Universidade Estadual de Londrina

ÁTILA MARCONCINE DE SOUZA

UTILIZAÇÃO DE MODELOS DE *DEEP LEARNING* PARA O RECONHECIMENTO AUTOMÁTICO DE DEFEITOS E SEUS NÍVEIS DE SEVERIDADE EM PAVIMENTOS ASFÁLTICOS URBANOS

Londrina
2026

ÁTILA MARCONCINE DE SOUZA

UTILIZAÇÃO DE MODELOS DE *DEEP LEARNING* PARA O RECONHECIMENTO AUTOMÁTICO DE DEFEITOS E SEUS NÍVEIS DE SEVERIDADE EM PAVIMENTOS ASFÁLTICOS URBANOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Civil da Universidade Estadual de Londrina, como requisito à obtenção do título de Mestre em Engenharia Civil.

Orientadora: Profª. Dra. Heliana Barbosa Fontenele

Londrina
2026

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UEL

S713 Souza, Átila Marconcine de .
Utilização de modelos de *deep learning* para o reconhecimento automático de defeitos e seus níveis de severidade em pavimentos asfálticos urbanos / Átila Marconcine de Souza. - Londrina, 2026.
128 f. : il.

Orientador: Heliana Barbosa Fontenele.
Dissertação (Mestrado em Engenharia Civil) - Universidade Estadual de Londrina, Centro de Tecnologia e Urbanismo, Programa de Pós-Graduação em Engenharia Civil, 2026.
Inclui bibliografia.

1. Aprendizado de máquina - Tese. 2. Detecção de objetos - Tese. 3. Inteligência artificial - Tese. 4. Visão computacional - Tese. I. Fontenele, Heliana Barbosa. II. Universidade Estadual de Londrina. Centro de Tecnologia e Urbanismo. Programa de Pós-Graduação em Engenharia Civil. III. Título.

CDU 62

ÁTILA MARCONCINE DE SOUZA

UTILIZAÇÃO DE MODELOS DE *DEEP LEARNING* PARA O RECONHECIMENTO AUTOMÁTICO DE DEFEITOS E SEUS NÍVEIS DE SEVERIDADE EM PAVIMENTOS ASFÁLTICOS URBANOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Civil da Universidade Estadual de Londrina, como requisito à obtenção do título de Mestre em Engenharia Civil.

BANCA EXAMINADORA

Prof.ª. Dra. Heliana Barbosa Fontenele
Universidade Estadual de Londrina - UEL

Prof.ª. Dra. Lilian Tais de Gouveia
Universidade Estadual de Ponta Grossa - UEPG

Prof. Dr. Manoel Denis Costa Ferreira
Universidade Estadual de Londrina - UEL

Londrina, 25 de fevereiro de 2026.

AGRADECIMENTOS

Primeiramente, à minha mãe, Maria Aparecida Conceição Marconcini Prestes, e ao meu pai, Samuel Pereira de Souza, pelo apoio incondicional, pelo suporte prestado e pelo incentivo constante. Vocês são o meu exemplo.

À minha namorada, Jornê Cabral Macedo, pelo amor, companhia, paciência, apoio e compreensão. Agradeço imensamente por ter mudado de cidade para me acompanhar, sua presença em Londrina foi fundamental nesta caminhada.

À minha orientadora, Prof.ª. Dra. Heliana Barbosa Fontenele, pelas orientações prestadas e pelas sugestões valiosas que contribuíram para minha formação como pesquisador e futuro docente.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela disponibilização da bolsa de mestrado.

À Universidade Estadual de Londrina (UEL), em especial ao Centro de Tecnologia e Urbanismo (CTU), ao Programa de Pós-Graduação em Engenharia Civil (PPGEciv) e ao Laboratório de Engenharia de Transportes (LET-CTU) por toda a infraestrutura disponibilizada para realização desse trabalho.

À plataforma Roboflow, pela concessão da licença estudantil, recurso fundamental que viabilizou o treinamento de diversos dos modelos de detecção utilizados nesta pesquisa.

Aos membros da banca Prof.ª. Dra. Lilian Tais de Gouveia e Prof. Dr. Manoel Denis Costa Ferreira, pelo tempo dedicado na leitura deste trabalho e pelas sugestões fornecidas.

Aos professores da pós-graduação em engenharia civil pelos ensinamentos dados no decorrer do mestrado, em especial ao Prof. Dr. Carlos Alberto Prado da Silva Junior pelas sugestões e conselhos.

Aos meus amigos do Maranhão, que mesmo à distância, me fizeram companhia e proporcionaram momentos de descontração.

A todos os que colaboraram direta ou indiretamente com o desenvolvimento deste trabalho.

O planeta sonho será Terra
A dissonância será bela.

Flávio Venturini, Vermelho e
Márcio Borges – 14 Bis

RESUMO

SOUZA, Átila Marconcine. **Utilização de Modelos de *Deep Learning* para o Reconhecimento Automático de Defeitos e seus Níveis de Severidade em Pavimentos Asfálticos Urbanos**. 2026. 128 f. Dissertação de Mestrado (Pós-Graduação em Engenharia Civil) – Universidade Estadual de Londrina, Londrina, 2026.

As vias urbanas desempenham um papel fundamental na sociedade. Para garantir o conforto e a segurança dos usuários dessas vias, é necessário a realização de procedimentos de avaliação, que em grande parte são lentos e cansativos. Visando automatizar esses procedimentos, o presente trabalho tem como objetivo investigar se modelos de *deep learning* são capazes de efetuar o reconhecimento de defeitos e seus níveis de severidade em pavimentos asfálticos urbanos. Para isso, foram coletadas, em campo, imagens dos defeitos trincas por fadiga, buracos e remendos. Essas imagens foram utilizadas em dois bancos de dados, um considerando apenas os defeitos e outro considerando os defeitos e os níveis de severidade. Os dois bancos de dados foram treinados em dois ambientes (local e *online*), através de diferentes versões das arquiteturas de detecção de objetos *You Only Look Once* e *Detection Transformer*. Para testar os melhores modelos treinados foram empregadas cem imagens, coletadas em campo, que não foram utilizadas durante o treinamento. Já para validação prática, o melhor modelo foi utilizado para detectar defeitos em um ortomosaico. Além disso, um aplicativo para desktop foi desenvolvido, integrando os modelos treinados em uma interface gráfica simples e intuitiva. Na etapa de treinamento, o modelo RF-DETR apresentou as melhores métricas de avaliação. Já na etapa de teste, o YOLOv12s demonstrou desempenho adequado para aplicação prática. Na etapa de validação prática, a utilização conjunta do YOLOv12s com o framework *Slicing Aided Hyper Inference* apresentou potencial. De forma geral, os modelos empregados apresentaram facilidade no reconhecimento da severidade baixa das trincas por fadiga e dos remendos. Para o defeito buraco, constatou-se que apenas a detecção de objetos não fornece resultados satisfatórios para distinção entre os níveis de severidade.

Palavras-chave: Aprendizado de Máquina; Detecção de Objetos; Inteligência Artificial; Redes Neurais; Visão Computacional.

ABSTRACT

SOUZA, Átila Marconcine. **Application of Deep Learning Models for Automatic Recognition of Distresses and Their Severity Levels in Urban Asphalt Pavements**. 2026. 128 p. Master's Thesis (Master's in Civil Engineering) – State University of Londrina, Londrina, 2026.

Urban roads play a fundamental role in society. To ensure the comfort and safety of road users, it is necessary to perform evaluation procedures, which are largely slow and tedious. Aiming to automate these procedures, the present study investigates whether deep learning models are capable of recognizing distresses and their severity levels in urban asphalt pavements. For this purpose, field images were collected for the distresses fatigue cracks, potholes, and patches. These images were used in two datasets: one considering only the distresses and another considering both the distresses and their severity levels. Both datasets were trained in two environments (local and online) using different versions of the You Only Look Once and Detection Transformer object detection architectures. To test the best trained models, one hundred field-collected images that were not used during training were employed. For practical validation, the best model was used to detect distresses in an orthomosaic. Additionally, a desktop application was developed, integrating the trained models into a simple and intuitive graphical interface. In the training stage, the RF-DETR model presented the best evaluation metrics. In the testing stage, YOLOv12s demonstrated adequate performance for practical application. In the practical validation stage, the combined use of YOLOv12s with the Slicing Aided Hyper Inference framework showed potential. Overall, the models employed showed ease in recognizing low-severity fatigue cracks and patches. For the pothole distress, it was found that object detection alone does not provide satisfactory results for distinguishing between severity levels.

Key-words: Artificial Intelligence; Computer Vision; Neural Network; Machine Learning; Object Detection

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

A2	*Area Attention*
AP	Average Precision
ASTM	*American Society for Testing and Materials*
BD1	Banco de Dados 1
BD2	Banco de Dados 2
C2PSA	*Cross Stage Partial with Spatial Attention*
CNN	*Convolutional Neural Networks*
COCO	*Common Objects in Context*
DETR	*DEtection TRansformer*
DFN	*Discriminative Feature Network*
DIM	*Distress Identification Manual*
DL	*Deep Learning*
DNIT	Departamento Nacional de Infraestrutura de Transportes
DNN	*Deep Neural Networks*
DSJ	Defeitos e Severidades Juntos
DSS	Defeitos e Severidades Separados
DT	*Decision Trees*
ES	Especificação de Serviço
FFU	*Feature Fusion Upsampling*
FFN	*Feed Forward Network*
FHWA	*Federal Highway Administration*
GPU	*Graphics Processing Unit*
IA	Inteligência Artificial
ICAP	Índice de Condição Atual do Pavimento
ICPU	Índice de Condição de Pavimentos Urbanos

ICPU$_{JP}$	Índice de Condição de Pavimento Urbano de João Pessoa
IGG	Índice de Gravidade Global
INO	*National Optics Institute*
KNN	*K-Nearest Neighbors*
LCMS	*Laser Crack Measurement System*
LTPP	*Long-Term Pavement Performance*
mAP	mean Average Precision
MDA	*Multiscale Dilated Attention*
ME	Melhor Época
M&R	Manutenção e/ou Reabilitação
MID	Manual de Identificação de Defeitos
ML	*Machine Learning*
MVF	Matriz de Valores Fixos
NAS	*Neural Architecture Search*
NE	Número de Épocas
PCI	*Pavement Condition Index*
PSPNet	*Pyramid Scene Parsing Network*
PSR	*Present Serviceability Ratio*
PRO	Procedimento
R-ELAN	*Residual Efficient Layer Aggregation Networks*
ReLU	*Rectified Linear Unit*
RNA	Rede Neural Artificial
RNN	*Recurrent Neural Networks*
SA	Severidade Alta
SAHI	*Slicing Aided Hyper Inference*
SB	Severidade Baixa
SfM	*Structure from Motion*

SGP	Sistema de Gerência de Pavimentos
SGPU	Sistema de Gerência de Pavimentos Urbanos
SL	*Shallow Learning*
SM	Severidade Média
SNN	*Shallow Neural Networks*
SPPF	*Spatial Pyramid Pooling Fast*
SVM	*Support Vector Machines*
TER	Terminologia
TPV	Tampa de Poço de Visita
TT	Tempo de Treinamento
UA	Unidade Amostral
VPAIPA	Veículo para Procedimento de Aquisição de Imagens do Pavimento e Avaliação
VSA	Valor de Serventia Atual
YOLO	*You Only Look Once*

SUMÁRIO

1 INTRODUÇÃO

O transporte rodoviário possui significativa importância socioeconômica, sendo utilizado diariamente para o transporte de cargas e passageiros. Essa importância se estende para as áreas urbanas, onde as vias, geralmente de revestimento asfáltico, desempenham um papel fundamental para o desenvolvimento das cidades. Dessa forma, a conservação adequada dos pavimentos asfálticos urbanos é essencial para garantir conforto e segurança aos usuários da via. Contudo, a deterioração da superfície devido à ocorrência de defeitos compromete diretamente esses fatores.

Os defeitos interferem na estrutura do pavimento, comprometendo a sua vida útil e dificultando a trafegabilidade de veículos. Defeitos severos, como os buracos, além de ocasionar danos aos veículos, representam um risco de acidentes, que em casos mais graves, podem ser fatais. Para evitar que esse tipo de problema ocorra, é fundamental que atividades de Manutenção e Reabilitação (M&R) sejam realizadas no momento adequado. Para tal, a superfície do pavimento deve ser acompanhada constantemente, através das avaliações de pavimentos.

As avaliações de pavimentos são uma parte essencial do Sistema de Gerência de Pavimento (SGP) e do Sistema de Gerência de Pavimentos Urbanos (SGPU), pois elas são o ponto de partida para futuras decisões a serem tomadas pelo sistema. Porém, grande parte dessas avaliações são realizadas por caminhamento, um procedimento extremamente lento e cansativo, podendo demandar horas para avaliação de um único trecho e exigindo o percurso de longas distancias a pé. Adicionalmente, riscos à segurança e a saúde dos avaliadores podem ocorrer, devido a exposição ao tráfego de veículos e condições climáticas adversas.

Com o intuito de contornar essas desvantagens, diversos estudos passaram a explorar algoritmos de inteligência artificial, especificamente de *machine learning* e *deep learning*, para o reconhecimento automático de defeitos. Essa abordagem reduz a necessidade de mão de obra humana e proporciona uma maior velocidade e eficácia no processo de avaliação de pavimentos.

No entanto, a maioria dos estudos existentes nesta área negligenciam um aspecto fundamental dos defeitos: o seu nível de severidade. O presente trabalho visa preencher essa lacuna, propondo efetuar o reconhecimento automático de todos os três níveis de severidade dos três defeitos mais comuns de ocorrerem em pavimentos asfálticos urbanos: as trincas por fadiga, os buracos e os remendos. Para isso, foram

empregados cinco detectores de objetos estado da arte: YOLOv12 (Tian; Ye; Doermann, 2025), RF-DETR (Robicheaux *et al.*, 2025), YOLOv11 (Ultralytics, 2024), YOLOv8 (Ultralytics, 2023) e YOLO-NAS (DeciAi, 2023).

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Investigar se modelos de *deep learning* são capazes de efetuar o reconhecimento automático de defeitos e seus níveis de severidade em pavimentos asfálticos urbanos.

1.1.2 Objetivos Específicos

Com o intuito de alcançar o objetivo geral, os seguintes objetivos específicos são necessários:

- Verificar na literatura como o *deep learning* é utilizado para o reconhecimento de defeitos e seus níveis de severidade;
- Treinar versões recentes dos algoritmos de detecção de objetos *You Only Look Once* (YOLO) e *DEtection TRansformer* (DETR), e definir qual obtém o melhor desempenho;
- Verificar quais defeitos e quais níveis de severidade os modelos treinados têm mais facilidade de reconhecer;
- Comparar o comportamento de dois bancos de dados: um com anotações relativas apenas aos defeitos e outro com anotações de defeitos e níveis de severidade;
- Elencar vantagens e desvantagens de treinar modelos em um ambiente local e em um ambiente online;
- Testar os melhores modelos em imagens não incluídas na etapa de treinamento com o intuito de constatar a sua aplicação prática;
- Realizar a validação prática do melhor modelo;

- Propor um *framework* visando facilitar a reprodutibilidade do método de pesquisa desenvolvido;
- Desenvolver um aplicativo que possibilite a integração dos modelos treinados e uma interface gráfica simples e intuitiva.

1.2 JUSTIFICATIVA

O presente trabalho se justifica pois o reconhecimento de defeitos e seus níveis de severidade são etapas cruciais das avaliações funcionais, que atualmente são realizadas por métodos predominantemente manuais. A automatização dessas avaliações, por meio de técnicas de IA, além de proporcionar diversas vantagens em relação os procedimentos manuais, permite uma implementação mais eficaz do SGPU.

Embora as técnicas de IA, principalmente o *machine learning* e o *deep learning*, venham sendo exploradas com frequência pela literatura para o reconhecimento de defeitos em pavimentos, diversas lacunas são perceptíveis. Dentre elas é possível destacar: a ausência de todos os três níveis de severidade, onde geralmente a severidade média é ignorada; abordagens que não contemplam múltiplos defeitos simultaneamente ou focam em classes especificas (e.g. detecções apenas de trincas); e a falta da avaliação em campo, tanto para confecção de banco de dados quanto para validação dos modelos treinados, comprometendo a aplicabilidade prática dos algoritmos.

Dessa forma, o presente trabalho leva em consideração os três níveis de severidade para os três defeitos mais comuns em áreas urbanas: trincas por fadiga, buracos e remendos. Para tal são empregadas arquiteturas modernas de *deep learning*, que possuem um alto desempenho no reconhecimento de objetos. Além disso, o banco de dados empregado para treinamento dos algoritmos de *deep learning* foi coletado em campo, seguindo as diretrizes da *American Society for Testing and Materials* - ASTM D6433-24. Também foi conduzida uma etapa de teste em campo, em imagens não utilizadas na etapa de treinamento.

Ao abordar todos os níveis de severidade desses três defeitos e empregar uma metodologia de coleta de dados e validação de modelos feita em campo, este estudo não apenas preenche lacunas científicas, mas também contribui para a otimização do SGPU, fornecendo economia e segurança.

1.3 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está dividido em 7 capítulos:

Capítulo 1 – Parte introdutória do trabalho, onde são apresentados o contexto, objetivos e justificativas deste estudo.

Capítulo 2 – Apresentação dos conceitos acerca do SGP e do SGPU. Também são abordados conceitos relativos à avaliação de pavimentos, defeitos superficiais e determinação do nível de severidade.

Capítulo 3 – Especifica os principais conceitos relativos a Inteligência Artificial, com enfoque no *machine learning*, *deep learning* e visão computacional. As arquiteturas escolhidas e as métricas de avaliação também são explicadas.

Capítulo 4 – Aborda os artigos da literatura que aplicam o *deep learning* para o reconhecimento automático de defeitos e níveis de severidade.

Capítulo 5 – Descrição dos métodos empregados para realização do trabalho, incluindo a confecção do banco de dados, treinamento dos modelos, avaliação dos modelos e desenvolvimento do aplicativo.

Capítulo 6 – Apresentação dos resultados e discussões, onde são feitas diversas análises (gerais es específicas) em relações aos modelos treinados e seu desempenho para o reconhecimento de defeitos e seus níveis de severidade.

Capítulo 7 – Aborda as conclusões obtidas e implementações a serem realizadas futuramente.

2 SISTEMA DE GERÊNCIA DE PAVIMENTOS

Um Sistema de Gerência de Pavimentos (SGP), de acordo com Haas, Hudson e Zaniewski (1994), consiste em um conjunto de ferramentas ou métodos que auxiliam os tomadores de decisão a encontrarem as melhores estratégias para fornecer e manter pavimentos em boas condições de serventia por um determinado período de tempo. O objetivo principal do SGP, conforme o Departamento Nacional de Infraestrutura de Transportes - DNIT (2011), é alcançar a melhor aplicação possível para os recursos públicos disponíveis e oferecer um transporte rodoviário seguro, compatível e econômico.

Tradicionalmente, o processo decisório de um SGP pode ser classificado em dois níveis: de rede e de projeto. O nível de rede caracteriza-se por estudar uma grande área ou malha viária, na qual se situam muitas rodovias. Nesse nível busca-se o conhecimento da malha como um todo, onde, através de avaliações, são indicados os trechos prioritários que devem ser objeto de investimentos em atividades de Manutenção e/ou Reabilitação (M&R). Já o nível de projeto envolve atividades detalhadas do próprio projeto e da execução de obras em um determinado trecho do pavimento. As atividades relativas ao nível de projeto devem subsidiar orçamentos e programas de curto prazo (DNIT, 2011).

Os detalhes de um SGP podem variar dependendo da agência ou organização que ele for implementando. De forma geral, o SGP possui diversos componentes que interagem entre si, sendo eles: planejamento, programação, projeto, construção, manutenção e reabilitação. Além desses componentes, fatores externos também afetam o SGP, como: orçamentos, informações e políticas administrativas não quantificáveis (Haas; Hudson; Zaniewski, 1994). Porém, de acordo com Páez (2015), o componente principal de um SGP é a sua base de dados, que deve ser coletada e atualizada de forma periódica para definição das melhores estratégias de M&R com o melhor custo-benefício possível.

Os conceitos apresentados acerca do SGP, apesar de serem desenvolvidos com foco nos pavimentos rodoviários, também são válidos para pavimentos urbanos, e com alguns ajustes é possível obter um Sistema de Gerência de Pavimentos Urbanos (SGPU) (Zanchetta, 2017). O SGPU, conforme Páez (2015), é a combinação de procedimentos de análises, formas detalhadas, medições, critério de decisão e ferramentas (*e.g.* programas computacionais), os quais fornecem aos administradores

métodos sistemáticos ótimos para a gerência.

O SGPU precisa levar em considerações características particulares dos pavimentos urbanos, que não ocorrem em pavimentos rodoviários. Segundo Danieleski (2004) essas características são:

- Grande quantidade de interseções, em consequência, as velocidades dos veículos são menores e a frenagens são mais frequentes;
- Presença de redes subterrâneas (água e esgoto, energia elétrica, gás e telefonia). Dessa forma, qualquer obra de manutenção e/ou ampliação dessas redes, exige intervenção nos pavimentos, além da presença de tampas de poços de visita no revestimento dos pavimentos;
- Segregação do tráfego, com faixas exclusivas para ônibus;
- Pavimentos revestidos com calçamentos;
- Presença de vegetação nas bordas dos pavimentos, na qual as raízes podem interferir na estrutura do pavimento;
- Demasiado trânsito de pedestres, exigindo assim uma maior sinalização horizontal;
- Adaptação do projeto geométrico em cidades com topografia acidentada, gerando inclinações elevadas.

Apesar do SGPU necessitar de algumas modificações em relação ao SGP para ser implementando, devido as características mencionadas, em ambos os sistemas a avaliação de pavimentos é uma das etapas mais importantes.

As avaliações de pavimentos, de acordo com DNIT (2011), são o ponto de partida para as futuras decisões a serem tomadas no sistema. Ressalta-se que o presente trabalho tem como foco as avaliações funcionais, uma vez que os defeitos de superfície são o objeto de estudo. Dessa forma, o tópico 2.1 aborda as avaliações de pavimentos (com enfoque nas avaliações funcionais) enquanto o tópico 2.2 aborda os defeitos de superfície e a determinação do seu nível de severidade.

2.1 AVALIAÇÃO DE PAVIMENTOS

As avaliações de pavimentos podem ser divididas em funcionais, estruturais e de segurança. As funcionais têm como foco os defeitos superficiais e seus impactos no conforto ao rolamento. A avaliação estrutural, por sua vez, consiste em determinar a capacidade de carga do pavimento e a necessidade de reforço da estrutura

existente, através de ensaios destrutivos e/ou não-destrutivos. Já a avaliação da segurança trata da interação pneu-pavimento (*i.e.* atrito, aquaplanagem e resistência à derrapagem).

As avaliações funcionais podem ser subjetivas ou objetivas. Enquanto as avaliações subjetivas utilizam conceitos qualitativos, as avaliações objetivas utilizam a quantificação numérica e a determinação do nível de severidade dos defeitos (DNIT, 2006). Essas avaliações seguem procedimentos estabelecidos em normas e manuais, na qual são determinados índices que avaliam o estado da superfície do pavimento e como este estado influencia no conforto ao rolamento. Dentre os índices subjetivos é possível mencionar o *Present Serviceability Ratio* (PSR), elaborado por Carey Jr e Irick (1960) e sua versão brasileira, normatizada pelo DNIT através da norma de Procedimento (PRO) DNIT 009/2003 – PRO, o Valor de Serventia Atual (VSA). No que se refere aos índices objetivos, o *Pavement Condition Index* (PCI), normatizado pela *American Society for Testing and Materials* (ASTM), por meio da ASTM D6433−24, é o mais utilizado mundialmente. No Brasil, dentre os índices objetivos destaca-se Índice de Gravidade Global (IGG), normatizado pela DNIT 006/2003 – PRO.

