nov.. 2024.

Ultralytics. (2023). *YOLOv8: State-of-the-art object detection and segmentation*. Disponível em: https://github.com/ultralytics/ultralytics. Acesso em: 28 nov.. 2024.

Yolov8. (2024). What is box loss in yolov8?. Disponível em: https://yolov8.org/what-is-box-loss-in-yolov8/ . Acesso em: 28 nov.. 2024.