Ressalta-se que existem diversos índices brasileiros que foram desenvolvidos especificamente para áreas urbanas, das quais é possível mencionar o Índice de Condição de Pavimentos Urbanos (ICPU) (Páez, 2015), o Índice de Condição de Pavimento Urbano de João Pessoa ($ICPU_{JP}$) (Albuquerque, 2017), a Matriz de Valores Fixos (MVF) (Zanchetta, 2017) e o Índice de Condição Atual do Pavimento (ICAP) (Salviatto, 2021).

Para a determinação dos índices supracitados (sejam específicos da área urbana ou não) é necessário realizar a coleta de dados relativos aos defeitos de superfície (*e.g.* tipo de defeito, quantidade, propriedades geométricas). Valipour *et al.* (2023) afirma que esses dados podem ser obtidos de três formas: manual, semiautomática e automática. O método manual é geralmente realizado por caminhamento, na qual avaliadores andam nas vias coletando informações dos defeitos presentes, utilizando planilhas e equipamentos de medição. No método semiautomático, a coleta de dados é feita automaticamente, através de câmeras, sensores e lasers, porém um avaliador é responsável pelo processamento dos dados obtidos. No método automático, por sua vez, tanto a coleta, quanto o processamento de dados é realizada de forma automática, sem a necessidade de um avaliador.

As avaliações automáticas e semiautomáticas oferecem mais vantagens em relação as avaliações manuais, uma vez que essas consistem em procedimentos demorados, tediosos e cansativos, além de fornecerem risco aos avaliadores, que estão expostos de forma excessiva ao sol e suscetíveis à acidentes de trânsito.

Visando a automatização das avaliações de pavimentos, diversos artigos na literatura vêm empregando a Inteligência Artificial (IA), em particular modelos de *Machine Learning* (ML), *Deep Learning* (DL) e visão computacional, para o reconhecimento de defeitos em pavimentos. Sholevar, Golroo e Esfahani (2022) destacam que o bom desempenho apresentado por esses modelos tem uma importância significativa para as avaliações de redes viárias, uma vez que estas envolvem a análise e monitoramento de conjuntos de dados massivos (centenas de milhares de quilômetros de rodovias). Além disso, os autores abordam que os modelos de ML oferecem uma grande capacidade de generalização, apresentando um melhor desempenho em dados inéditos quando comparados aos métodos tradicionais.

Informações teóricas acerca da IA e suas subáreas (*i.e.* ML, DL e visão computacional) e artigos recentes da literatura que utilizam os modelos mencionados na identificação automática de defeitos serão abordados com mais detalhes nos Capítulos 3 e 4, respectivamente.

2.2 Defeitos de Superfície e Níveis de Severidade

Os defeitos de superfície são os danos ou as deteriorações que ocorrem na superfície dos pavimentos asfálticos, que podem ser identificados a olho nu e classificados através de uma norma técnica. Existem diversos defeitos, e cada norma apresenta tipos, conceitos e procedimentos de avaliação diferentes.

No Brasil, o DNIT (2003), através da norma de Terminologia (TER) DNIT 005/2003 - TER, menciona a existência de 8 tipos de defeitos, alguns com mais de um tipo de classificação, como por exemplo as fendas, que são consideradas um defeito só, mas possuem diversas subclassificações (*i.e.* Fissuras, Trincas longitudinais, Trincas Transversais, Trincas por fadiga, entre outros). Já a ASTM (2024), por meio da ASTM D6433, considera que existem 20 defeitos, cada um com 3 níveis de severidade.

Além de normas técnicas, manuais também são bastante empregados, no

Brasil destaca-se o Manual de Identificação de Defeitos (MID), elaborado por Domingues (1993), que aborda 24 defeitos diferentes. No âmbito internacional, têm-se o *Distress Identification Manual* (DIM) desenvolvido para o programa *Long-Term Pavement Performance* (LTPP), da *Federal Highway Administration* (FHWA) de autoria de Miller e Bellinger (2014), que considera 15 defeitos. No Quadro 1 é apresentada uma síntese de quais defeitos estão presentes em cada um dos manuais e normas supracitados.

Quadro 1 - Síntese dos defeitos de superfícies de pavimentos asfálticos

ID	Tipo de Defeito	ASTM D6433 (2024)	DIM (2014)	DNIT (2003)	MID (1993)
1	Trincas por Fadiga	X	X	X	X
2	Exsudação	X	X	X	X
3	Trinca em bloco	X	X	X	X
4	Saliências/Deformações	X			
5	Corrugação	X		X	X
6	Depressão	X			X
7	Trincas de borda	X	X		X
8	Trinca por propagação de juntas	X	X		X
9	Desnível entre pista e acostamento	X	X		X
10	Trincas Longitudinais e Transversais	X	X	X	X
11	Remendo	X	X	X	X
12	Agregado Polido	X	X		X
13	Buraco	X	X	X	X
14	Passagem de ferrovia	X			
15	Afundamento de trilha de roda	X	X	X	X
16	Deformação plástica do revestimento (Escorregamento)	X	X	X	X
17	Trinca parabólica	X			X
18	Empolamento	X			X
19	Desintegração	X	X		X
20	Desgaste (Intemperismo)	X			X
21	Bombeamento e afloramento de água		X		X
22	Fissuras			X	X
23	Trincas de retração			X	
24	Desagregação				X
25	Separação entre pista e acostamento				X

Fonte: Autoria Própria (2025)

Conforme é notável no Quadro 1, existem 25 tipos de defeitos que podem ocorrer na superfície dos pavimentos asfálticos. Porém, de acordo com Danieleski (2004), defeitos como as trincas de borda, desnível pista e acostamento e cruzamento ferroviário não são significativos nos pavimentos urbanos. Dentre os defeitos mais frequentes em pavimentos urbanos, Llopis-Castelló *et al.* (2021) destacam as trincas (longitudinais, transversais e por fadiga), os buracos, os remendos e as desintegrações.

Além da identificação do tipo do defeito, é de suma importância o conhecimento do nível de severidade, item levado em consideração no MID, no DIM e na ASTM D6433. O nível de severidade consiste no estágio da evolução do defeito e pode ser classificada como baixa, média ou alta. O conhecimento dessa informação permite um melhor direcionamento de recursos para execução de atividades de M&R.

No presente trabalho, três defeitos frequentes em pavimentos urbanos são o objeto de estudo: as trincas por fadiga, os buracos e os remendos.

2.2.1 Trinca por Fadiga

Para a determinação do nível de severidade das trincas por fadiga, a ASTM (2024) postula que na severidade baixa as trincas existentes são extremamente finas e paralelas, com pouca ou nenhuma interconexão e sem erosões. Já para a severidade média, as trincas possuem uma maior evolução em relação a severidade baixa, com maiores interconexões e podendo apresentar leves erosões. Para a severidade alta, o padrão das trincas é bem definido e apresenta erosões nas bordas, além disso algumas peças podem movimentar com o tráfego. Na Figura 1 é possível visualizar os três níveis de severidade nas trincas por fadiga.

Figura 1 - Níveis de severidade das trincas por fadiga

Fonte: Autoria Própria (2025)

2.2.2 Buracos

Os níveis de severidade do defeito buraco, de acordo com a ASTM (2024), são definidos com base no diâmetro e na profundidade, conforme é possível observar na Tabela 1. Na Figura 2, é apresentado um exemplo de buraco com cada um dos 3 níveis de severidade.

Tabela 1 - Níveis de severidade para buracos

	Diâmetro médio (mm)		
Profundidade Máxima (mm)	100 a 200 mm	200 a 450 mm	450 a 750 mm
< 25 mm	Baixa	Baixa	Média
> 25 e < 50 mm	Baixa	Média	Alta
> 50mm	Média	Média	Alta

Fonte: Adaptado de ASTM (2024)

Figura 2 - Níveis de severidade dos buracos

Severidade baixa **Severidade média** **Severidade alta**

Fonte: Autoria Própria (2025)

2.2.3 Remendos

O remendo, apesar de ser uma medida corretiva, é considerado um tipo defeitos pela ASTM (2024), uma vez que a área remendada não vai performar tão bem quanto a seção original do pavimento. No Brasil o DNIT (2010), através da Especificação de Serviço (ES) DNIT 154/2010, normatiza a forma correta de execução de um remendo. Porém, na prática, essa norma não é seguida, resultando em remendos mal executados. Na Figura 3 é possível observar a diferença entre um remendo bem executado e um remendo mal executado.

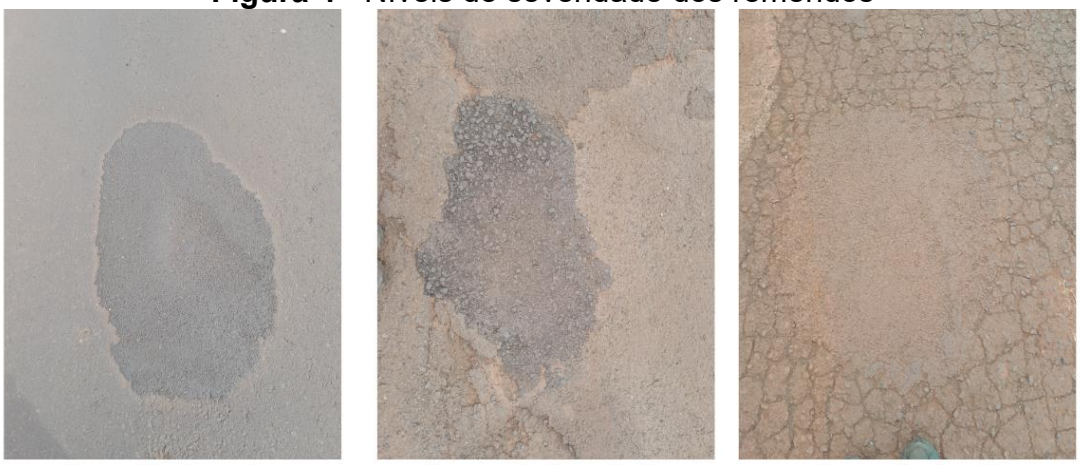
Figura 3 - Remendo bem executado e remendo mal executado

Remendo bem executado **Remendo mal executado**

Fonte: Autoria Própria (2025)

Conforme a ASTM (2024), esse defeito apresenta o nível de severidade baixo quando ele possui boa condição e desempenho satisfatório, nível de severidade médio quando está moderadamente deteriorado e nível de severidade alto quando apresenta deterioração elevada e necessita de substituição em breve. Na Figura 4 é apresentado um exemplo de cada nível de severidade para o defeito remendo.

Figura 4 - Níveis de severidade dos remendos



Fonte: Autoria Própria (2025)

3 INTELIGÊNCIA ARTIFICIAL

O campo da IA, de acordo com Russell e Norvig (2021), concentra-se na compreensão e desenvolvimento de entidades inteligentes, máquinas capazes de calcular como agir de forma eficaz e segura em diversas situações. A IA é uma área de estudo muito ampla e possui diversas subáreas, como o ML, a visão computacional, o processamento de linguagem natural, o reconhecimento de voz, a robótica e entre outros. No presente trabalho serão abordados apenas as subáreas ML e visão computacional, uma vez que apenas estas têm relação com a temática proposta.

3.1 MACHINE LEARNING

Existe uma certa confusão para o público em geral entre os termos “inteligência artificial” e “aprendizado de máquina”. O aprendizado de máquina, ou *Machine Learning* (ML), é uma subárea da IA que estuda a capacidade de melhorar o desempenho com base na experiência. Alguns sistemas de IA utilizam métodos de ML, mas outros não (Russell; Norvig, 2021).

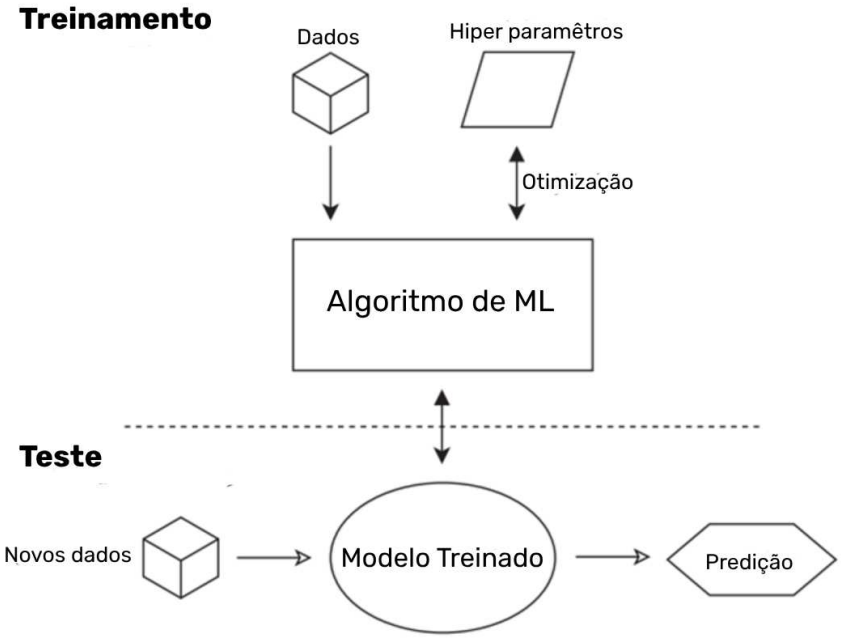
O ML fornece aos sistemas a capacidade de aprender e aprimorar a partir da experiência, de forma automática, através de um processo chamado treinamento (Sarker, 2021a). No treinamento o algoritmo é alimentado com uma grande quantidade de dados dos quais ele irá identificar padrões e fazer previsões, além disso, são utilizados hiper parâmetros com o intuito de otimizar esse processo (Verbraeken *et al.*, 2020). Um dos hiper parâmetros mais relevantes é o número de épocas, que consiste em uma passagem completa do algoritmo pelo conjunto de dados (Afaq; Rao, 2020). O resultado final da etapa de treinamento consiste em um modelo treinado (Verbraeken *et al.*, 2020).

Vale ressaltar que na etapa de treinamento, os dados empregados podem ser rotulados (*i.e.* quando os dados possuem anotações que indicam seu significado) ou não rotulados (*i.e.* quando não possuem anotações). Ressalta-se também que, durante o treinamento, é fornecido ao algoritmo um conjunto de dados de validação. Esses dados são diferentes dos empregados para o treinamento e são utilizados para o cálculo das métricas de avaliação. As métricas de avaliação são calculadas de forma periódica durante o treinamento, visando monitorar o desempenho do modelo. Mais

detalhes sobre as essas métricas serão abordados no tópico 3.5.

Após o treinamento, é realizada a etapa de predição (também chamada de etapa de teste). Nessa etapa, o modelo treinado é apresentado a um novo conjunto de dados, na qual, predições são realizadas para verificar o funcionamento do modelo na prática (Verbraeken *et al.*, 2020). Na Figura 5 é apresentado o funcionamento das etapas mencionadas.

Figura 5 - Treinamento e teste no ML



Fonte: Adaptado de Verbraeken *et al.* (2020)

Existem quatro formas de aprendizado no ML: o supervisionado, o não supervisionado, o semi-supervisionado e o por reforço (Sarker, 2021a). No aprendizado não supervisionado, os algoritmos identificam fatores-chave e classificam dados não rotulados com base em suas correlações, já os algoritmos de aprendizado supervisionado são treinados com dados já rotulados, minimizando assim erros e classificações incorretas do modelo (Dong *et al.*, 2021). No aprendizado semi-supervisionado, por sua vez, o dados submetidos ao modelo são uma mistura de dados rotulados e não rotulados (Sarker, 2021a). Por fim, o aprendizado por reforço visa utilizar observações coletadas da interação com o ambiente para tomar ações, sem a necessidade de um banco de dados tradicional, uma vez que os dados são gerados a cada interação (Sarker, 2021a).

Ressalta-se que o presente trabalho abordará com mais detalhes apenas o

aprendizado supervisionado, uma vez que todos os algoritmos empregados neste presente trabalho são supervisionados.

Dong *et al.* (2021) afirmam que os algoritmos mais populares de aprendizado supervisionado são as Redes Neurais Artificiais (RNAs). Para Wei *et al.* (2019) uma RNA consiste em um sistema de processamento de informações não programado e adaptativo, inspirado no sistema nervoso humano e formado por várias unidades de processamento interconectadas chamadas de neurônios artificiais. McCulloch e Pitts (1943) foram os pioneiros no estudo das RNAs, pois foram os primeiros a proporem um neurônio artificial por meio de um modelo matemático.

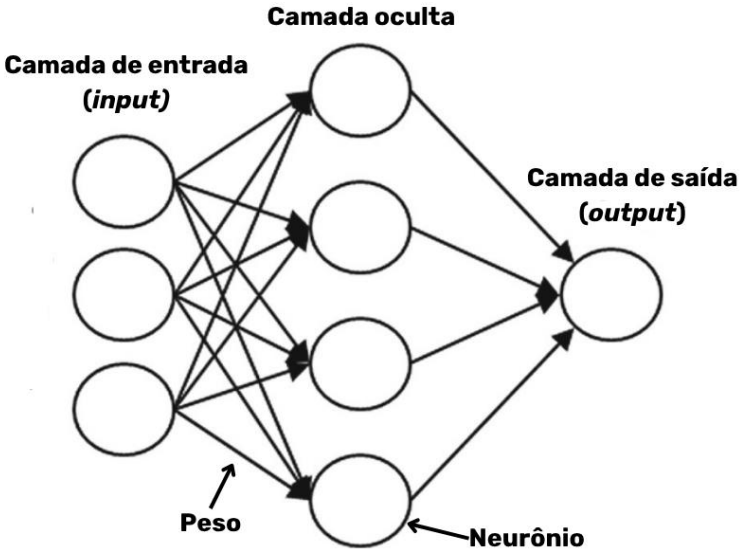
Existem diversos tipos de RNAs, dentre elas as *Shallow Neural Networks* (SNN) e as *Deep Neural Networks* (DNN), onde "*deep*" (profundo) e "*shallow*" (raso) são referentes a quantidade de camadas presentes nas redes. As SNN são arquiteturas mais simples, associadas ao *Shallow Learning* (SL) e as DNN são arquiteturas mais complexas e fazem parte do campo do *Deep Learning* (DL). SL e DL são as duas subdivisões do ML, que serão abordadas a seguir.

3.1.1 *Shallow Learning*

O SL, de acordo com Jafari, Moradi e Shafiee (2024), compreende modelos que não possuem arquiteturas complexas e normalmente trabalham com dados estruturados (números e caracteres). Esses modelos são adequados para resolver problemas relativamente mais simples e são eficazes quando as características dos dados são aparentes. Xu *et al.* (2021) postulam que os algoritmos de SL são aqueles lançados antes de 2006, dentre eles estão incluídos os *Support Vector Machines* (SVM), as *Decision Trees* (DT), o *K-nearest Neighbors* (KNN), e as SNN.

As SNN (Figura 6), são RNAs que possuem três camadas: uma camada de entrada, uma única camada oculta e uma camada de saída. A camada de entrada é responsável pelo recebimento de dados, que são processados pela camada oculta e transmitidos para camada de saída (Sarker, 2021b). Esse procedimento é realizado por neurônios, que são conectados através de links de conexão, que possuem pesos associados. Os neurônios da camada oculta aplicam uma função de ativação aos dados de entrada, transmitindo-os para a camada de saída (López; López; Crossa, 2022). Existem diversas funções de ativação, dentre elas: *Rectified Linear Unit* (ReLU), *Tanh*, *Sigmoid* e *Softmax* (Sarker, 2021b).

Figura 6 - *Shallow Neural Networks*



Fonte: Adaptado de Pradeep *et al.* (2022)

As técnicas de SL possuem diversas limitações, dentre elas, a capacidade de processar dados na sua forma “bruta”. Lecun, Bengio e Hinton (2015) comentam que por décadas, construir um sistema de ML capaz de reconhecer padrões exigia uma engenharia cuidadosa e domínio em programação para projetar um extrator de características que transformasse dados brutos (*e.g.* pixels de uma imagem) em uma representação interna adequada a partir do qual o sistema de aprendizagem pudesse detectar os padrões na entrada Este problema só foi contornado com o surgimento do DL, no ano de 2006.

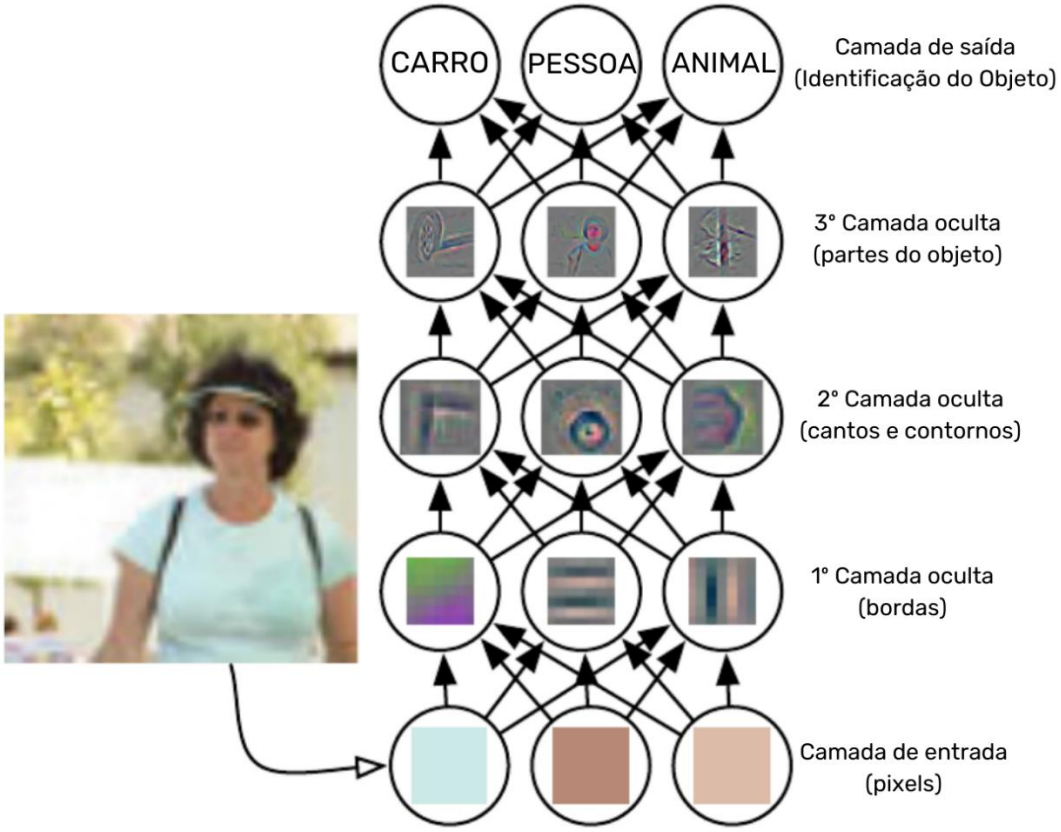
3.1.2 *Deep Learning*

O DL é considerado por si só uma ramificação do ML e envolve a utilização de DNNs, redes com múltiplas camadas ocultas capazes de aprender automaticamente padrões intrincados de dados não estruturados (imagens, áudios e vídeos). Dentre as arquiteturas de DNN mais utilizadas no DL, Jafari, Moradi e Shafiee (2024).citam as *Convolutional Neural Networks* (CNN) e as *Recurrent Neural Networks* (RNN)

Na Figura 7 é possível observar a ilustração de uma arquitetura de DL (mais especificamente uma CNN) com uma camada de entrada, três camadas ocultas e uma camada de saída. A camada de entrada (*input*) contém os pixels provenientes da imagem submetida ao modelo. As camadas ocultas são responsáveis pela extração

de características desses pixels, primeiramente as características mais simples (bordas, cantos e contornos) e em seguida as mais complexas (partes de objetos). Ao final, na camada de saída (*output*), o modelo consegue identificar o “objeto” presente na imagem.

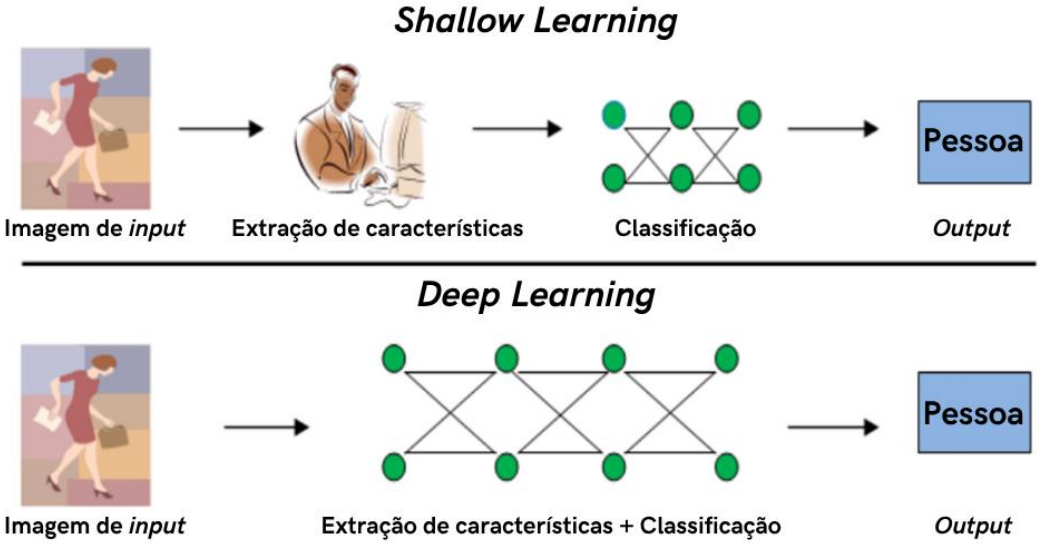
Figura 7 - *Deep Neural Network*



Fonte: Adaptado de Goodfellow, Bengio e Courville (2016)

O aspecto fundamental das arquiteturas de DL é que as características extraídas pelas camadas ocultas são aprendidas a partir de dados (Lecun; Bengio; Hinton, 2015). Enquanto no SL, esse procedimento é realizado manualmente, por meio da utilização de algoritmos de extração para posterior aplicação de algoritmos de aprendizado (Alom *et al.*, 2019). Essa diferença pode ser visualizada na Figura 8.

Figura 8 - *Shallow Learning e Deep Learning*



Fonte: Adaptado de Krishna e Kalluri (2019)

O início das pesquisas voltadas para o DL ocorreram em 2006. Neste ano Hinton e Salakhutdinov (2006) propuseram uma nova estratégia de pré-treinamento, contornando as limitações da época e possibilitando o treinamento de DNNs. Após essa publicação, Zhong; Ling; Wang (2019) constatam que rapidamente surgiram outros artigos propondo ideias similares e correlatas.

É válido mencionar que antes de 2006 alguns pesquisadores já haviam utilizados redes neurais com múltiplas camadas ocultas, como por exemplo a LeNet-5, proposta por Lecun *et al.* (1998). Porém, segundo Xu *et al.* (2021), algumas limitações da época impediam o avanço das pesquisas nesta área como a quantidade de dados e a capacidade computacional dos *hardwares*.

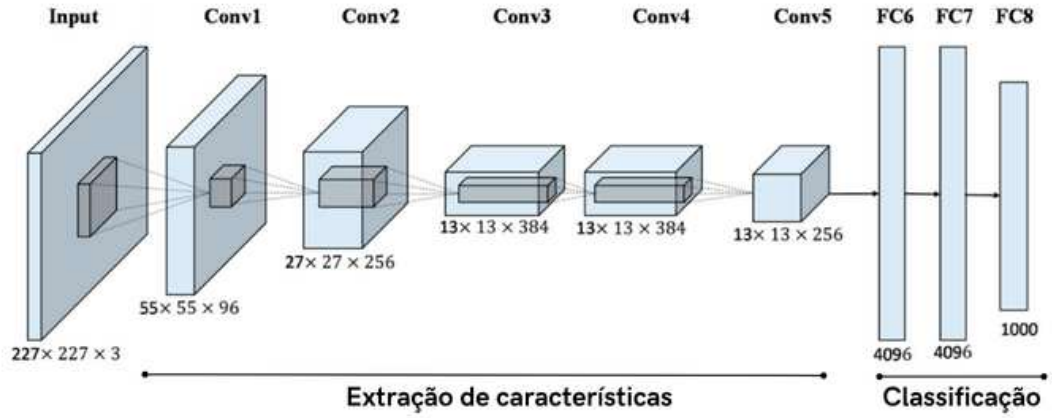
Em 2012 outro marco para o DL ocorreu, Krizhevsky; Sutskever; Hinton (2012) propuseram a CNN AlexNet. Os autores, treinaram essa rede com o banco de dados ImageNet, que na época continha 15 milhões de imagens anotadas (distribuídas em 22,000 categorias), obtendo uma taxa de erro de 15,3%. Zhong, Ling e Wang, (2019) postulam que este acontecimento não apenas mostrou a vantagem dos algoritmos de DL sobre os de SL, mas também atraiu a atenção da indústria e da área acadêmica para o DL. Desde então, diversas CNNs vem sendo desenvolvidas, dentre as quais é possível destacar a VGG (Simonyan; Zisserman, 2015), a ResNet (He *et al.*, 2015) e a DarkNet-53 (Redmon; Farhadi, 2018).

As CNNs, segundo Alom *et al.* (2019), são classificadas como algoritmos de DL de aprendizado supervisionado, devido a necessidade de dados anotados no

processo de treinamento. Esse tipo de rede possui diversas vantagens sobre outras DNNs, incluindo a capacidade de processar imagens 2D e 3D e serem efetivas em aprender características desses tipos de dados. Alom *et al.* (2019) complementam mencionando que geralmente as arquiteturas das CNNs consistem em duas partes principais: a extração de características e a classificação

Na figura 9 é possível visualizar a já mencionada AlexNet, na qual as primeiras 5 camadas dessa rede são responsáveis pela extração das características da imagem submetida como *input*, já as 3 últimas são camadas *fully-connected*, onde as características aprendidas são utilizadas para fazer classificações.

Figura 9 - Arquitetura da AlexNet



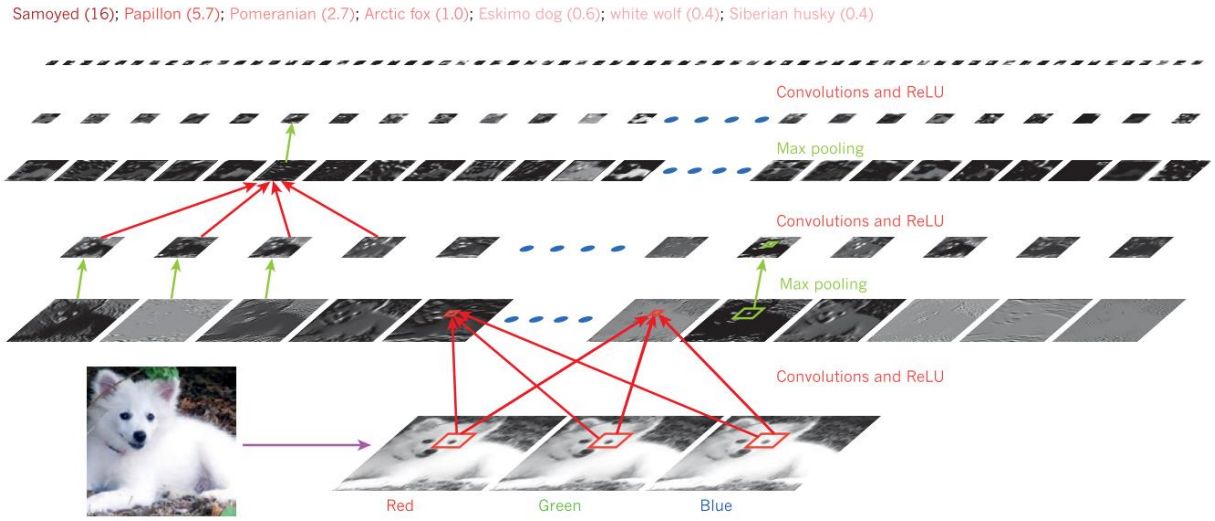
Fonte: Adaptado de Han et al. (2017)

Nas CNNs a etapa de extração de características é composta de dois tipos de camadas: camadas convolucionais e camadas de *pooling* (Lecun; Bengio; Hinton, 2015). As camadas convolucionais processam cada região local da imagem utilizada como input de forma independente, utilizando parâmetros compartilhados através de uma operação chamada convolução. Essa operação pode ser considerada como um pequeno filtro deslizante que é passado por toda a imagem, criando um mapa de características (Zhao, Xia *et al.*, 2024). A camada de *pooling*, por sua vez, tem o objetivo de combinar características semanticamente semelhantes em uma única característica (Lecun; Bengio; Hinton, 2015).

Na Figura 10 é notável o funcionamento das camadas iniciais de uma CNN, onde cada retângulo corresponde a um mapa de características obtido através de filtros aplicados a imagem submetida como *input*. As informações, neste caso, fluem de baixo para cima, onde as características mais simples da imagem são aprendidas nas primeiras camadas e as mais complexas nas últimas. Ao final, no *output*, uma

pontuação é computada para cada classe de imagem (Lecun; Bengio; Hinton, 2015).

Figura 10 - *Convolutional Neural Network*



Fonte: Lecun, Bengio e Hinton (2015)

Devido a essa capacidade das arquiteturas de CNN de identificarem padrões e características em imagens, elas são amplamente utilizadas na área da visão computacional.

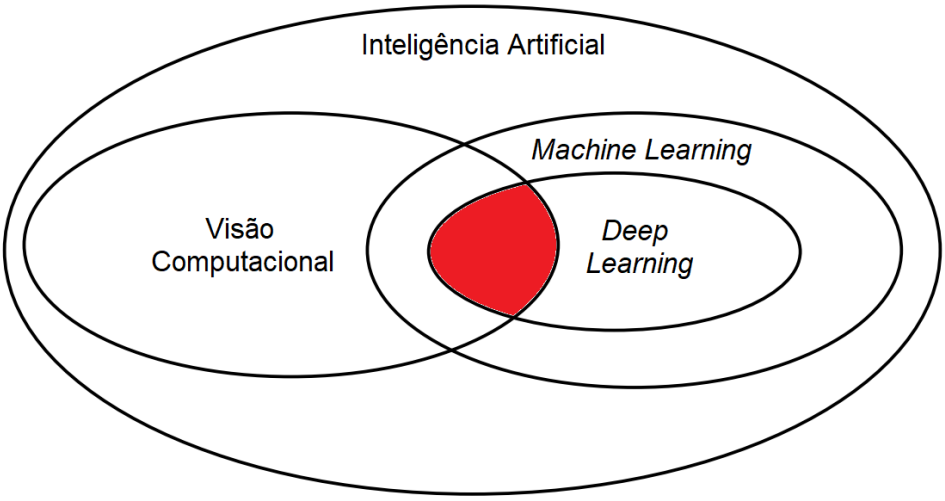
3.2 VISÃO COMPUTACIONAL

A visão computacional é um campo de estudo interdisciplinar, que estuda como reproduzir em um computador a capacidade de ver (Torralba; Isola; Freeman, 2024). A utilização dos métodos de visão computacional não são recentes, sendo aplicada em pesquisas na década de 60 (Spencer; Hoskere; Narazaki, 2019).

O divisor de águas desta área ocorreu apenas com o lançamento da AlexNet, em 2012. Torralba, Isola e Freeman (2024) enfatizam que após este acontecimento o aprendizado se tornou o ingrediente-chave em quase todas as pesquisas sobre visão computacional, sendo difícil encontrar, após 2020, sistemas de visão computacional que não utilizem DL.

Na Figura 11 é apresentado um Diagrama de Venn, que mostra a relação entre a visão computacional com a IA, ML e DL.

Figura 11 - Diagrama de Venn



Fonte: Adaptado de Johnson (2019)

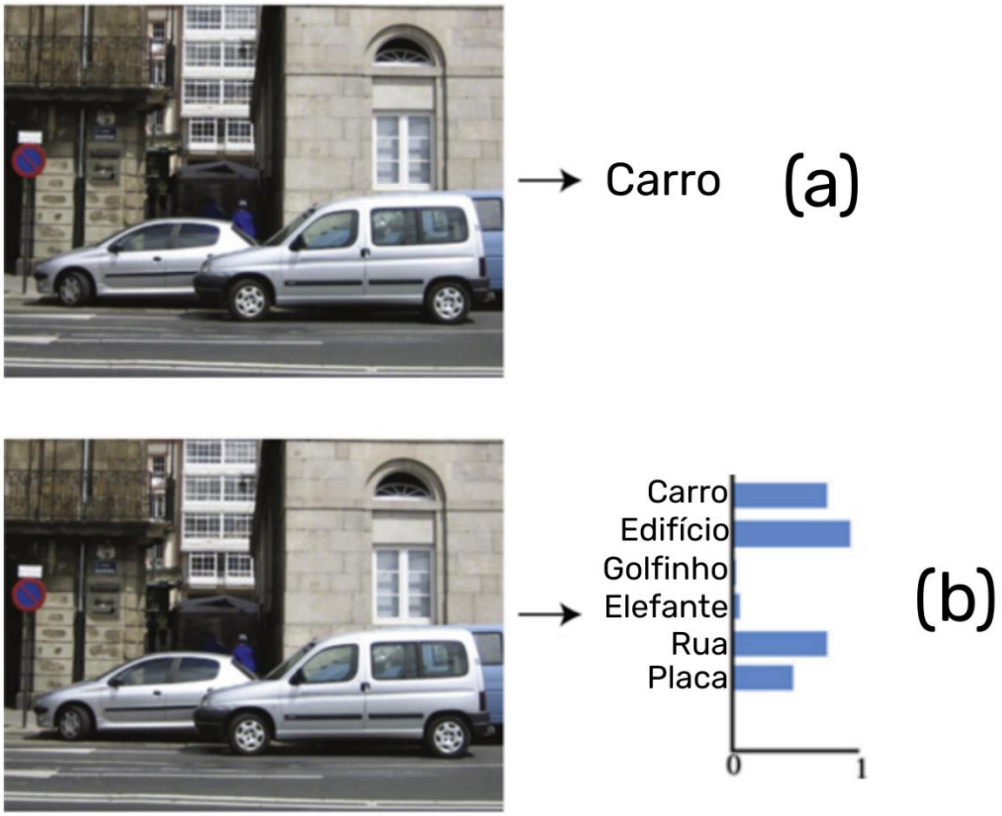
Nota-se, através da Figura 11, que a visão computacional possui uma intersecção (marcada em vermelho) com o DL, uma vez que algoritmos de DL são utilizados em subcampos da visão computacional. Dentre esses algoritmos, destacam-se as CNN, que são empregadas na classificação de imagens, detecção de objetos e segmentação de objetos, atividades da visão computacional que serão abordados a seguir.

3.2.1 Classificação

A classificação de imagens, segundo Zhao *et al.* (2024), é um dos desafios fundamentais da visão computacional, e seu objetivo é identificar entre classes distintas de objetos quais estão presentes em uma imagem ou vídeo.

Esse tipo de atividade pode ser realizado com uma ou múltiplas classes, conforme ilustrado na Figura 12. No caso da Figura 12a, o classificador foi treinado apenas para classificar carros, então o *output* será apenas uma classe. Já na Figura 12b, o modelo é capaz de identificar múltiplas classes presentes na imagem de *input*.

Figura 12 - Classificação de Imagens



Fonte: Adaptado de Torralba, Isola e Freeman (2024)

De acordo com Zhao *et al.* (2024), existem diversos tipos de classificadores, porém as CNNs se destacam, uma vez que elas aprendem automaticamente características a partir de uma quantidade massiva de dados, fornecendo aos modelos treinados habilidades de generalização mais eficazes quando comparados aos algoritmos convencionais de classificação de imagens que extraem recursos manualmente.

Existem várias CNN capazes de classificar imagens, dentre elas: AlexNet (Krizhevsky; Sutskever; Hinton, 2012), VGG (Simonyan; Zisserman, 2015) e ResNet (He *et al.*, 2015).

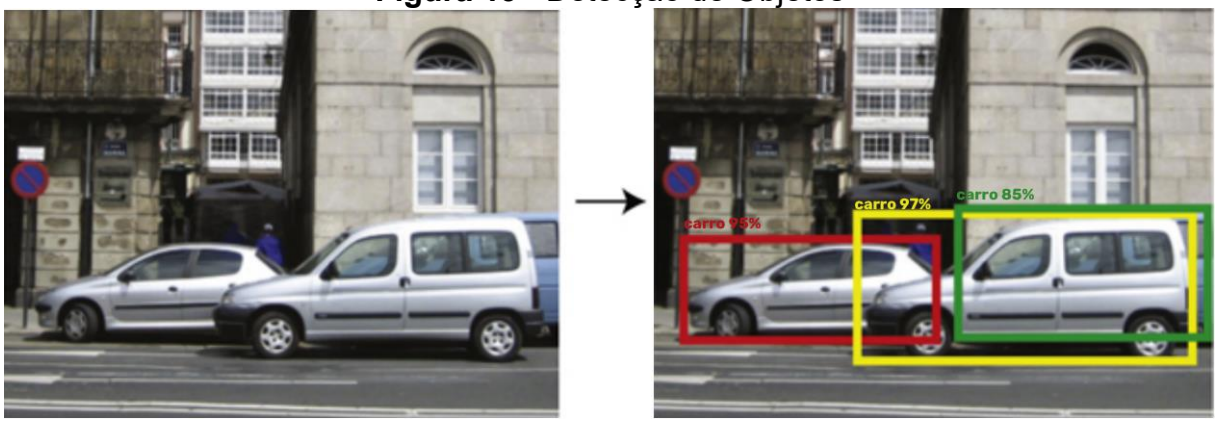
3.2.2 Detecção de Objetos

Diferentemente da classificação, que apenas indica quais objetos estão presentes em uma imagem, a detecção de objetos se concentra em identificar regiões locais de uma imagem e conjuntos específicos de classes de objetos (Zhao, Xia *et al.*, 2024). Para tal são utilizadas *bounding boxes*, que são retângulos que delimitam a localização do objeto de interesse na imagem (Torralba; Isola; Freeman, 2024). Além

de delimitar a localização, as *bounding boxes* também indicam um pontuação de confiança, expressando o quão confiante o modelo está sobre a previsão realizada (Rainio; Teuho; Klen, 2024).

Na figura 13 é possível observar o funcionamento de um detector de objetos, onde o *output* consiste na imagem de *input* com *bounding boxes* indicando os carros (objetos de interesse) e sua porcentagem de confiança.

Figura 13 - Detecção de Objetos



Fonte: Adaptado de Torralba, Isola e Freeman (2024)

Os modelos de detecção baseados em DL, conforme Kang *et al.* (2022), podem ser divididos *one-stage detector* e *two-stage detector.* Os *one-stage* realizam a localização e classificação do objeto simultaneamente, ao passo que os *two-stage* primeiramente propõem regiões de interesse, para em seguida efetuar a classificação e determinação das classes.

Dentre os *two-stage detectors* é possível citar o R-CNN (Girshick *et al.*, 2014) e seus sucessores, Fast R-CNN (Girshick, 2015) e Faster R-CNN (Ren *et al.*, 2015). Já dentre os *one-stage detectors*, é possível citar o SSD (Liu *et al.*, 2016), RetinaNet (Lin *et al.*, 2017) e as diversas versões da arquitetura *You Only Look Once* (YOLO), que será abordada com mais detalhes no tópico 3.3.

3.2.3 Segmentação

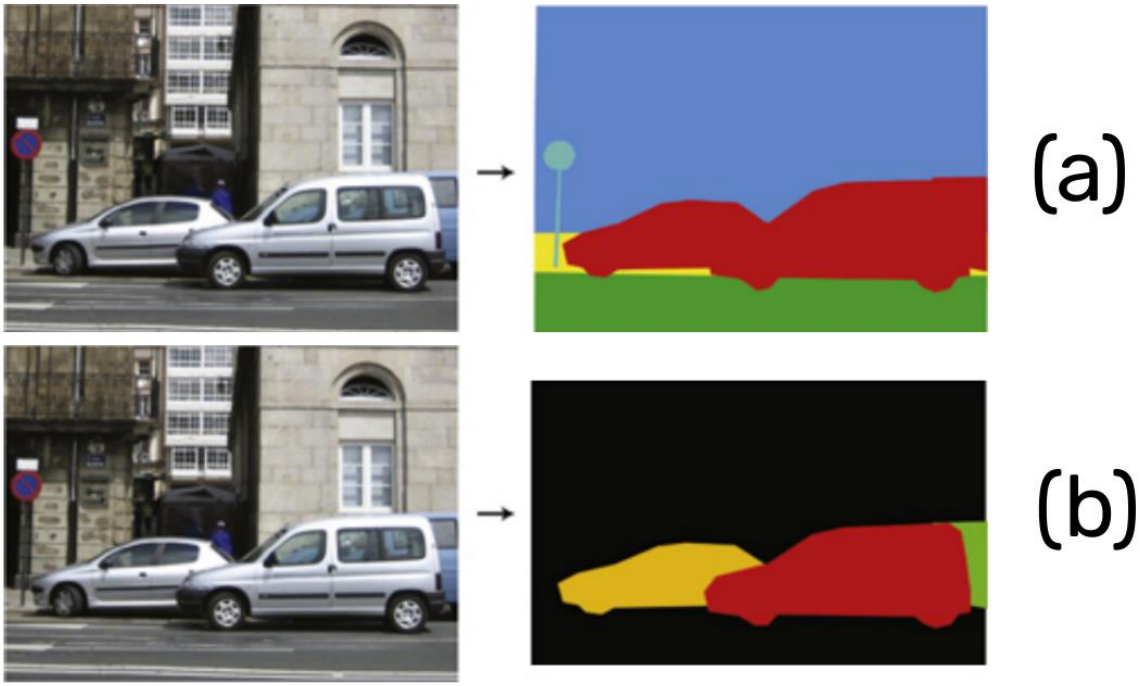
Um objeto é algo localizado no espaço, mas existem outras coisas que não podem ser localizadas, como neblina, luz, entre outras. Portanto, nem tudo é bem descrito por uma *bounding box*. Uma alternativa seria classificar cada pixel presente em uma imagem com uma classe de objeto, o que é chamado de segmentação

semântica (Torralba; Isola; Freeman, 2024).

Existem também outras atividades de segmentação, como a segmentação de instâncias, que de acordo com Hafiz e Bhat (2020) visa prever, em uma imagem, a classe de um objeto e a máscara de instância do objetivo específica para cada pixel. Dessa forma, He *et al.* (2017) constatam que a segmentação de instâncias combina a segmentação semântica e a detecção de objetos.

A diferença da segmentação de instâncias em relação à segmentação semântica é que, se houver X objetos do mesmo tipo na imagem, o modelo de segmentação de instâncias produzirá X máscaras. Enquanto isso, a segmentação semântica fornece apenas a região agregada de pixels, não indicando a quantidade de objetos do mesmo tipo e nem diferenciando dois objetos do mesmo tipo que se sobrepõem (Torralba; Isola; Freeman, 2024). Na Figura 14a é possível observar a segmentação semântica e na Figura 14b a segmentação de instâncias.

Figura 14 - Segmentação semântica e segmentação de instâncias



Fonte: Adaptado de Torralba, Isola e Freeman (2024)

Com o advento do DL e das CNNs, muitas arquiteturas de segmentação semântica e segmentação de instância foram desenvolvidas. Das arquiteturas que utilizam segmentação semântica é possível mencionar a U-Net (Ronneberger; Fischer; Brox, 2015), FCN (Shelhamer; Long; Darrell, 2015) e SegNet (Badrinarayanan; Kendall; Cipolla, 2017). Das que utilizam segmentação de instância

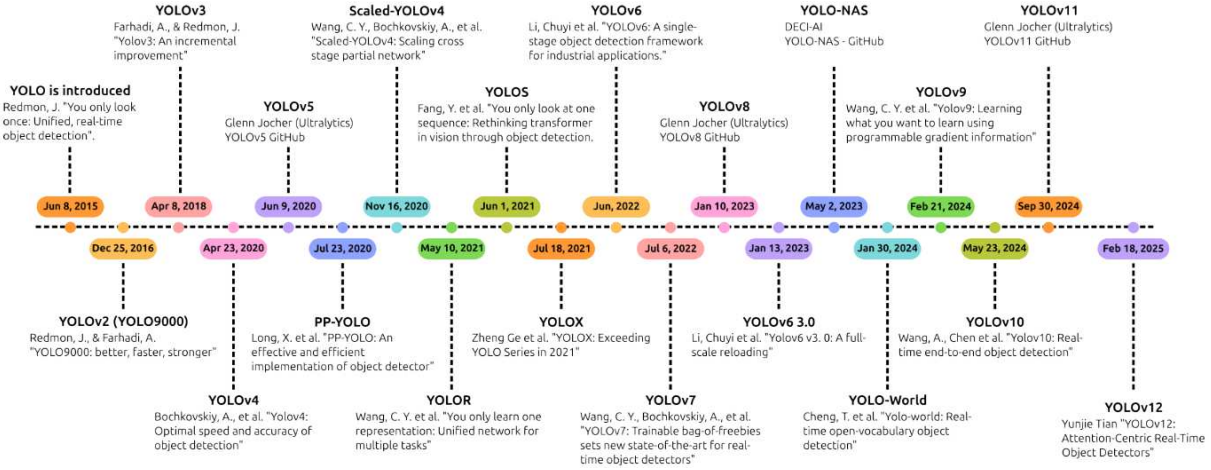
destacam-se a Mask R-CNN (He *et al.*, 2017) e a HTC (Chen *et al.*, 2019). As últimas versões da arquitetura YOLO também são capazes de realizar a segmentação de instancias, como a YOLOv8 (Ultralytics, 2023), YOLOv9 (Wang; Yeh; Liao, 2024), YOLOv11 (Ultralytics, 2024) e YOLOv12 (Tian; Ye; Doermann, 2025).

3.3 YOU ONLY LOOK ONCE

O *You Only Look Once* (YOLO) é um algoritmo de detecção de objetos que foi proposto inicialmente por Redmon *et al.* (2016). A primeira versão do YOLO simplificou a atividade de detecção de objetos, através de uma arquitetura de CNN *one-stage*, efetuando a localização e a classificação dos objetos simultaneamente. Ou seja, no YOLOv1 a CNN processa a imagem apenas uma vez para efetuar as predições, diferentemente dos detectores *two-stage* utilizados na época, justificando assim o nome "*you only look once*". O sistema proposto pelos autores alcançou uma velocidade de processamento superior aos detectores da época. Contudo, apesar da simplicidade e rapidez, a sua precisão era inferior. Esse fato mudou nos anos seguintes, com o surgimento de novas versões do algoritmo.

É importante destacar que devido à sua grande popularidade e por ser um algoritmo de código aberto (*open-source)*, uma nova versão do YOLO pode ser desenvolvida por qualquer grupo de pessoas. A Figura 15 mostra uma linha do tempo com as principais versões YOLO até o momento, bem como seus respectivos autores.

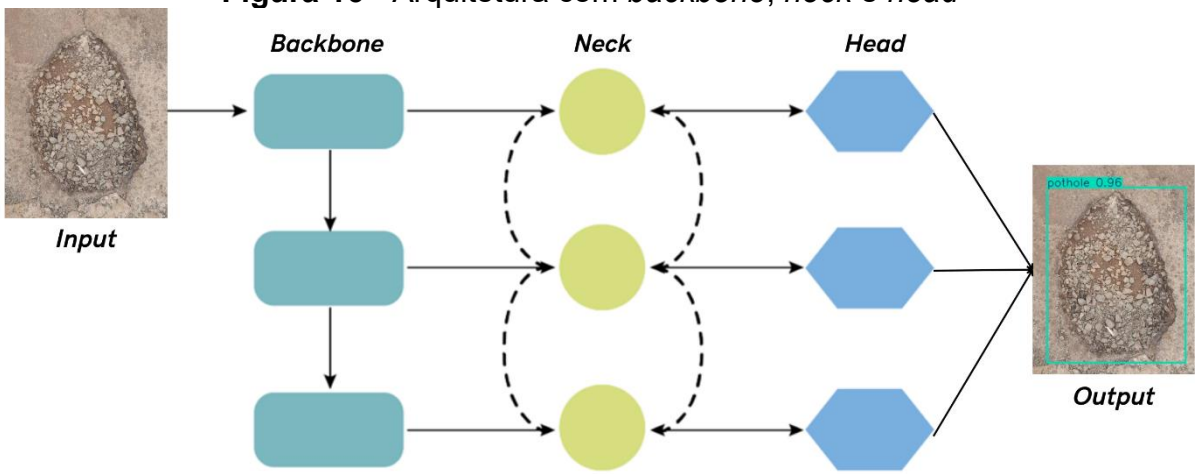
Figura 15 - Linha do tempo do YOLO



Fonte: Jegham *et al.* (2025)

De forma geral, as arquiteturas YOLO (a partir da YOLOv4) podem ser divididas em três partes: *backbone*, *neck* e *head*. O *backbone*, geralmente uma CNN, é responsável por extrair as características da imagem de entrada. Essas características são transferidas para o *neck*, que as agrega e refina, aprimorando assim as informações recebidas para posteriormente enviá-las para a *head*. Essa, que consiste no último componente de um detector de objetos, é responsável por realizar as predições na imagem. Tipicamente, a *head* consiste em uma ou mais módulos capazes de efetuar a localização, classificação e/ou segmentação na imagem (Terven; Córdova-Esparza; Romero-González, 2023). Na Figura 16 é possível observar uma arquitetura com esses três componentes.

Figura 16 - Arquitetura com *backbone*, *neck* e *head*



Fonte: Adaptado de Terven, Córdova-Esparza e Romero-González (2023)

Vale ressaltar que na presente dissertação, foram utilizadas as versões mais recentes dessa arquitetura, sendo elas: YOLONAS, YOLOv8, YOLOv11 e YOLOv12.

3.3.1 YOLOv8

O YOLOv8 foi desenvolvido pela Ultralytics (2023), empresa que também desenvolveu o YOLOv5 e posteriormente o YOLOv11. O YOLOv8, além da detecção de objetos suporta outras atividades de visão computacional, sendo elas: segmentação, estimativa de pose, rastreamento e classificação.

A arquitetura do YOLOv8 utiliza um *backbone* similar ao do YOLOv5, com algumas diferenças, que de acordo com Terven, Córdova-Esparza e Romero-González (2023) são: a utilização de uma CNN modificada (CSPDarknet53) no

backbone e a substituição do CSPLayer pelo módulo C2f que combina características de alto nível com informação de contexto para melhorar a acurácia do processo de detecção.

Além disso, o YOLOv8 apresenta uma camada chamada *Spatial Pyramid Pooling Fast* (SPPF), que acelera o processo de computação ao agrupar características em um mapa de tamanho fixo. No YOLOv8, cada operação de convolução apresenta normalização por lote e uma ativação SiLU. A *head* do YOLOv8 é desacoplada, com o intuito de processar as tarefas de regressão e classificação de forma independente. Também na *head* são utilizadas as funções de perda CIoU e DFL, visando melhorar a detecção de objetos, principalmente os pequenos (Terven; Córdova-Esparza; Romero-González, 2023).

O YOLOv8 apresenta diferentes versões, que variam em tamanho e, consequentemente, em desempenho. As diferentes versões do YOLOv8 são: YOLOv8n (*nano*), YOLOv8s (*small*), YOLOv8m (*medium*), YOLOv8l (*large*) e YOLOv8x (*extra-large*) (Ultralytics, 2023).

Geralmente, as versões com mais parâmetros, como a YOLOv8x, tendem a alcançar uma maior acurácia, porém o tempo de treinamento é maior e a velocidade de detecção menor. Já em versões com menos parâmetros, como a YOLOv8n, ocorre o contrário: o tempo de treinamento é menor, e a velocidade de detecção maior. Consequentemente, esses modelos possuem uma menor acurácia. Essa constatação aplica-se para todos os modelos YOLO com diferentes tamanhos.

3.3.2 YOLO-NAS

O YOLO-NAS foi lançado em maio de 2023 pela empresa DeciAi (2023). Essa empresa foi adquirida pela Nvidia em 2024, sendo dissolvida como uma entidade independente.

Esta arquitetura foi gerada automaticamente utilizando a técnica de *Neural Architecture Search* (NAS) por meio de um sistema chamado AutoNAC, também desenvolvido pela Deci AI (Terven; Córdova-Esparza; Romero-González, 2023). A NAS tem como objetivo buscar arquiteturas de rede para atividades específicas de forma automática. Para isso, primeiramente é definido um espaço de pesquisa, que contém uma grande quantidade de possíveis redes candidatas. Em seguida, a NAS

define estratégias para procurar as redes com melhor desempenho em um banco de dados específico. Dessa forma, a NAS é uma ferramenta prática que visa otimizar o *design* de redes neurais, apresentando vantagens sob o *design* manual (Liu; Zhang; Jin, 2022)

No YOLO-NAS, a NAS foi utilizada através do AutoNAC. Esse sistema foi utilizado para determinar os tipos e quantidade de blocos, quantidade de canais para cada estágio (*backbone*, *neck* e *head*), bem como os seus tamanhos e estruturas (Kumar; Kumar, 2024). Além disso, durante o processo de NAS, módulos de quantização QSP e QCI foram utilizados para minimizar a perda de acurácia pós-treinamento. Variando a profundidade e posição desses blocos, três arquiteturas YOLO-NAS foram geradas: YOLO-NAS-S (small), YOLO-NAS-M (medium) e YOLO-NAS-L (large) (Terven; Córdova-Esparza; Romero-González, 2023).

3.3.3 YOLOv11

O YOLOv11 foi lançado em setembro de 2024 e desenvolvido pela empresa (Ultralytics, 2024). Assim como o YOLOv8, o YOLOv11 também possui diferentes versões dependendo da quantidade de parâmetros (*i.e.* YOLOv11n, YOLOv11s, YOLOv11m, YOLOv11l e YOLOv11x) e tem a capacidade de realizar outras atividades de visão computacional além da detecção de objetos.

Uma melhoria significativa no YOLOv11 é a introdução do módulo C3k2, que substitui o C2f (utilizado na YOLOv8). O módulo C3k2 possui uma maior capacidade computacional, contribuindo para o processamento mais rápido da rede. O YOLOv11 também manteve o módulo SPPF, porém adicionou um novo em sequência, chamado de *Cross Stage Partial with Spatial Attention* (C2PSA). O C2SPA permite o modelo focar nas regiões mais importantes de uma imagem, potencializando a acurácia de detecção (Khanam; Hussain, 2024).

Com as melhorias no design do modelo, o YOLOv11 apresenta um desempenho melhor que suas versões anteriores. O YOLOv11m, por exemplo, atingiu o maior mAP no banco de dados Common Objects in Context (COCO), com 22% menos parâmetros que o YOLOv8m. Sendo assim eficiente, do ponto de vista computacional, sem comprometer a acurácia (Ultralytics, 2024).

3.3.4 YOLOv12

O YOLOv12, lançado em fevereiro de 2025, foi desenvolvido por Tian, Ye e Doermann (2025) e é a última versão da série YOLO até o momento. De forma similar as versões YOLOv8 e YOLOv11, a YOLOv12 também possui 5 variantes (YOLOv12n, YOLOv12s, YOLOv12m, YOLOv12l e YOLOv12x) e suporte para diversas atividades de visão computacional.

O grande diferencial da estrutura do YOLOv12 é que ela é centrada em mecanismos de atenção. Conforme Niu, Zhong e Yu (2021), esses mecanismos são inspirados na função de atenção do sistema biológico humano, função essa que tende a focar em partes importantes ao processar grandes quantidades de informações. A utilização desses mecanismos no DL, melhora a eficiência e a precisão no processamento de informações.

A arquitetura do YOLOv12 possui três principais modificações: (1) adição do módulo *Area Attention* (A2); (2) introdução do bloco *Residual Efficient Layer Aggregation Networks* (R-ELAN); (3) diversas melhorias na arquitetura geral do YOLO, como por exemplo a introdução do *FlashAttention.* Dessa forma, o YOLOv12 apresenta melhores métricas de avaliação que as versões anteriores (quando treinado no banco de dados MSCOCO 2017), mantendo uma rápida velocidade de inferência. (Tian; Ye; Doermann, 2025).

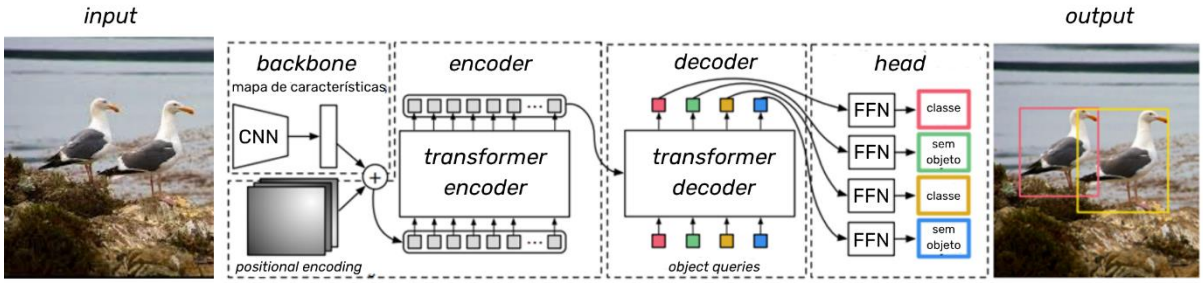
3.4 DETECTION TRANSFORMER

O *DEtection TRansformer* (DETR) é uma arquitetura proposta inicialmente por Carion *et al.* (2020), com o intuito de simplificar as arquiteturas de detecção de objetos, eliminando a necessidade de componentes elaborados manualmente (e.g. *non max supression* e geração de âncoras). Para isso os autores propuseram uma estrutura composta de três partes: uma CNN como *backbone*, um *transformer encoder-decoder* e uma simples *Feed Forward Network* (FFN) como *head*.

Na Figura 17 é possível observar a arquitetura do DETR. Nela, a imagem de entrada (*input*) é processada por uma CNN, que extrai as características da imagem, gerando um mapa de características 2D. Esse mapa é convertido para uma sequência de vetores 1D e em conjunto com uma codificação posicional espacial (*spatial*

positional encoding) é transferido para o *transformer encoder*. O *encoder* processa esses vetores utilizando mecanismos de autoatenção, que capturam as relações globais e o contexto da imagem. Em seguida, o *transformer decoder* recebe um número fixo de vetores aprendidos (*object queries*) que são processados em conjunto com as informações codificadas da imagem, fornecendo posteriormente um conjunto de vetores de saída. Esses vetores são passados para uma FFN, que efetua a previsão de classe e das *bounding boxes* (Carion *et al.*, 2020).

Figura 17 - Arquitetura do DETR



Fonte: Adaptado de Carion *et al.* (2020)

Com o passar do tempo outras versões da DETR foram propostas, possuindo melhorias em comparação à versão original. Algumas dessas arquiteturas são a *Deformable* DETR (Zhu *et al.*, 2021), DINO (Zhang *et al.*, 2022), RT-DETR (Zhao *et al.*, 2024), LW-DETR (Chen *et al.*, 2024) e RF-DETR (Robicheaux *et al.*, 2025). O presente trabalho irá se ater apenas nesta última, a versão mais recente da arquitetura DETR, desenvolvida pela empresa Roboflow.

O RF-DETR, atualmente, possui quatro versões: a RF-DETR-Nano; a RF-DETR-Small; a RF-DETR-Medium e a RF-DETR-Base. A arquitetura RF-DETR combina dois dos melhores algoritmos modernos DETR: o LW-DETR e um *backbone* pré-treinado DINOv2. Com essa utilização conjunta, o RF-DETR apresenta uma melhor performance no banco de dados COCO que os algoritmos YOLOv11 e YOLOv8 (Robicheaux *et al.*, 2025).

3.5 Métricas de Avaliação

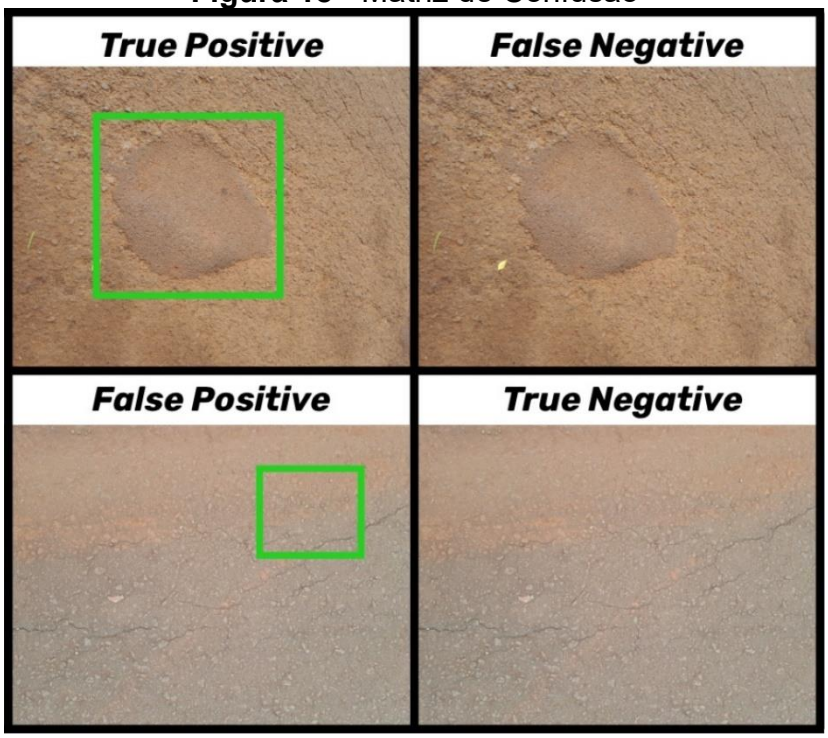
Ao longo deste capítulo foram mencionados diversos algoritmos de ML. Contudo, como essa área de estudo é bastante ampla, existem inúmeros outros que não foram abordados. Dada a quantidade de algoritmos existentes (e com novos

surgindo a cada momento), Rainio, Teuho e Klen (2024) destacam a importância de avaliá-los. Segundo os autores, a avaliação tem dois propósitos: algoritmos que apresentam desempenho insatisfatório podem ser descartados e aqueles que são promissores podem ser otimizados. Para realizar essa análise são utilizadas métricas de avaliação.

Existem diversas métricas de avaliação, e a escolha da métrica adequada depende do tipo de tarefa realizada pelo algoritmo de ML. Para algoritmos de classificação, por exemplo, são utilizadas métricas diferentes das empregadas na detecção de objetos. Porém, antes de abordar as métricas, é fundamental conceituar a matriz de confusão.

A matriz de confusão consiste em uma matriz 2x2, onde a diagonal principal representa as previsões corretas (acertos) do algoritmo e a diagonal secundária as previsões incorretas (erros). Os acertos são classificados como Verdadeiros Positivos (*True Positive* - TP) e Verdadeiros Negativos (True *Negative* - TN). Enquanto os erros são os Falsos Positivos (False Positive - FP) e os Falsos Negativos (False Negative - FN). Na Figura 18 é possível observar um exemplo de matriz de confusão aplicada no reconhecimento do defeito remendo.

Figura 18 - Matriz de Confusão



Fonte: Autoria Própria (2025)

Através da Figura 18, é perceptível que no TP o modelo realiza uma previsão correta, pois uma *bounding box* é marcada onde existe um remendo na imagem. No FN, isso não ocorre, uma vez que a imagem apresenta um remendo e o modelo não indica nenhuma *bounding box*, sendo assim, uma predição incorreta. No FP o modelo também realiza uma predição incorreta, pois a imagem não contém nenhum remendo e mesmo assim o modelo demarca uma *bounding box*. Por fim, no TN, o modelo realiza uma predição correta, pois a imagem não apresenta nenhum remendo e o modelo não efetua nenhuma detecção.

Por meio dos elementos da matriz de confusão (TP, FP, FN e TN) é possível realizar o cálculo de diversas métricas de avaliação. Uma métrica bastante utilizada, principalmente em algoritmos de classificação é a acurácia (Equação 1). A acurácia consiste em mensurar a proporção de previsões corretas (TP e TN) realizadas pelo modelo em relação a todas as previsões (TP, FN, FP e TN).

$$Acurácia = \frac{TP + TN}{TP + FN + FP + TN} \quad (1)$$

Apesar de ser amplamente utilizada em algoritmos de classificação, é válido ressaltar que a acurácia não é empregada na detecção de objetos, visto que nesta atividade, os TNs não são levados em consideração. Para os algoritmos de detecção de objetos, as métricas mais utilizadas são: Precisão, *Recall*, *F1-score*, AP (*Average Precision*) e mAP (*mean Average Precision*).

A Precisão (Equação 2) indica a capacidade do modelo de, dentre as previsões positivas (aquelas que contém uma *bounding box*, abrangendo tanto TP quanto FP), determinar quais de fato estão corretas, ou seja, os TPs. Já o *recall* (Equação 3), mensura a capacidade do modelo de identificar os objetos que realmente existem na imagem, ou seja, a proporção de TP, dentre todos os objetos que deveriam ter sido detectados (TP e FN). Por fim, o *f1-score* (Equação 4), consiste em uma média harmônica entre precisão e *recall*, visando observar o balanceamento entre essas duas métricas.

$$Precisão = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

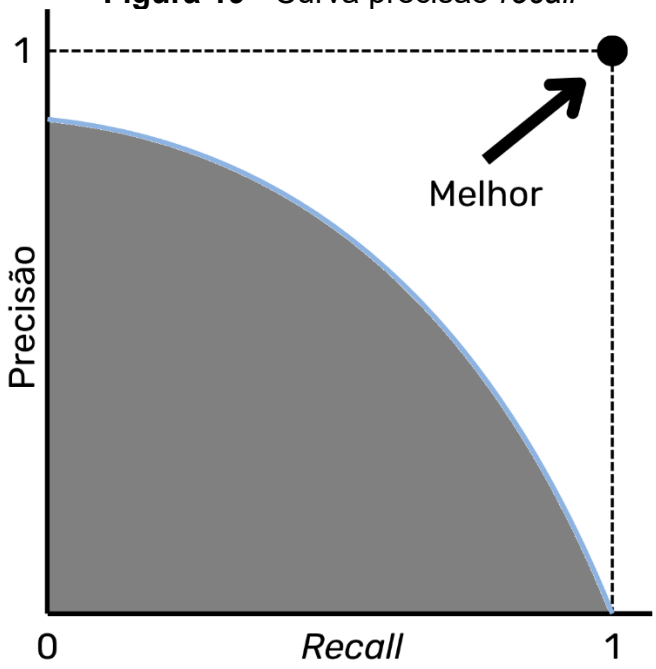
$$F1 - score = \frac{2 \times Precisão \times Recall}{Precisão + Recall} \quad (4)$$

Todas as métricas abordadas variam de 0 a 1. A precisão, quanto mais próxima de 1, indica que as previsões feitas pelo modelo (as que contém *bounding boxes*) estão corretas, ou seja, possuem poucos FP. O *recall*, por sua vez, quanto mais elevado, indica que o modelo consegue identificar os objetos de interesse (poucos FN). Já um f1-score elevado indica um bom equilíbrio entre as métricas precisão e *recall*.

Rainio, Teuho e Klen (2024) abordam que, como os detectores de objetos definem uma pontuação de confiança para cada *bounding box*, é possível estabelecer um limiar (*threshold*) de confiança e remover as previsões abaixo desse valor. Ou seja, se o *threshold* for definido como 50%, as detecções abaixo dessa porcentagem não são considerados pelo modelo.

A alteração do *threshold* afeta a quantidade de TP, FP e FN e, consequentemente a precisão e o *recall*. Dessa forma, é possível obter a curva precisão-*recall*. Essa curva pode ser obtida ao plotar a precisão (eixo Y) em função do *recall* (eixo X) para todos os limiares de confiança possíveis (Rainio; Teuho; Klen, 2024). Um exemplo de curva precisão-*recall* pode ser visualizado na Figura 19, sendo importante ressaltar que quanto mais próximo do canto superior direito (precisão e *recall* iguais a 1), melhor o desempenho do modelo.

Figura 19 - Curva precisão-*recall*



Fonte: Autoria Própria (2025)

A área abaixo dessa curva é utilizada para calcular a AP (Equação 5). Essa métrica é empregada para avaliar o desempenho do modelo para uma única classe. Quando o modelo possui mais de uma classe, seu desempenho é mensurado através do mAP (Equação 6), que consiste na média aritmética da AP para todas as classes. Essas métricas também variam de 0 a 1, e quanto mais próximo de 1, melhor é o modelo treinado.

$$AP = \int_0^1 Precisão\ (recall)\ drecall \tag{5}$$

$$mAP = \frac{1}{N} \times \sum_{i=1}^{N} APi \tag{6}$$

4 UTILIZAÇÃO DE DEEP LEARNING NA DETECÇÃO DE DEFEITOS E RECONHECIMENTO DO NÍVEL DE SEVERIDADE

O ML é amplamente utilizado atualmente, possuindo aplicação em diversas áreas do conhecimento, como a medicina, a agricultura e a engenharia. Na engenharia civil, esses algoritmos são bastante utilizados para o reconhecimento automático de defeitos em diversas estruturas civis como pontes, barragens, edificações e pavimentos.

A detecção de defeitos em pavimentos utilizando ML é um tema que vem sendo extensamente explorado na literatura a décadas, possuindo uma grande quantidade de publicações. Em uma revisão bibliométrica conduzida por Zheng *et al.* (2024), foram identificadas 543 publicações, no período de 2016 a 2024, que utilizaram o DL para detectar defeitos em pavimentos (asfálticos e de concreto). Esse número de publicações seria consideravelmente maior caso os autores tivessem considerado os artigos publicados antes de 2016 e artigos que empregaram algoritmos de SL.

Além de artigos científicos, a aplicação de ML no reconhecimento de defeitos também é utilizada na prática. No Brasil, por exemplo, desde 2022 o DNIT vem empregando o *software* DNIT-ICM, capaz de reconhecer, por meio do ML, buracos, remendos e trincas de forma automática em filmagens realizadas em rodovias pavimentadas.

Apesar da grande quantidade de artigos e eficiência comprovada de algoritmos de ML no reconhecimento de defeitos, Gopalakrishnan (2018), Ali *et al.* (2022) e Sholevar, Golroo e Esfahani (2022), constatam, em seus artigos de revisão de literatura, que o reconhecimento automático do nível de severidade utilizando ML é uma temática pouco abordada na literatura, necessitando de uma maior atenção devido a sua importância no SGP.

Em uma revisão sistemática conduzida por Souza e Fontenele (2025, submetido) foi identificada a existência de 28 artigos que aplicaram algoritmos de ML para a determinação do nível de severidade. Desses 28 artigos, 15 utilizaram DL, 9 SL e 4 empregaram ambas as técnicas. Dos 15 artigos que utilizaram DL, os autores verificaram que a determinação do nível de severidade ocorre de duas formas, intitulada pelos mesmos de Defeitos e Severidades Juntos (DSJ) e Defeitos e Severidades Separados (DSS).

Na primeira forma, a DSJ, é utilizando um banco de dados para treinamento

que contenham rótulos do defeito estudado e do seu respectivo nível de severidade, dessa forma, após o treinamento, o modelo será capaz de identificar o defeito e seu nível de severidade simultaneamente. Já na segunda forma, a DSS, o banco de dados empregado para treinamento contém rótulos apenas dos defeitos. Nesse caso, a determinação do nível de severidade é efetuada posterior ao treinamento do modelo.

4.1 Forma 1 – Defeitos e Severidades Juntos (DSJ)

A primeira forma, conforme Souza e Fontenele (2025, submetido), é a mais utilizada na literatura, sendo empregada, através de diferente critérios e arquiteturas, por 11 autores.

A arquitetura YOLOv3 foi utilizada por Peraka, Biligiri e Kalidindi (2020) na detecção de buracos e seus 3 níveis de severidade. Porém, apesar de obterem bons resultados (mAP de 89,23%), os autores não realizaram medições em campo, sendo considerada a textura das imagens para definir a severidade (*i.e* os autores assumiram que se a textura estava escura, a severidade dos buracos era alta e vice e versa). Em outro trabalho dos mesmos autores, Peraka, Biligiri e Kalidindi (2021) trabalharam com bancos de dados públicos de múltiplos defeitos (trincas, buracos e remendos), realizando o reconhecimento por meio da YOLOv4. Contudo, novamente os autores levaram em consideração a textura e não efetuaram aferições em campo.

Já Roberts, Inzerillo e Mino (2020) utilizaram um critério próprio, baseado na subjetividade, para determinar a severidade (baixa e alta) de trincas e deformações visco-plásticas. Esses autores efetuaram o treinamento nas arquiteturas SSD e Faster R-CNN, essa última obtendo os melhores resultados, com uma precisão de 93,33%, *recall* de 93,80% e *f1-score* de 93,56%. Os autores Hassan *et al.* (2023) também utilizaram a subjetividade para verificar a severidade de trincas. Efetuando o treinamento também nas arquiteturas SSD e Faster R-CNN, porém obtendo resultados inferiores, com a maior precisão sendo de 66,00%.

Hsieh, Yang e Tsai (2021) por sua vez, empregaram o *Laser Crack Measurement System* (LCMS) do *National Optics Institute* (INO), que consiste em um laser de linha e uma câmera de alta velocidade com o princípio de triangulação de luz estruturada para obter dados de trincas em pavimentos de concreto. As imagens foram treinadas em diversas arquiteturas ResNet50 propostas pelos autores, onde o melhor mAP foi de 85,03%. Ressalta-se que esses autores adotaram um critério que

o nível de severidade de trincas não depende de medições, não sendo necessária a avaliação em campo.

Treze algoritmos diferentes de classificação, sendo 1 MobileNetvV2, 2 ResNet, 2 DenseNet e 8 EfficientNet, foram utilizados por Liu, Liu e Wang (2022), para treinar imagens de trincas por fadiga de três diferentes severidades. As imagens foram coletadas em um campus universitário por meio de uma câmera infravermelha. Os melhores resultados ocorreram na EfficientNet-B4, com uma acurácia de 99,32%.

Alqethami *et al.* (2022) propuseram um novo modelo de classificação, chamado de RoadNet, capaz de reconhecer múltiplos defeitos e dois níveis de severidade (alto e baixo). A RoadNet obteve uma acurácia e precisão de 97%, e *recall* e *f1-score* de 96%, mostrando assim o bom desempenho do modelo proposto. Contudo, os autores não especificam se os níveis de severidades foram obtidos através de avaliação em campo ou não.

A arquitetura YOLOv5m foi utilizada por Valipour *et al.* (2023) para a detecção da severidade alta e baixa de trincas em bloco. Nesse trabalho, a severidade média foi incluída em conjunto com as imagens de severidade alta pois, conforme os autores, nesse tipo de trinca as severidades média e alta possuem diferenças de padrões que dificilmente são reconhecidas pelos modelos de ML, podendo prejudicar o desempenho do modelo. Dessa forma, os melhores resultados ocorreram na severidade alta, com um *f1-score* de 75,88%, enquanto a severidade baixa apresentou um *f1-score* de 73,41%. Já Ganeshan, Sharif e Apeagyei (2023), utilizaram o sucessor da YOLOv5, o YOLOv8, no reconhecimento de diversos tipos de trincas (longitudinal, transversal, em bloco e por fadiga) e seus três níveis de severidade, obtendo um *f1-score* de 55,01%. Contudo, esses autores não especificaram como os níveis de severidade foram obtidos, uma vez que foi utilizado um banco de dados de terceiros.

Dentre os 11 autores que utilizaram bancos de dados DSJ incluídos na revisão de Souza e Fontenele (2025, submetido), apenas Tran *et al.* (2021) e Ibrahim *et al.* (2021) especificaram no artigo que medições de defeitos por meio de avaliações em campo foram efetuadas. Tran *et al.* (2021) empregou a RetinaNet para o reconhecimento de trincas transversais, longitudinais e por fadiga, realizando as medições em campo após o treinamento, para verificar a acurácia do modelo, que foi de 84,90%. Ibrahim *et al.* (2021), por sua vez, realizou as medições em campo para a confecção do banco de dados, utilizando um paquímetro para aferir a largura de trincas. O modelo treinado pelos autores, através do MATLAB, obteve uma acurácia

de 93,27%.

Recentemente, no trabalho de Souza, Cestari e Fontenele (2025) também foram realizadas medições em campo. Neste caso, para a confecção de um banco de dados de buracos e seus três níveis de severidade. Para treinamento, os autores utilizaram a YOLOv8s e YOLOv8m. Ambos os modelos se mostraram funcionais apenas no reconhecimento de buracos com o nível de severidade alta, com resultados imprecisos para as demais severidades. Dessa forma, os autores levantaram a hipótese de que a ausência da informação de profundidade nas imagens utilizadas afetou o desempenho do modelo, uma vez que o modelo empregado reconhece apenas padrões 2D, enquanto a profundidade é uma informação 3D.

O presente trabalho também se encaixa na forma DSJ, porém preenchendo as lacunas percebidas. Dentre elas é possível destacar: a falta de abordagem de todos os três níveis de severidade, onde geralmente a severidade média é ignorada; abordagens que não contemplam múltiplos defeitos simultaneamente ou focam em classes especificas e a ausência da avaliação em campo, tanto para confecção de banco de dados quanto para validação dos modelos treinados, comprometendo a aplicabilidade prática dos algoritmos.

4.2 Forma 2 – Defeitos e Severidades Separados (DSS)

Em relação a forma 2 - DSS, Souza e Fontenele (2025, submetido), constataram que apenas 4 artigos, dentre os 15 que utilizaram DL, empregaram esse método em diferentes tipos de arquiteturas.

Uma *Deep Convolutional Neural Network* (DCNN) foi proposta por Song *et al.* (2019) para a detecção automática de trincas com a ResNet como *backbone* em conjunto com os módulos *Multiscale Dilated Attention* (MDA) e *Feature Fusion Upsampling* (FFU). Além disso, os autores compararam o modelo proposto com outros modelos de segmentação, sendo eles: SegNet, U-Net, *Pyramid Scene Parsing Network* (PSPNet), DeepLab v3+ e *Discriminative Feature Network* (DFN). Os melhores resultados aconteceram na DCNN proposta, com uma precisão de 98,74%, *recall* de 98,05% e *f1-score* de 98,39%. Após o processo de detecção os autores determinaram tanto a classe das trincas (transversal, longitudinal, bloco ou por fadiga) quanto o seu nível de severidade (baixa e alta) tendo como base o número de ramos

do defeito, a sua largura média e distâncias entre cada ramo. Com essa metodologia a acurácia na classificação das trincas transversais foi de 95%, nas longitudinais 96,1%, nas de bloco 87,9% e nas por fadiga 86,2%.

A Mask R-CNN foi empregada por Tran *et al.* (2020) no reconhecimento de trincas longitudinais, transversais e por fadiga. Para as trincas por fadiga, os autores adotaram a Forma 1 – DDS, utilizando a própria Mask R-CNN para reconhecer o nível de severidade desse tipo de trinca. Porém, nas trincas lineares (longitudinais e transversais), os autores utilizaram a conectividade de pixels para verificar a largura média, determinando após o processo de treinamento o seu nível de severidade. O método proposto alcançou uma acurácia de 92% para a identificação do tipo de trincas e 90.15% na identificação e reconhecimento do nível de severidade.

Ranyal, Sadhu e Jain (2023) utilizaram dois bancos de dados públicos de buracos para o treinamento de uma RetinaNet modificada, que obteve uma precisão e um recall de 99% no reconhecimento do defeito em questão. Para determinação do nível de severidade, os autores aplicaram o processo fotogramétrico de *Structure from Motion* (SfM) para obtenção de informações geométricas 3D, a partir de imagens 2D. Através desse procedimento, foi obtido uma nuvem de pontos que foi empregada em um algoritmo para estimar a profundidade dos buracos e em seguida determinar o seu nível de severidade. Os autores contataram que a estimativa de profundidade através do algoritmo proposto apresenta um erro médio de 5% em relação às medidas obtidas em campo.

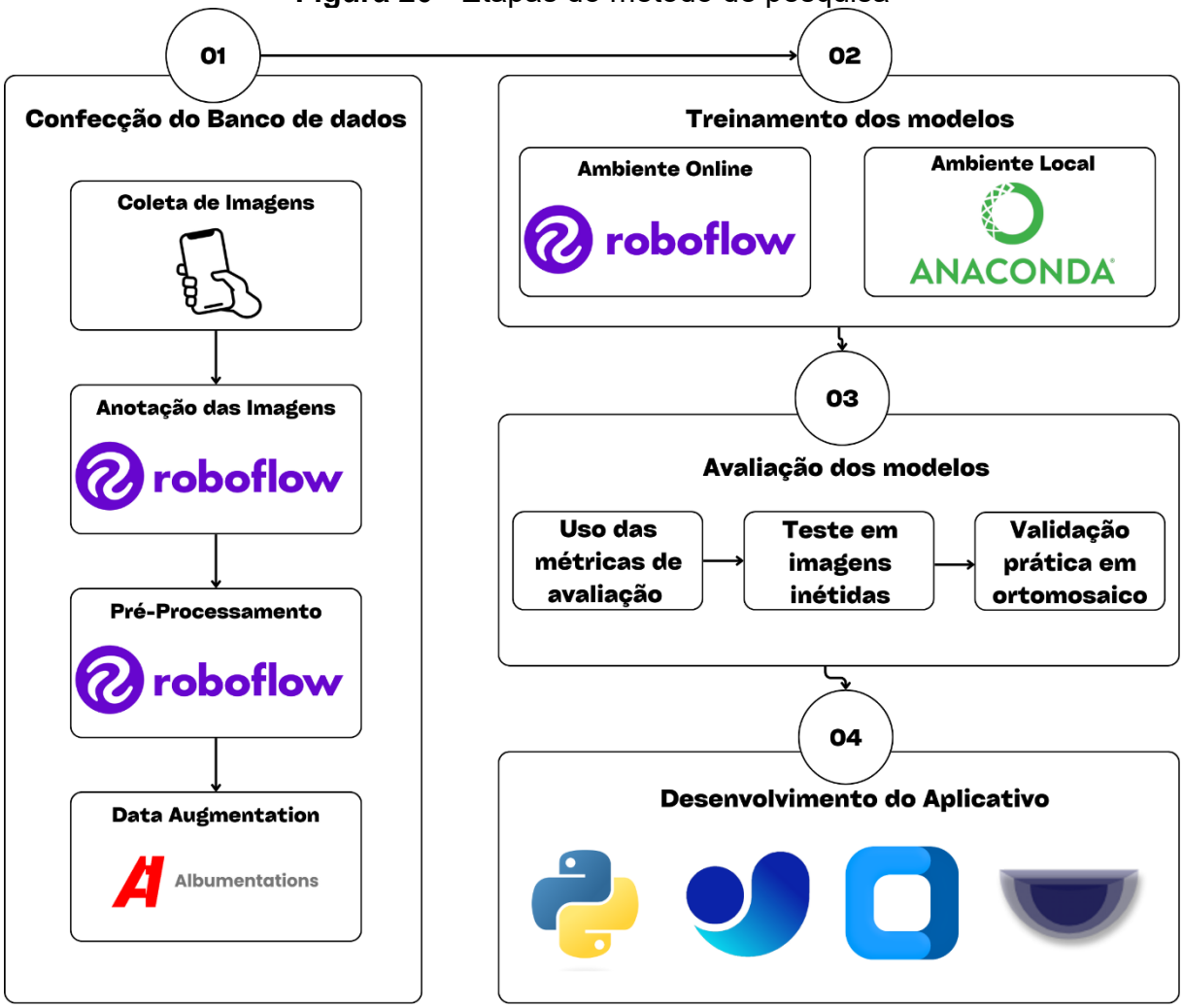
Em outro trabalho, Ranyal, Sadhu e Jain (2024) focaram no reconhecimento de trincas por longitudinais e transversais em pavimentos de concreto, através do treinamento da YOLOv5, SSD, RetinaNet, onde essa última, em conjunto com módulos de atenção obteve os melhores resultados, alcançando um *f1-score* de 90%. Já para a determinação do nível de severidade das trincas, o autor utilizou: extração das regiões de interesse, processamento digital de imagens, detecção *Canny Edge*, extração e conectividade de pixels e estimativa de largura de forma algébrica. Dessa forma, os autores obtiveram um erro relativo médio de 6% em relação a medições manuais da largura das trincas.

Apesar desta forma de reconhecimento possuir uma quantidade menor de artigos, é notável que ela apresenta um bom desempenho. Dessa forma, a DSS necessita ser explorada por mais estudos.

5 MÉTODO DE PESQUISA

Para realização do presente trabalho o método de pesquisa foi dividido em 4 etapas, conforme ilustrado na Figura 20, sendo elas: Confecção do banco de dados (coleta de imagens, anotação das imagens, pré-processamento e *data augmentation*), Treinamento dos Modelos (no ambiente *online* e no ambiente local), Avaliação dos modelos (Uso das métricas de avaliação, teste em imagens inéditas e validação prática em ortomosaico) e desenvolvimento do aplicativo Detector de Defeitos em Pavimentos (DDP).

Figura 20 - Etapas do método de pesquisa



Fonte: Autoria Própria (2025)

Com o intuito de permitir a reprodutibilidade do presente trabalho um *framework* ilustrado foi elaborado com base nas principais etapas do método empregado na pesquisa. O framework pode ser observado no APÊNDICE A.

5.1 CONFECÇÃO DOS BANCO DE DADOS

A primeira etapa consistiu na confecção do banco de dados, procedimento fundamental para o treinamento de modelos de ML. Para tal, primeiramente foi realizada a coleta de imagens de defeitos presentes na superfície de pavimentos em áreas urbanas. As imagens foram obtidas através de um *smartphone* Galaxy A13, que possui uma câmera de 50 *megapixels.* É válido mencionar que a coleta foi realizada em dias ensolarados (visando eliminar a presença de sombras nas imagens) e no turno matutino (com o intuito de não interromper o trânsito). Além disso, as vias escolhidas para coleta estavam localizadas em bairros residenciais que possuem uma baixa incidência de tráfego.

Foram coletadas imagens de defeitos com maior incidência em áreas urbanas, sendo eles: buracos, remendos e trincas por fadiga. Ressalta-se, que além de fotografar os defeitos, também foi determinado o nível de severidade em campo, em conformidade com os procedimentos estabelecidos na norma técnica internacional ASTM D6433–24. Na Figura 21 é possível observar parte do processo de medição para a determinação do nível de severidade do defeito buraco.

Figura 21 – Processo de medição do defeito buraco

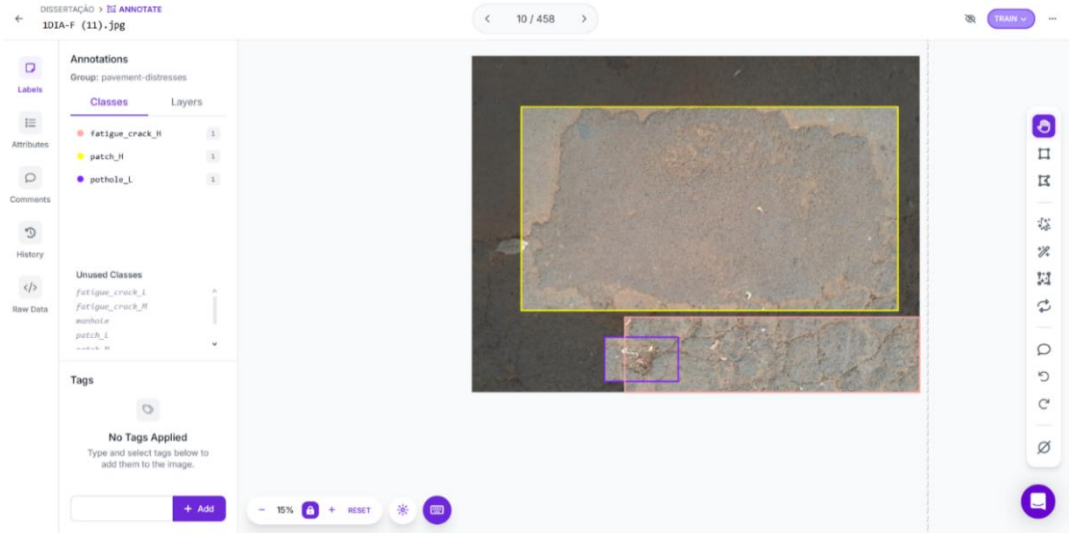
Fonte: Autoria Própria (2025)

Também foram coletadas imagens de Tampas de Poços de Visitas (TPVs), popularmente conhecidos como “bueiros”. Esse elemento, presente de forma abundante em áreas urbanas, foi levado em consideração pois, devido a seu formato similar a buracos, pode induzir os modelos de ML a predições incorretas. Além disso,

Loprencipe e Pantuso (2017), destacam que as TPVs podem causar danos e acidentes quando se encontram irregulares, soltas, ligeiramente abertas ou mal posicionadas.

Dessa forma, inicialmente foram obtidas 581 imagens. Todas essas imagens passaram por um procedimento de anotação. Esse procedimento de anotação consistiu em marcar com retângulos, chamados de *bounding boxes*, os defeitos presentes na imagem. Foi por meio dessas anotações que o modelo de ML, na etapa de treinamento, conseguiu identificar qual elemento da imagem ele iria “aprender”. As anotações foram efetuadas através da plataforma Roboflow, cuja interface pode ser observada na Figura 22.

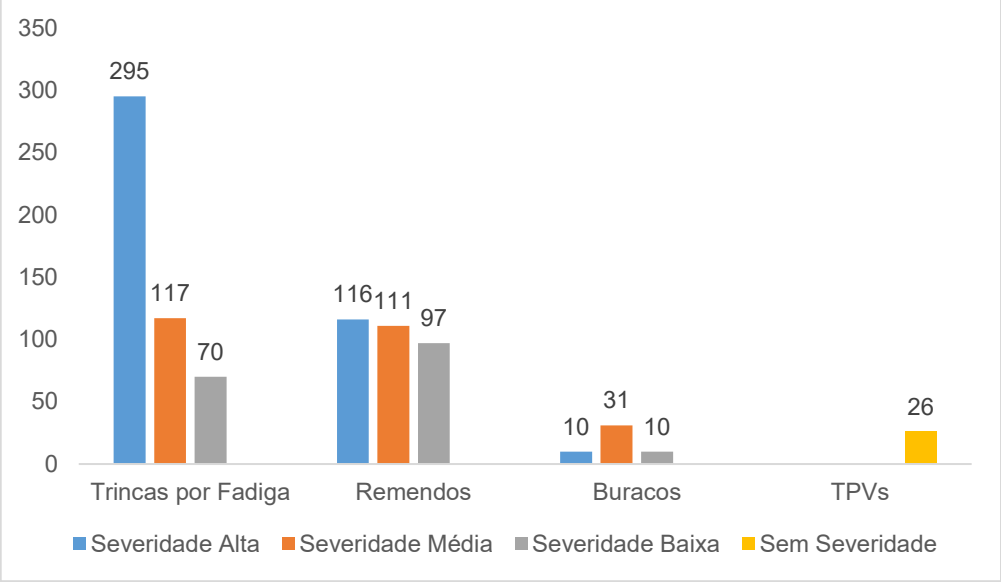
Figura 22 - Interface de anotação do Roboflow



Fonte: Autoria Própria (2025)

Em sequência verificou-se que as 581 imagens apresentavam 893 anotações. Um número maior de anotações ocorreu devido a algumas das imagens conterem mais de um tipo de defeito e/ou nível de severidade. No Gráfico 1, é possível observar a distribuição das severidades para cada tipo de defeito.

Gráfico 1 - Distribuição das severidades por tipo de defeito



Fonte: Autoria Própria (2025)

Devido à pequena quantidade de anotações de buracos e TPVs em relação aos demais defeitos, foram utilizados bancos de dados de terceiros. Dessa forma, utilizou-se o banco de dados de Ferrari *et al.* (2023), com 12 imagens de buracos (10 de severidade alta e 2 de severidade média) e o banco de dados de Souza, Cestari; e Fontenele (2025) com 60 imagens de buracos (20 de severidade alta, 20 de severidade média e 20 de severidade baixa). Ressalta-se que nesses dois trabalhos, os autores utilizaram o MID para determinação do nível de severidade do defeito buraco. Em relação as TPVs, novas imagens foram retiradas de bancos de dados públicos presentes na plataforma Roboflow Universe, na qual utilizou-se 157 imagens do SWUnionProject (https://universe.roboflow.com/swunionproject/manhole-c1n3x) e 61 imagens do Workstation (https://universe.roboflow.com/workstation-gegoz/manhole-detection-xwwqe).

Após a incorporação desses 4 bancos de dados de terceiros, 100 imagens (10 de cada tipo de defeito e severidade e 10 de TPVs) foram retiradas do banco de dados confeccionado. Essas imagens serão utilizadas posteriormente na etapa de teste para verificar a eficácia dos modelos empregados.

As demais imagens passaram por um pré-processamento (no próprio Roboflow), onde realizou-se duas operações: auto orientação e redimensionamento. A auto orientação, conforme consta em Dwyer (2020), foi realizada para padronizar a ordem dos pixels presentes nas imagens e evitar bugs e erros. O redimensionamento,

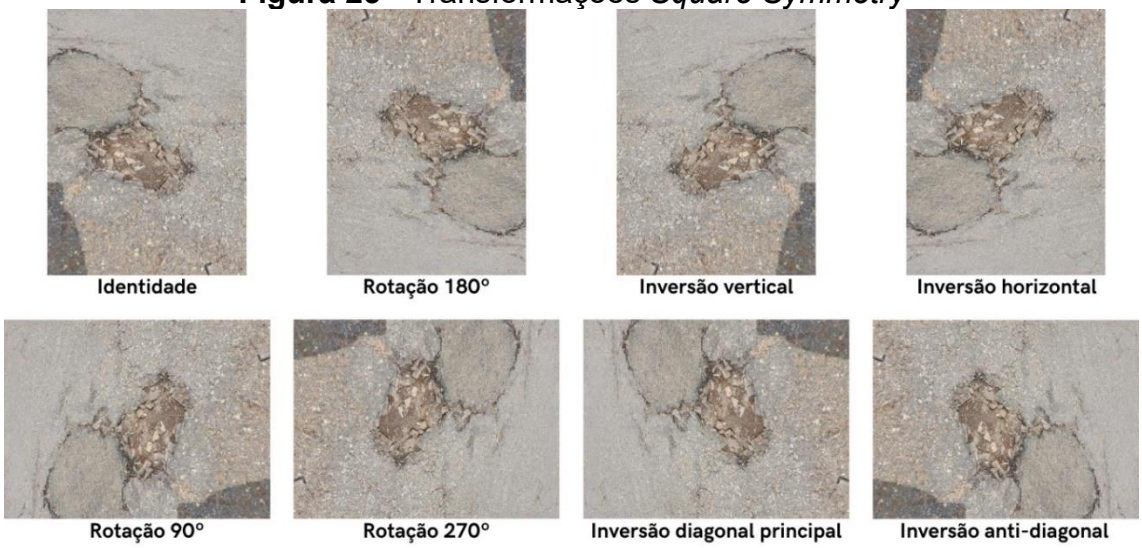
de acordo com o que consta em Nelson (2020), foi realizado com o intuito de reduzir a dimensão das imagens para obter um tempo de treinamento menor. Dessa forma as imagens obtidas com o *smartphone* foram redimensionadas de 3060x4080 (verticais) e 4080x3060 (horizontal) para 480x640 e 640x480, respectivamente.

Após o pré-processamento, o banco de dados foi divido em imagens de treinamento e imagens de validação. A proporção escolhida para a divisão foi de 70/30, ou seja, 70% das imagens foram destinadas para treinamento e 30% para validação.

Em seguida, foi empregada a técnica de *data augmentation*, visando aumentar o tamanho e a diversidade de bancos de dados anotados, conforme mencionado por Buslaev *et al.* (2020). Para implementar o *data augmentation* foi utilizada a biblioteca de Python Albumentations, através do ambiente de desenvolvimento integrado PyCharm.

No Albumentations foram escolhidas 3 transformações: *SquareSymmetry*, *Scale* e *Rotate*. É válido mencionar que a *Square Symmetry* utiliza oito possíveis transformações de simetria quadrada, que podem ser visualizadas na Figura 23.

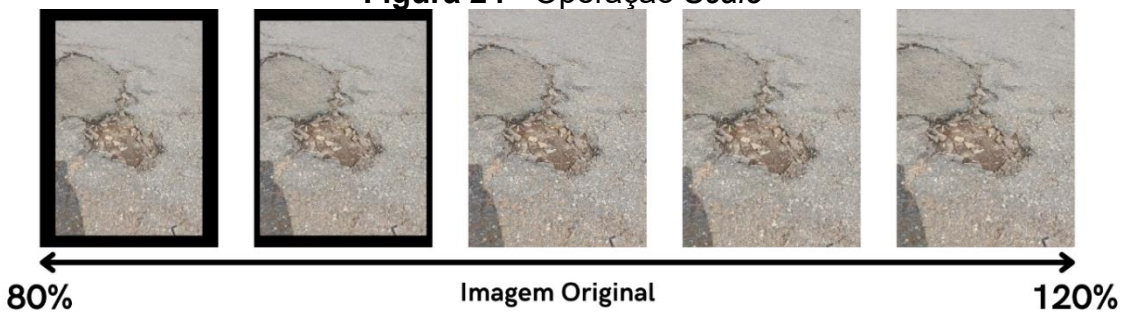
Figura 23 - Transformações *Square Symmetry*



Fonte: Autoria Própria (2025)

A operação *Scale*, por sua vez, aplica *zoom* na imagem, podendo ser *zoom in* ou *zoom out*. Os valores escolhidos para variação de zoom foram entre 80% e 120%, conforme pode ser observado na Figura 24.

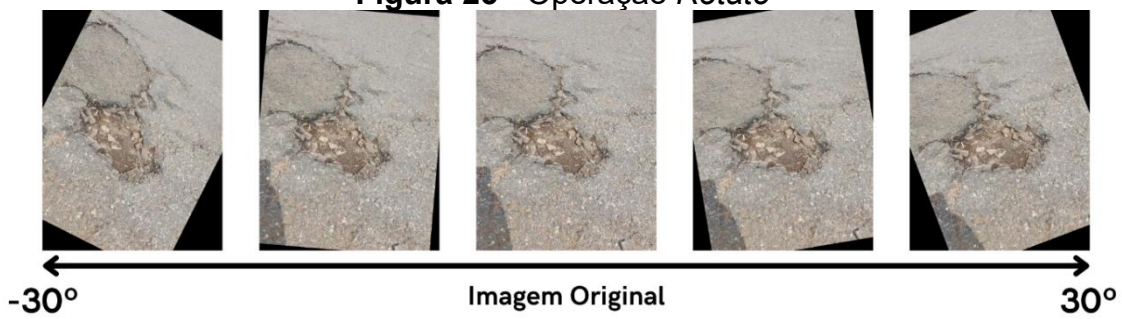
Figura 24 - Operação *Scale*



Fonte: Autoria Própria (2025)

Por fim, a operação *Rotate* foi empregada para simular ligeiras inclinações de câmera, onde os valores escolhidos foram inclinações entre -30º e 30º. A variação de inclinação pode ser visualizada na Figura 25.

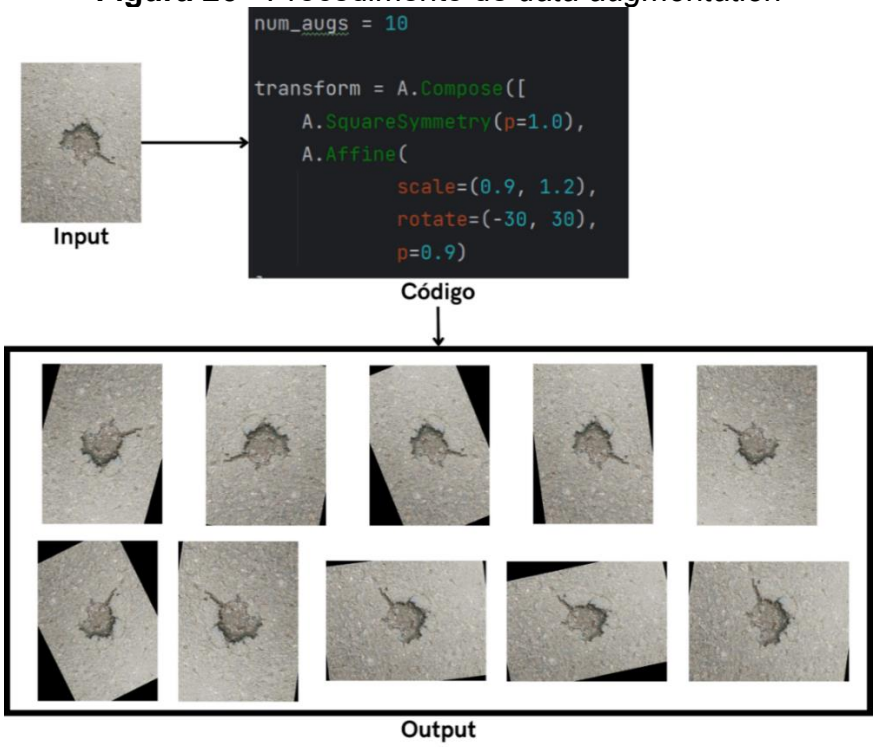
Figura 25 - Operação *Rotate*



Fonte: Autoria Própria (2025)

Durante o processo de expansão foi levada em consideração o balanceamento do banco de dados, tentando manter de forma mais equilibrada possível a quantidade de imagens para cada tipo de defeito e severidade. Dessa forma, o defeito buraco foi o que mais sofreu expansões, onde as imagens com severidade baixa e alta foram aumentadas em dez vezes, ou seja, cada imagem original foi transformada em dez imagens distintas por meio das técnicas empregadas, assim como demonstrado na Figura 26. Vale ressaltar que durante o *data augmentation* todas as três operações apresentadas (*SquareSymmetry*, *Scale* e *Rotate)* são aplicadas simultaneamente na imagem de forma aleatória.

Figura 26 - Procedimento de *data augmentation*



Fonte: Autoria Própria (2025)

Dentre os demais defeitos incluídos no *data augmentation* estão os buracos com severidade média (aumentados em 7 vezes), as trincas por fadiga de severidade baixa, remendo de severidade baixa e alta (aumentados em 5 vezes) e remendos de severidade média e trincas por fadiga de severidade (aumentados em 3 vezes).

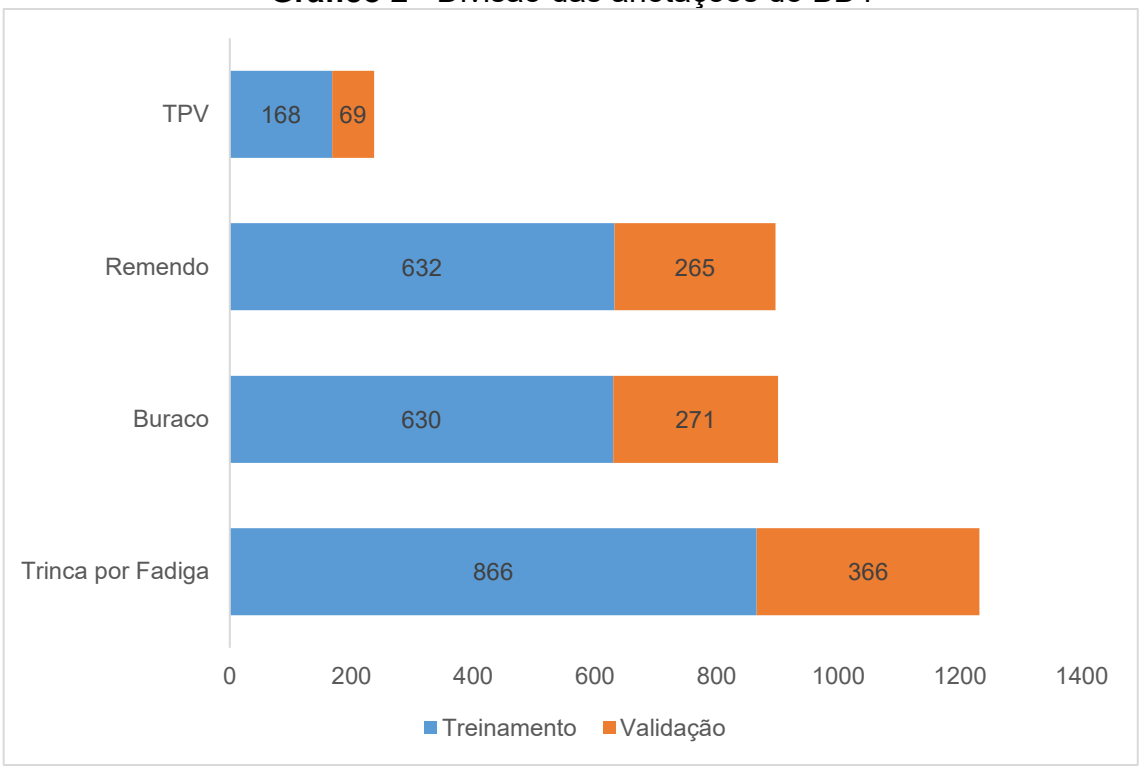
Evitou-se ao máximo expandir imagens que continham trincas por fadiga de severidade alta (devido à grande quantidade de anotações desse defeito antes mesmo da expansão, vide Gráfico 1). Contudo, isso foi inevitável, pois esse defeito estava presente em diversas imagens de buracos (uma vez que um dos motivos para o surgimento de buracos são as trincas por fadiga). É válido mencionar que as TPVs não foram submetidas ao procedimento de expansão.

Dessa forma, após o *data augmentation*, o banco de dados passou a conter 2315 imagens (1615 para treinamento e 700 para validação) e 3267 anotações. Esse banco de dados foi utilizado de duas formas distintas, a primeira forma, chamada de Banco de Dados 1 (BD1) levou em consideração apenas os defeitos, sem distinção entre níveis de severidade, ou seja, apenas com 4 classes: trinca por fadiga, buraco, remendo e TPVs). Já na segunda forma, intitulada de Banco de Dados (BD2), foram considerados os defeitos e suas respectivas severidade, totalizando 10 classes:

trincas por fadiga Severidade Baixa (SB), trincas por fadiga Severidade Média (SM), trincas por fadiga Severidade Alta (SA), remendo SB, remendo SM, remendo SA, buraco SB, buraco SM, buraco SA e TPVs.

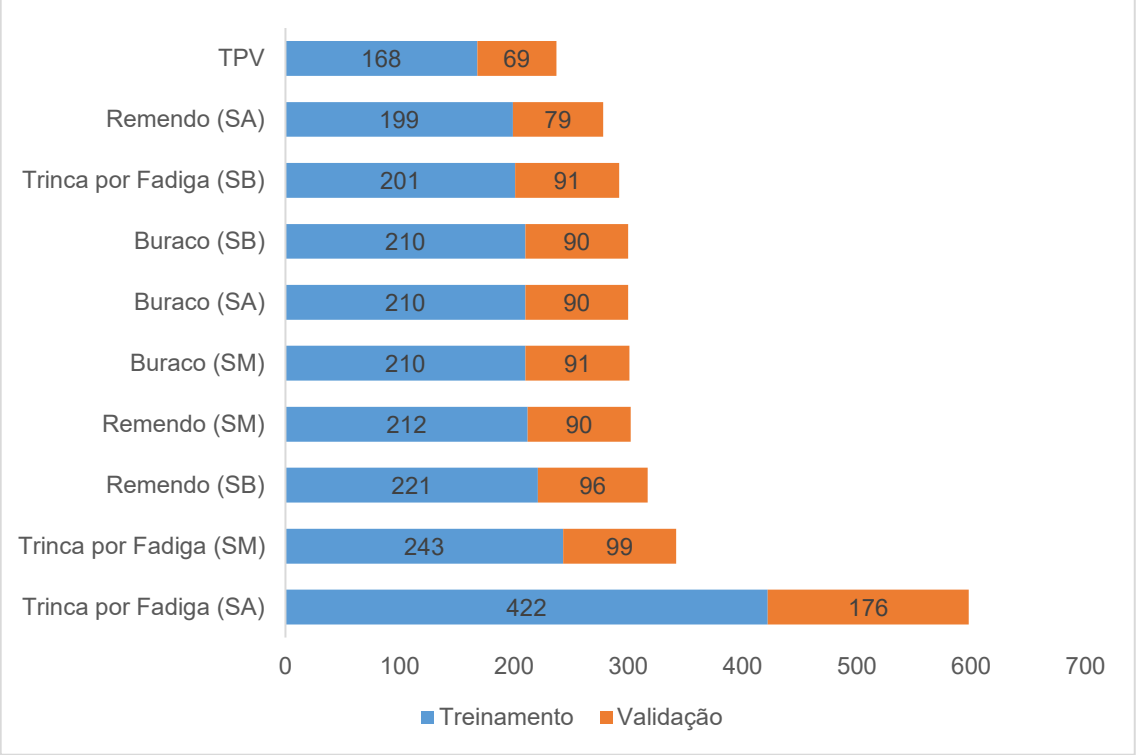
A divisão entre treinamento e validação para as anotações de cada uma das classes do BD1 e BD2 estão expostas no Gráfico 2 e no Gráfico 3, respectivamente.

Gráfico 2 - Divisão das anotações do BD1



Fonte: Autoria Própria (2025)

Gráfico 3 - Divisão das anotações do BD2



Fonte: Autoria Própria (2025)

Ressalta-se que os dois bancos de dados utilizados neste trabalho estão disponíveis publicamente de forma gratuita na plataforma Roboflow Universe. O BD1 pode ser acessado através do link: https://universe.roboflow.com/atila/dissertacao-atila-bd1 e o BD2 através do link: https://universe.roboflow.com/atila/dissertacao-atila-bd2.

5.2 TREINAMENTO DOS MODELOS

Na etapa de treinamento foram utilizados 9 arquiteturas, sendo elas: YOLOv8 (small, *medium*), YOLOv11 (*nano, small, medium*), YOLOv12 (*nano*, *small*), YOLO-NAS (*small*), RF-DETR (Base). Totalizando dessa forma 18 treinamentos (9 para cada banco de dados).

A escolha das arquiteturas levou em consideração o estado-da-arte da detecção de objetos. Para tal selecionou-se as versões mais recentes do YOLO (*i.e* YOLOv8, YOLOv11, YOLOv12 e YOLO-NAS), que combinam desempenho elevado com eficiência de treinamento, e a RF-DETR, que utiliza uma arquitetura baseada em *transformers encoder-decoder*, apresentando uma performance competitiva.

Os treinamentos foram efetuados em dois ambientes: online e local. Como

ambiente online foi utilizando a plataforma Roboflow, que através de um acesso estudantil, permitiu que os algoritmos de DL fossem treinados de forma gratuita (com algumas limitações que serão abordadas no Capítulo 6). No ambiente local, foi utilizado o ambiente de desenvolvimento Anaconda, a linguagem de programação Python e as bibliotecas Ultralytics e PyTorch, em um notebook com uma *Graphics Processing Unit* (GPU) NVIDIA GeForce RTX 4050, processador Intel Core i7 13th geração e 32 GB de memória RAM.

Os modelos YOLOv11n, YOLOv12n, YOLO-NAS-S e RF-DETR-Base foram treinados através do ambiente *online* Roboflow e os modelos YOLOv8s, YOLOv8m, YOLOv11s, YOLOv11m e YOLOv12s através do ambiente local Anaconda.

É válido ressaltar que todos os modelos utilizados foram treinados utilizando a técnica de transferência de aprendizagem (*transfer learning*). Esse procedimento consiste em importar pesos (ou *checkpoints*) de outros modelos já treinados anteriormente. A principal vantagem da transferência de aprendizagem é a redução do tempo de treinamento, uma vez que os modelos pré-treinados já possuem conhecimentos prévio das informações mais básicas de uma imagem (*e.g*. bordas, formas e texturas). Dessa forma, a transferência de aprendizagem foi empregada da seguinte forma:

- **YOLOv8**: Para os modelos YOLOv8s e YOLOv8m, a abordagem utilizada foi a adaptação de domínio (*domain adaptation*), na qual os modelos treinados por Souza, Cestari e Fontenele (2025) para o reconhecimento de buracos e seus níveis de severidade foram importados para o reconhecimento de novos tipos de defeitos (*i.e*. trincas de fadiga, remendos e TPV). Essa abordagem tende a acelerar a convergência de treinamento nos novos objetos considerados.
- **RF-DETR:** No RF-DETR-Base o treinamento foi inicializado a partir de pesos pré-treinados no banco de dados Objects365. Esse banco de dados foi desenvolvido por Shao *et al.* (2019) e consiste em 3 milhões de imagens de 365 objetos distintos. Essa ampla diversidade de objetos contribui para uma maior capacidade de generalização do modelo.
- **Demais modelos:** Os demais modelos YOLO empregados (*i.e.* YOLO-NAS, YOLOv11 e YOLOv12) foram treinados a partir de pesos pré-treinados no COCO. O COCO é um banco de dados amplamente utilizado, que foi elaborado por Lin *et al.* (2014) e contém 330 mil

imagens de 80 categorias de objetos. De forma similar ao Objects365, a utilização de pesos pré-treinados no COCO potencializa a capacidade de generalização dos modelos.

Ressalta-se também que o número de épocas foi definido para 300. Os demais hiper parâmetros não foram modificados, sendo utilizado a configuração de treinamento base para todos os algoritmos YOLO. Porém, é importante enfatizar que apesar do treinamento ter sido configurado para 300 épocas, as arquiteturas YOLO recentes possuem a função *EarlyStopping.* Essa função encerra o treinamento antes do previsto caso o algoritmo perceba que não ocorreram melhorias nas últimas 100 épocas. É possível desativar essa função, porém optou-se por mantê-la visando a otimização do tempo de treinamento.

Além disso, destaca-se que todos os resultados considerados no capítulo seguinte são relativos à Melhor Época (ME) dos modelos. A ME é a época em que o modelo obteve as melhores métricas de avaliação. Ou seja, se um modelo treinou até a época 300, e a ME foi a 200, este trabalho considerou os resultados da época 200, e não da 300.

Por fim, é válido mencionar, que apesar de ser utilizada a versão *medium* do YOLOv8 e YOLOv11, optou-se por não treinar o YOLOv12m. Inicialmente essa versão tinha sido levada em consideração, contudo, no *setup* utilizado, o tempo para treinar uma única época corresponde a cerca de 40min, totalizando assim, 210 horas para um treinamento de 300 épocas. Esse tempo estimado, demasiadamente demorado, inviabilizou a utilização dessa versão. O longo tempo de treinamento também inviabilizou o treinamento local dos modelos DETR fornecidos pela Ultrlaytics.

5.3 Avaliação dos Modelos Treinados

Com o intuito de verificar o desempenho dos modelos treinados foram utilizadas as métricas de avaliação precisão, *recall*, *f1-score* e mAP. Ressalta-se que os modelos YOLO mais recentes retornam dois valores de mAP: o mAP50, que representa o desempenho do modelo no limiar de confiança de 50% e o mAP50-95, que mensura o desempenho do modelo ao longo de diversos limiares (variando de 50% até 95%).

Além da análise das métricas de avaliação, o tempo de treinamento de cada modelo também foi analisado, visando elencar o modelo que equilibra melhor o tempo

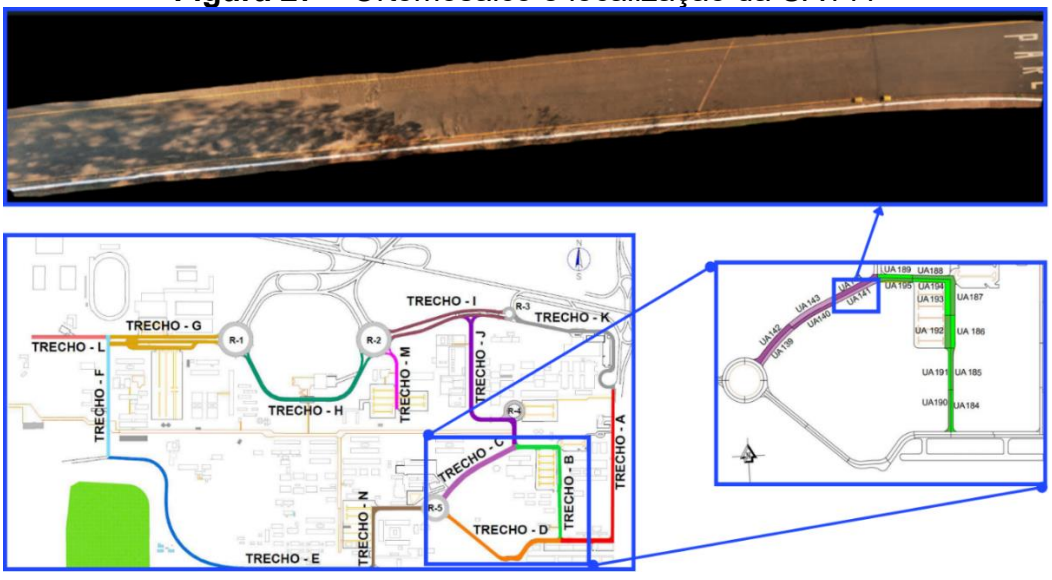
de treinamento com bons resultados. As vantagens e desvantagens de cada ambiente de treinamento também foi considerada.

Em sequência foi efetuada a etapa de teste. Essa etapa foi realizada em 100 imagens que não foram utilizadas durante o processo de treinamento. Para o BD1 as 100 imagens foram divididas em: 30 imagens para cada tipo de defeito e 10 para TPV. Já para o BD2, as 100 imagens foram divididas em 10 imagens por classe. Ressalta-se que cada imagem possui apenas um defeito, com exceção do remendo, onde das 30 imagens tem 37 remendos (sendo 11 de severidade baixa, 12 de severidade alta e 14 de severidade média).

Dessa forma, foi possível observar o comportamento de todos os modelos treinados em um conjunto de imagens inéditas bem como, comparar a severidade obtida em campo com a severidade predita pelos modelos. Após essa etapa, foi possível determinar o melhor modelo, dentre todos os 9 treinados. O modelo com os melhores resultados foi implementado no aplicativo proposto e na validação prática.

A validação prática consistiu na etapa final da avaliação dos modelos, na qual foi utilizado um dos ortomosaicos elaborados por Garcia (2025). A autora elaborou 47 ortomosaicos no total, cada um abrangendo uma Unidade Amostral (UA) da malha viária urbana da UEL. O ortomosaico selecionado para a etapa de validação prática corresponde a UA141 localizada no trecho C do campus (Figura 27). Esta UA foi escolhida pois além de ter sido avaliada com a classificação Péssima (de acordo com o PCI) ela apresenta os 3 defeitos levados em consideração neste trabalho.

Figura 27 – Ortomosaico e localização da UA141



Fonte: Adaptado de Gimenes (2025)

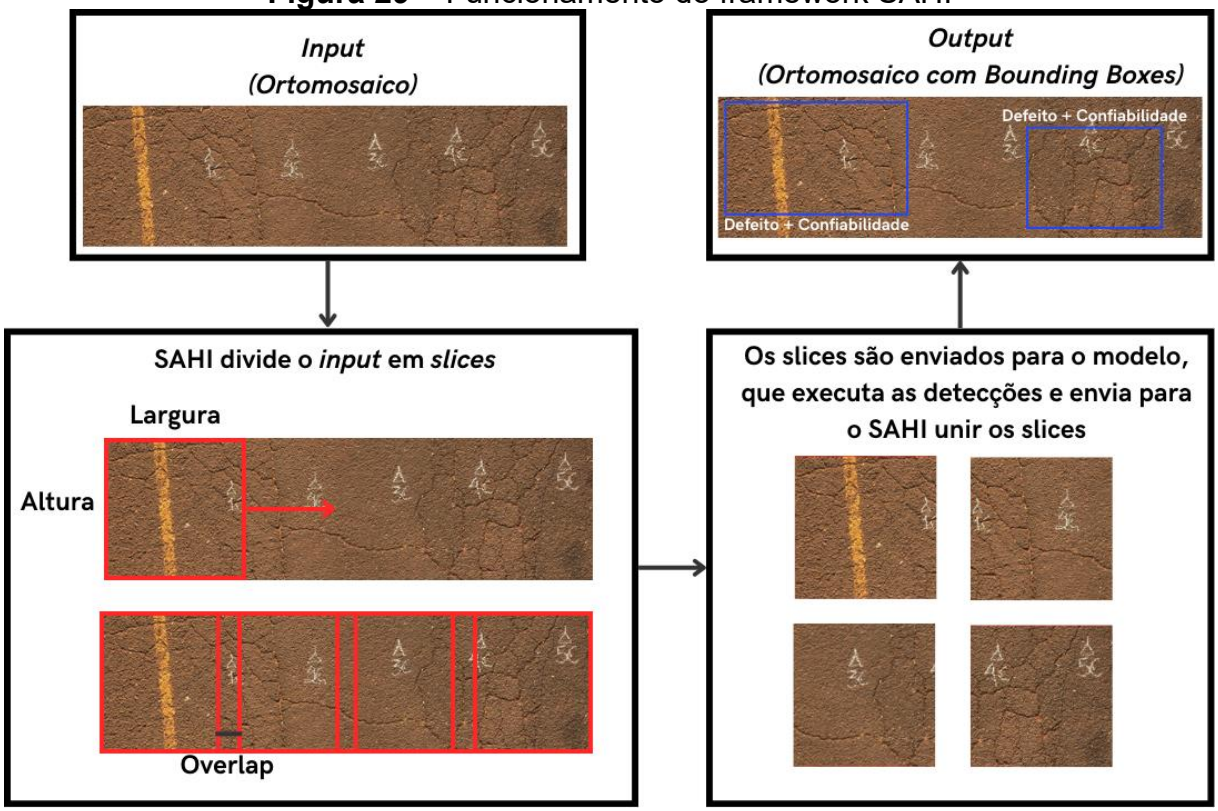
É válido ressaltar que o ortomosaico selecionado foi confeccionado a partir de vídeos utilizando uma GoPro Hero 9 acoplada no Veículo para Procedimento de Aquisição de Imagens do Pavimento e Avaliação (VPAIPA). O VPAIPA, proposto por Garcia (2025), consiste em um veículo de três rodas e estrutura de madeira, composto de três hastes, uma haste vertical (haste A) com dois metros de altura em relação ao solo, uma haste horizontal (haste B) de 1,5m e um terceira haste (haste C) com a função de acoplar a câmera. Na Figura 28 é possível observar a estrutura do VPAIPA.

Figura 28 – Estrutura do VPAIPA

Fonte: Garcia (2025)

Dada a grande dimensão do ortomosaico selecionado (9543x54008), foi necessário a implementação do *Slicing Aided Hyper Inference* (SAHI) para a validação prática do melhor modelo identificado na etapa de teste. O SAHI é um *framework* proposto por Akyon, Altinuc e Temizel (2022) para o reconhecimento de objetos pequenos. Através dele, foi possível dividir os ortomosaicos em *slices* (fatias), possibilitando o reconhecimento dos defeitos presentes no pavimentos. Além dos *slices,* também é possível variar o valor de sobreposição (*overlap*) e confiabilidade (*threshold*). Na Figura 29, é possível observar a aplicação prática do SAHI no ortomosaico.

Figura 29 – Funcionamento do framework SAHI



Fonte: Autoria Própria (2025)

5.4 Desenvolvimento da Versão Beta do Aplicativo

A etapa 4 consistiu no desenvolvimento de um aplicativo para *desktop*, chamado de Detector de Defeitos em Pavimentos (DDP). O DDP tem como público-alvo gestores públicos, engenheiros e acadêmicos que necessitem avaliar a superfície de pavimentos.

O DDP tem como intuito principal permitir uma integração entre os modelos treinados e uma interface gráfica simples e intuitiva. Essa integração possibilita que os usuários que não têm conhecimento em programação consigam utilizar modelos de detecção de objetos no procedimento de avaliação de pavimentos.

Através do DDP, é possível efetuar o *upload* de imagens de trechos de pavimentos com defeitos. Essas imagens são processadas por modelos YOLO, que reconhecem automaticamente a presença dos defeitos e determina seus níveis de severidade. Posteriormente, após avaliar todo o trecho, o aplicativo gera uma planilha com gráficos exibindo a quantidade de defeitos detectados e seus respectivos níveis de severidade. O framework geral do aplicativo proposto pode ser observado no

Apêndice B.

O DDP foi desenvolvido na linguagem de programação Python por meio do ambiente de desenvolvimento integrado Pycharm. Para confecção do DDP diversas bibliotecas de Python foram empregadas, sendo as principais: CustomTkinter (para confecção da interface gráfica), ultralytics (para importar os modelos YOLO treinados) e openpyxl (para permitir a criação de planilhas no Excel).

6 APRESENTAÇÃO E ANÁLISE DOS RESULTADOS

Neste Capítulo serão abordados e discutidos os resultados da pesquisa relativos ao treinamento e teste dos modelos. O Tópico 6.1, aborda as métricas de avaliação obtidas no processo de treinamento. No Tópico 6.2, o foco é a métrica mAP50 em cada uma das classes (4 classes para o BD1 e 10 classes para o BD2). No Tópico 6.3, é abordado o tempo de treinamento para cada modelo nos dois ambientes utilizados e no Tópico 6.4 são elencadas as vantagens e desvantagens desses ambientes. No Tópico 6.5 é discorrido os resultados da etapa de teste para os dois melhores modelos, bem como a validação prática. Por fim, no Tópico 6.6, o aplicativo desenvolvido é apresentado.

6.1 Análise Geral das Métricas de Avaliação do Treinamento

Neste tópico serão abordadas as métricas de avaliação mAP50, mAP50-95, precisão, *recall* e *f1-score* obtidas nos treinamentos dos dois bancos de dados empregados. Primeiramente a análise será geral, sem levar em consideração cada classe separadamente. Os resultados de cada classe serão discutidos no tópico posterior.

6.1.1 Banco de Dados 1 (BD1)

Para o BD1, as métricas de avaliação dos modelos treinados, bem como o Número de Épocas (NE) e o valor da Melhor Época (ME), podem ser visualizadas na Tabela 2.

Tabela 2 – Métricas de Avaliação para o BD1

Modelo	Ambiente	NE	ME	Métricas de avaliação (%)				
				mAP50	mAP50-95	Precisão	*Recall*	*F1-score*
YOLOv8s	Local	237	137	87,7	61,3	89,1	79,5	84,0
YOLOv8m	Local	237	137	87,6	63,4	**91,4**	79,7	85,1
YOLO-NAS-S	Online	57	22	87,2	55,6	90,5	78,2	83,9
YOLOv11n	Online	194	194	89,6	64,8	89,4	**84,2**	86,7
YOLOv11s	Local	179	79	88,6	63,3	88,9	83,0	85,9
YOLOv11m	Local	295	195	89,7	64,7	89,5	**84,2**	**86,8**
YOLOv12n	Online	96	96	89,1	62,9	90,1	82,3	86,0
YOLOv12s	Local	300	300	89,0	65,1	89,2	83,7	86,4
RF-DETR-Base	Online	46	46	**90,3**	**65,4**	85,2	80,0	82,5

Fonte: Autoria Própria (2025)

Através da Tabela 2, é perceptível que todos os modelos empregados apresentam bons resultados, onde nenhum mAP50 obteve valores menores que 87%. Além disso, verifica-se a superioridade do modelo RF-DETR-Base em relação aos modelos YOLO, alcançando o mAP50 de 90,3% e o mAP50-95 de 65,4% em 46 épocas de treinamento.

Já para as demais métricas, o YOLOv11 se destaca, pois o YOLOv11n obteve o maior *recall* e o YOLOv11m obteve o maior *recall* e *f1-score*. Enquanto para a métrica precisão o melhor resultado, de 91,4% foi alcançado pelo YOLOv8m.

6.1.2 Banco de Dados 2 (BD2)

Para o BD2, as métricas utilizadas, o NE e o valor da ME, podem ser visualizados na Tabela 3.

Tabela 3 - Métricas de Avaliação para o BD2

Modelo	Ambiente	NE	ME	Métricas de avaliação (%)				
				mAP50	mAP50-95	Precisão	*Recall*	*F1-score*
YOLOv8s	Local	202	102	62,7	40,4	62,9	63,9	63,4
YOLOv8m	Local	291	191	63,5	40,2	62,8	65,3	**64,0**
YOLO-NAS-S	Online	101	61	59,3	36,4	65,8	54,8	59,8
YOLOv11n	Online	188	188	62,0	42,5	66,1	60,5	63,2
YOLOv11s	Local	156	56	63,5	41,7	65,3	62,0	63,6
YOLOv11m	Local	191	91	62,6	41,0	**66,4**	59,9	63,0
YOLOv12n	Online	106	106	62,1	42,0	63,6	59,5	61,5
YOLOv12s	Local	142	42	**65,2**	43,4	60,8	**65,5**	63,1
RF-DETR-Base	Online	28	28	64,1	**44,3**	63,9	63,0	63,4

Fonte: Autoria Própria (2025)

Por meio da Tabela 3, é notável que os resultados de todas as métricas foram inferiores quando comparadas ao BD1. Valipour *et al.* (2023) também perceberam isso em seu trabalho. Os autores empregaram dois bancos de dados com imagens de trinca em bloco, um com anotação do defeito e outro com anotação do defeito e severidade. De forma similar ao presente trabalho, o banco de dados contendo o nível de severidade apresentou métricas com valores menores em relação ao banco de dados apenas com o defeito. Conforme Valipour *et al.* (2023) essa diferença ocorre devido ao maior número de características que o modelo tem que extrair para diferenciar o nível de severidade.

Mesmo apresentando resultados inferiores ao BD1, todas as métricas do BD2 são satisfatórias, com nenhum mAP50 apresentando valores abaixo de 59%. Nota-se que o melhor desempenho ocorreu no modelo YOLOv12s, que obteve o melhor mAP50 de 65,2% e o melhor *recall de* 65,5%. Destaca-se também o desempenho do YOLOv11m (maior precisão), RF-DETR (maior mAP50-95) e YOLOv8m (maior *f1-score*). No caso do BD2, verifica-se a vantagem dos modelos YOLO, sob o modelo RF-DETR.

6.2 Análise da Etapa de Treinamento por Classes

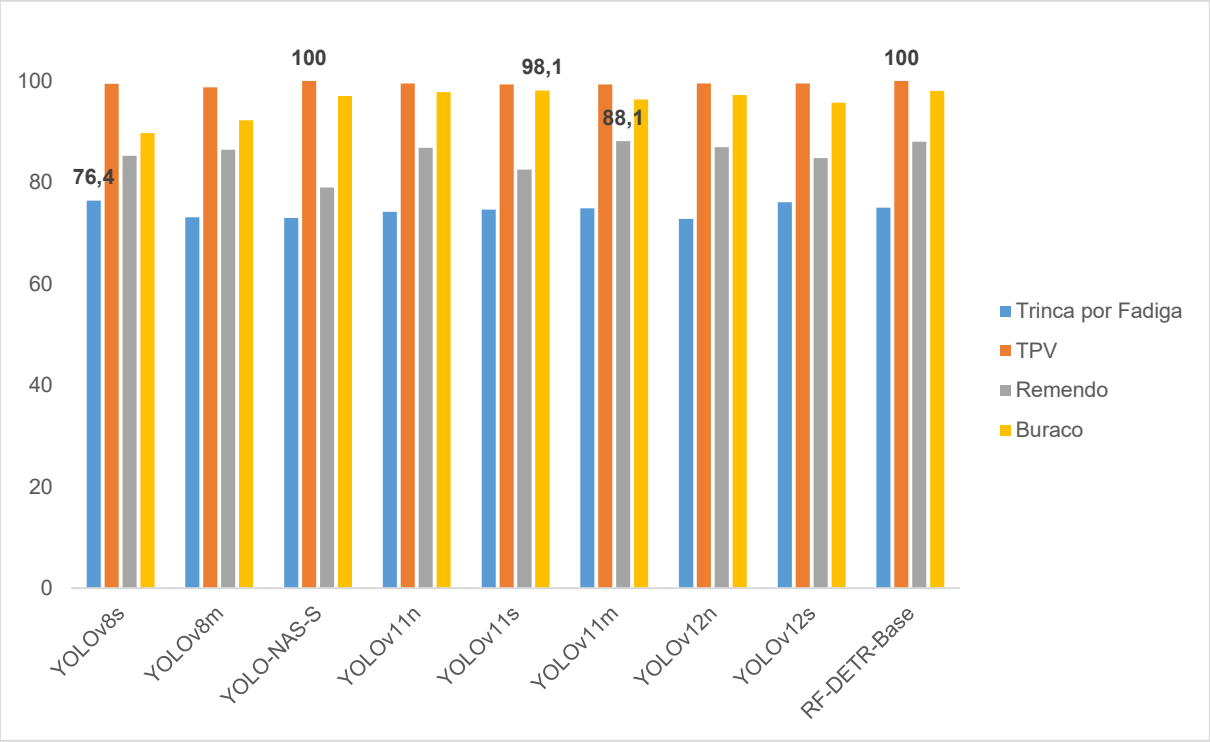
Neste tópico é apresentada a análise dos resultados da etapa de treinamento por classe, verificando a capacidade dos modelos em reconhecer os defeitos e seus

níveis de severidade. Primeiramente será abordado o BD1 (com 4 classes) e em seguida o BD2 (com 10 classes). A métrica de avaliação a ser discutida será o AP50, uma vez que ela agrega tanto a precisão como o *recall*.

6.2.1 Banco de Dados 1 (BD1)

Em relação ao BD1, será abordada a capacidade dos modelos em reconhecer os defeitos, uma vez que esse banco de dados não fornece o nível de severidade. Sendo assim, foi elaborado o Gráfico 4, onde os valores apresentados são relativos ao maior mAP50 de cada classe.

Gráfico 4 - Valores de AP50 (%) dos modelos para as classes do BD1



Fonte: Autoria Própria (2025)

Através do Gráfico 4, é notável que a trinca por fadiga foi o defeito que obteve o menor AP50 em todos os modelos treinados. O YOLOv8s foi o modelo que obteve o maior de AP50, de 76,4%, para essa classe. Vale destacar também o YOLOv12s, que obteve 76,1% para este mesmo defeito, uma diferença de apenas 0,3% em relação ao YOLOv8s.

A dificuldade no reconhecimento de trincas por fadiga pelos modelos pode ser

explicada pela diversidade do banco de dados empregado, uma vez que, no BD1, os 3 níveis de severidade foram incluídos na mesma classe. Dessa forma, o modelo pode não ter identificado padrões claros, principalmente em relação as trincas por fadiga de severidade baixa, que se diferem demasiadamente das trincas por fadiga de severidade média e alta.

Em relação as TPVs, é notável um comportamento excelente de todos os modelos, onde não ocorreu nenhum resultado abaixo de 98,7%. O destaque para essa classe ocorreu no YOLO-NAS-S e no RF-DETR-Base, que alcançaram 100% de AP50. Ressalta-se que esse valor é muito raro de ocorrer, pois indica que o modelo não identificou nenhum FP e nenhum FN durante o processo de treinamento/validação. O valor de 100% nesse caso se justifica pois o Roboflow fornece os resultados de AP50 para teste e validação sem casas decimais, ou seja, o ambiente online “arredonda” os valores (*e.g.* se o AP equivale a 99,8%, o roboflow considera como 100%).

Em relação ao remendo, nota-se também um bom funcionamento, atingindo um AP50 acima de 79% em todos os modelos, onde o maior, de 88,1%, ocorreu no YOLOv11m. O defeito buraco também obteve bons resultados, todos acima de 89,7%, com 98,1% ocorrendo no YOLOv11s.

Apesar da capacidade dos modelos YOLO em reconhecer defeitos em pavimentos já ter sido explorada de forma abundante na literatura, esse tipo de estudo necessita de constante atualização, uma vez que algoritmos melhores vão surgindo com o passar do tempo. Porém, além de avaliar a capacidade de detecção, é de suma importância que os estudos não se limitem apenas a isso. Expandir o escopo, verificando a capacidade dos modelos de avaliar a severidade dos defeitos, é fundamental para proporcionar uma avaliação automática mais completa e eficiente.

Além disso, ressalta-se que é possível unir o DL com outras técnicas, conforme observado por Souza e Fontenele (2025, submetido). Dessa forma, é possível utilizar a detecção de objetos para o reconhecimento do defeito e empregar outra técnica para determinação do nível de severidade. Mais discussões a cerca disso serão realizadas no tópico seguinte.

6.2.2 Banco de Dados 2 (BD2)

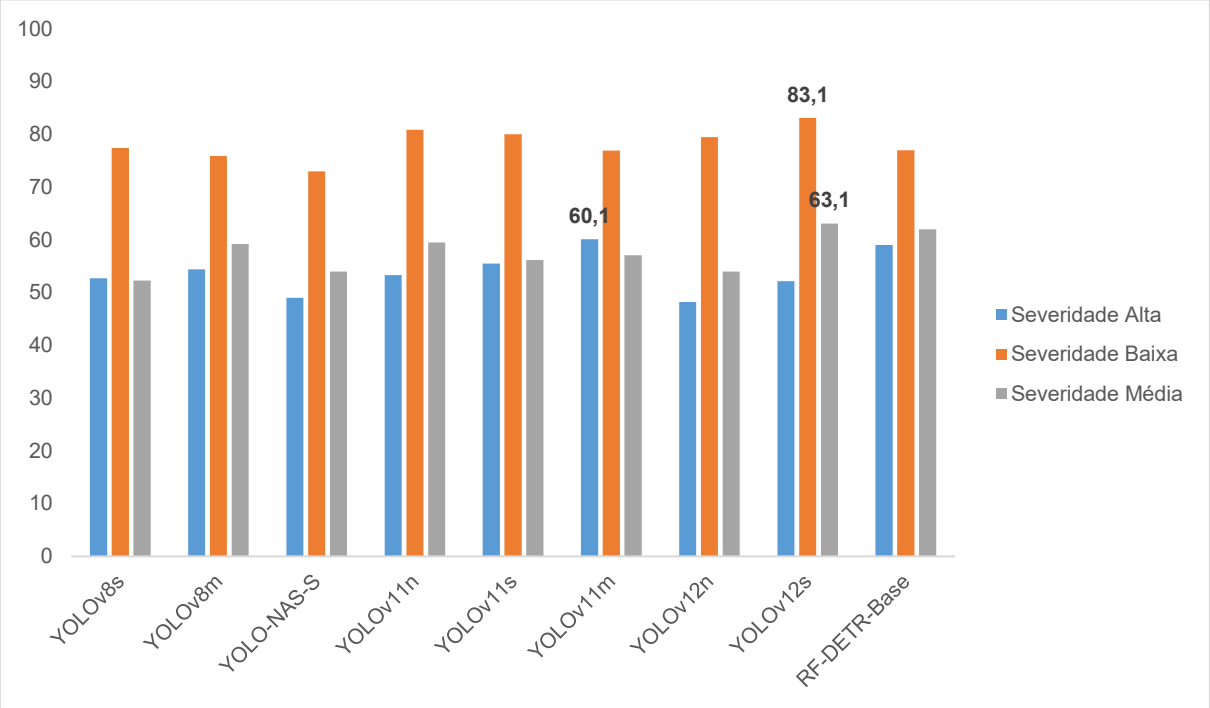
De forma similar ao BD1, a análise conduzida no BD2 também levou em

consideração apenas o AP50.

Devido a quantidade de classes presentes no BD2, os resultados para cada tipo de defeito, e seus níveis de severidade, serão abordados de forma separada. Além disso, os resultados relacionados as TPVs não serão abordados neste tópico, pois os resultados de AP50 foram similares aos ocorrentes no BD1.

Assim, primeiramente, no Gráfico 5 é possível observar os resultados para os três níveis de severidade do defeito trinca por fadiga, onde os valores exibidos são relativos ao maior AP50 de cada classe.

Gráfico 5 - Valores de AP50 (%) para as severidades das trincas por fadiga



Fonte: Autoria Própria (2025)

No Gráfico 5, é perceptível a superioridade do YOLOv12s no reconhecimento de trincas de severidade baixa e severidade média, onde os valores de AP50 obtidos por esse modelo foram respectivamente 83,1% e 63,1%. Essa capacidade de diferenciação entre os níveis de severidade pode ser justificada pelos mecanismos de atenção presentes na arquitetura YOLOv12. A severidade alta, por sua vez, teve seu maior valor, de 60,1%, ocorrendo no YOLOv11m.

No Gráfico 5, também é notável um comportamento similar em todos os modelos: a superioridade do AP50 da severidade baixa em relação às demais. Esse fato pode ser justificado devido às características distintas que a severidade baixa

apresenta (*e.g.* trincas finas, ausência de erosão, poucas ou nenhuma interconexão).

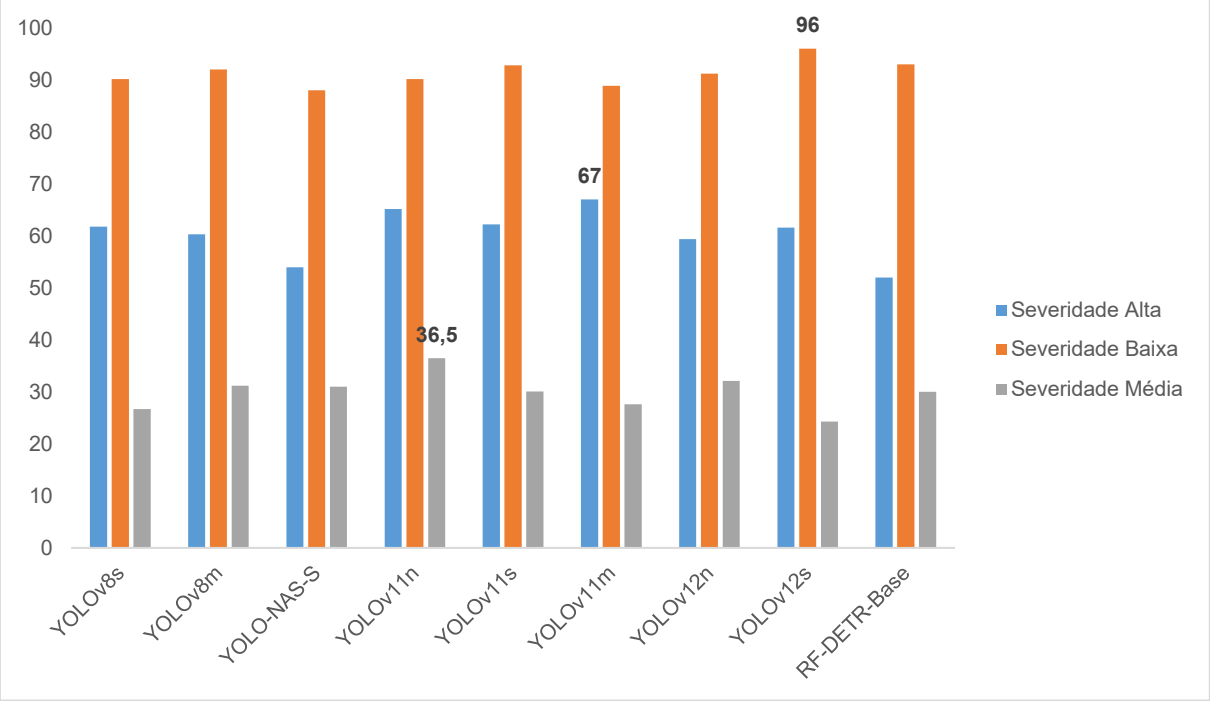
Ressalta-se também que a classe “trincas por fadiga com severidade baixa” do BD2 apresentaram melhores resultados que a classe trincas por fadiga do BD1. Isso fundamenta a hipótese de que os modelos treinados com o BD1 têm dificuldade em reconhecer trincas por fadiga devido à diversidade de imagens no banco de dados.

Já as severidades média e alta apresentam certas similaridades entre si e são difíceis de distinguir em campo. Apesar da ASTM (2024) postular definições diferentes para cada nível de severidade, ela não leva em consideração as propriedades geométricas das trincas por fadiga. Dessa forma, a avaliação em campo tende a ser mais visual e subjetiva, o que pode ocasionar inconsistências no processo de anotação, impactando negativamente o aprendizado do modelo. Trevor *et al.* (2024) e Pardede, Leksmono e Aji (2025) também postulam que a ASTM D6433 possui um certo grau de subjetividade na inspeção visual, onde a classificação de certos defeitos fica a critério dos avaliadores.

Uma forma de contornar esse problema é confeccionar um terceiro banco de dados (BD3), que contenha duas classes: uma classe para as trincas por fadiga severidade baixa e outra com a severidade média e alta unidas. Valipour *et al.* (2023) fizeram algo semelhante em seu banco de dados com imagens de trincas em bloco, pois constataram que existem poucas diferenças entre a severidade média e alta para esse defeito, sendo difícil para o modelo de ML identificar padrões.

Em relação ao remendo, foi elaborado o Gráfico 6, na qual é possível verificar os resultados de AP50, com destaque para os maiores resultados, em cada nível de severidade desse defeito.

Gráfico 6 - Valores de AP50 (%) para as severidades dos remendos



Fonte: Autoria Própria (2025)

Observando o Gráfico 6, constata-se um comportamento similar com a trinca por fadiga, onde a severidade baixa possui uma superioridade considerável em relação aos outros dois níveis de severidade. Porém, no defeito do tipo remendo, a severidade média apresentou resultados com valores bem abaixo dos demais, onde o maior AP corresponde a 36,5% no YOLOv11n. Enquanto para a severidade alta, o melhor resultado ocorreu no YOLOv11m com 67%.

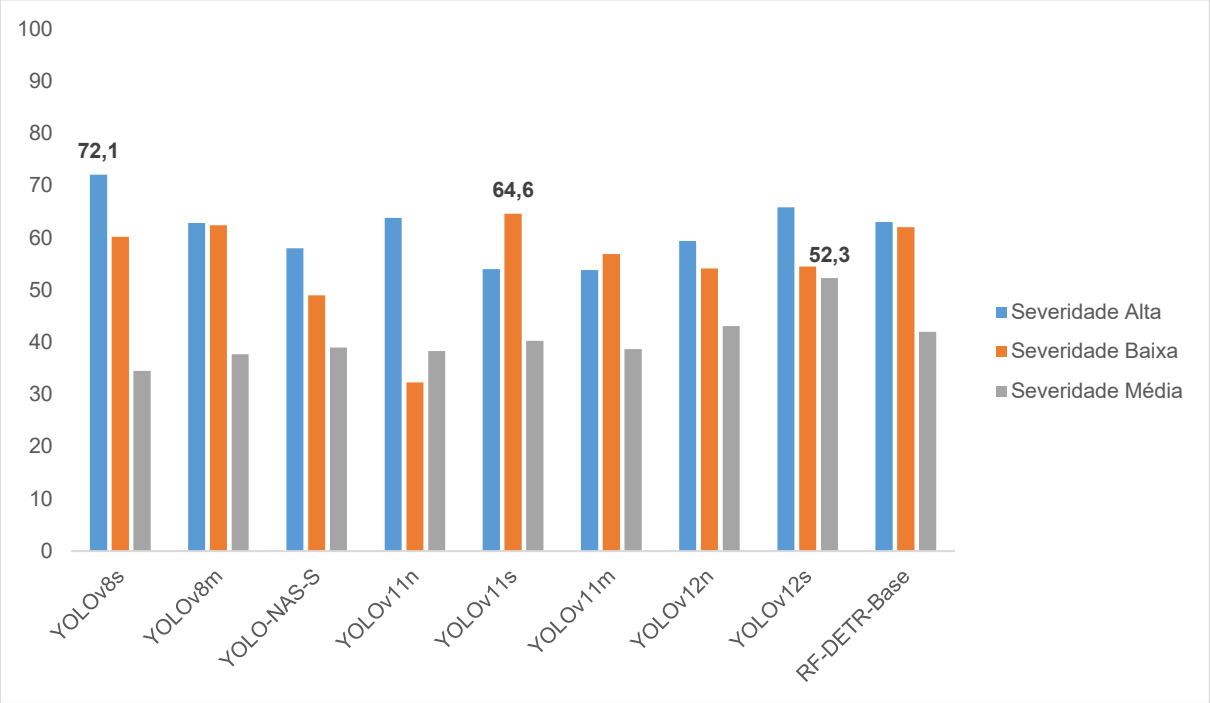
Na severidade baixa, o destaque ocorreu no modelo YOLOv12s que, devido aos seus mecanismos de atenção, obteve 96% de AP50. Assim como ocorreu na trinca por fadiga, o desempenho da classe severidade baixa para o remendo no BD2 apresentou melhores resultados que a classe remendo do BD1.

Além disso, assim como nas trincas por fadiga, foi observado em campo que os remendos de severidade média e alta são facilmente confundíveis, necessitando de uma certa avaliação subjetiva por parte dos avaliadores. Dessa forma, reforça-se a hipótese de que a criação de um BD3 traria melhoria nas métricas de avaliação referentes as severidades média e alta.

Ressalta-se, contudo, que a validação dessa hipótese não foi efetuada no presente trabalho, uma vez que exigiria o retreinamento integral das nove arquiteturas empregadas, excedendo assim, o escopo originalmente proposto.

Por fim, os resultados de AP50 relativos ao defeito buraco e seus 3 níveis de severidade podem ser visualizados no Gráfico 7, na qual os valores exibidos correspondem aos maiores AP50.

Gráfico 7 - Valores de AP50 (%) para as severidades dos buracos



Fonte: Autoria Própria (2025)

Os defeitos do tipo buraco, diferentemente dos dois tipos de defeitos abordados anteriormente, não apresentam um padrão nos resultados. No YOLOv8s, por exemplo, o maior AP50 (72,1%) ocorreu na classe severidade alta, o que pode ser justificado pelo seu pré-treinamento, onde utilizou-se os pesos obtidos por Souza, Cestari; e Fontenele (2025). Já no YOLOv11s a superioridade é na severidade baixa, que obteve 64,6% de AP50. A severidade média por sua vez, foi o nível de severidade com o menor desempenho, tendo o seu maior valor (52,3%) no modelo YOLOv12s.

Esse comportamento, pode ser justificado pelo fato de que o defeito buraco é o único, dentre os três tipos de defeitos abordados, em que a ASTM (2024) considera informações geométricas (área e profundidade) para determinação do nível de severidade. Assim, como a severidade dos defeitos buracos depende diretamente dessas informações e não de características que possam ser identificadas visualmente, os modelos de detecção de objetos podem apresentar resultados inconsistentes. Isso ocorre pois o *input* desse tipo de modelo consiste apenas na

imagem e em nenhum momento do treinamento o modelo tem acesso as medições efetuadas em campo.

Além disso, Souza, Cestari; e Fontenele (2025) ressaltam que os modelos de detecção de objetos reconhecem padrões apenas em duas dimensões, uma vez as imagens utilizadas durante todo o procedimento são 2D. Ou seja, os modelos não conseguem identificar padrões relativos à profundidade, uma informação geométrica da terceira dimensão.

Dessa forma, para o defeito buraco, ao contrário dos outros dois tipos de defeitos, não é necessária a confecção de um BD3, mas sim a investigação de outra forma de determinar o nível de severidade. Diversas técnicas podem ser empregadas para este fim, dentre elas, destacam-se o LiDAR, nos moldes que foi utilizado por Bento *et al.* (2025) , e a fotogrametria, como proposto por Ferrari *et al.*(2023) e por Ranyal, Sadhu e Jain (2023). Além disso, Dong *et al.* (2024) constatam que é possível estimar a área dos buracos, através da segmentação de instâncias, e a profundidade, através de algoritmos de *monocular depth estimation*.

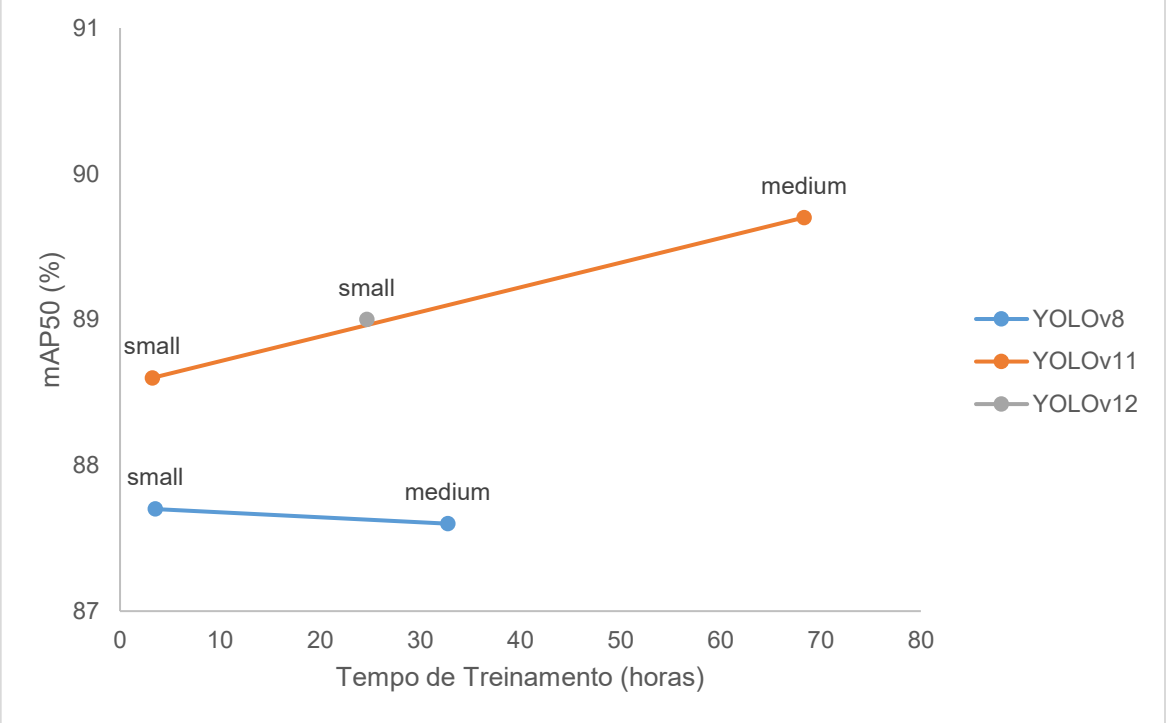
6.3 Tempo de Treinamento e mAP

Analisar o Tempo de Treinamento (TT) é crucial para verificar o gasto computacional dos modelos treinados. Esse fator varia de ambiente para ambiente, pois depende diretamente da GPU utilizada. Ressalta-se que o Roboflow (ambiente online) não especifica qual GPU foi empregada para treinamento. Devido a isso, aspectos mais técnicos em relação a capacidade computacional não foram levados em consideração. Sendo assim, neste tópico será considerada a métrica mAP50 e sua relação com o TT, onde foi verificado o desempenho de cada modelo e banco de dados nos dois ambientes utilizados.

6.3.1 Banco de Dados 1 (BD1)

Em relação ao BD1, é possível observar o Gráfico 8, onde é verificado o mAP50 (%) em relação ao TT (horas) no ambiente local.

Gráfico 8 - Relação entre o TT e mAP50 para o BD1 no ambiente local



Fonte: Autoria Própria (2025)

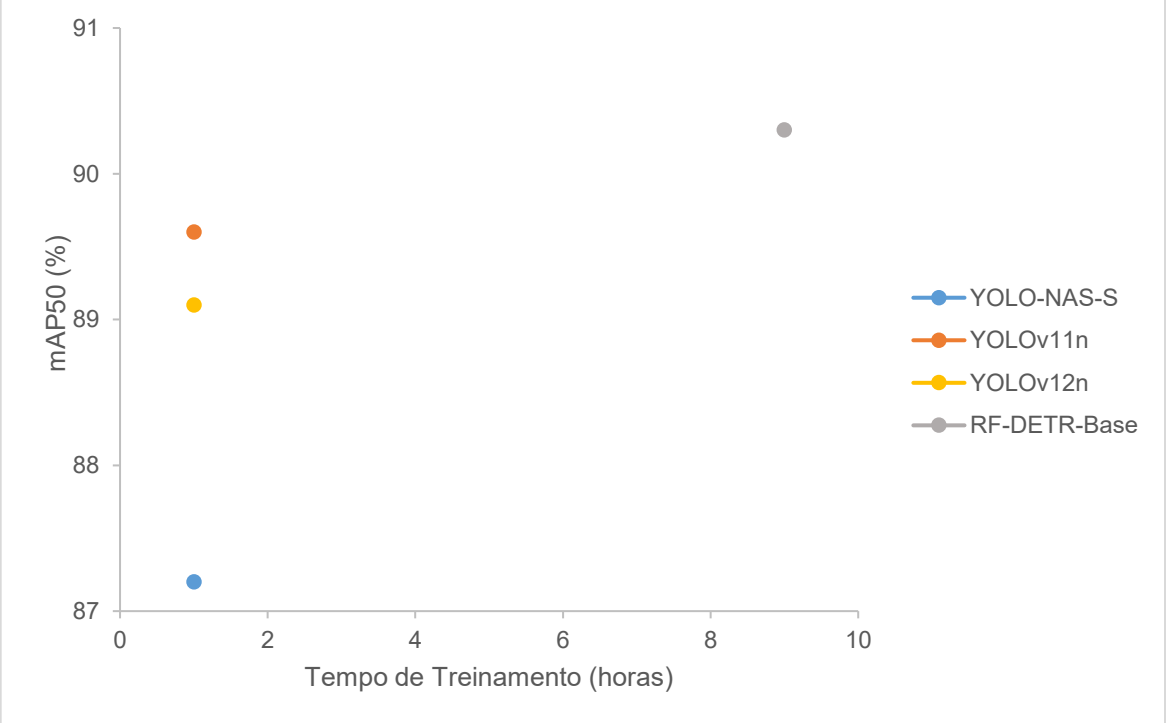
No ambiente local, o YOLOv11s apresenta um bom equilíbrio entre TT e mAP50, onde alcançou 88,6% em apenas 3,23 horas. Apesar dos modelos YOLOv12s e YOLOv11m apresentarem um maior mAP50, respectivamente 89,0% e 89,7%, os seus TTs são significativamente maiores. Dessa forma, constata-se que o YOLOv11s é 21x mais rápido para treinar do que o YOLOv11m e aproximadamente 8x mais rápido que o YOLOv12.

Vale mencionar que apesar do YOLOv12 empregado ser *small*, o seu tempo de treinamento foi elevado em comparação as versões *small* do YOLOv8 e YOLOv11. Isso pode ser justificado pelos mecanismos de atenção empregados na arquitetura do YOLOv12, que apesar de fornecerem bons resultados, comprometem a velocidade de treinamento.

Os modelos YOLOv8 por sua vez, além de apresentarem valores menores de mAP50, também demonstraram um comportamento atípico para esse banco de dados: uma queda de mAP50 da versão *small* para a versão *medium*.

O Gráfico 9 apresenta a relação entre TT e mAP50 para o BD1 no ambiente *online*.

Gráfico 9 - Relação entre o TT e mAP50 para o BD1 no ambiente *online*



Fonte: Autoria Própria (2025)

Em relação ao ambiente *online*, verifica-se um TT mais rápido que o ambiente local em todos os modelos, com a única exceção sendo o RF-DETR-Base, que efetuou o treinamento em 9 horas e obteve o maior mAP50 (90,3%).

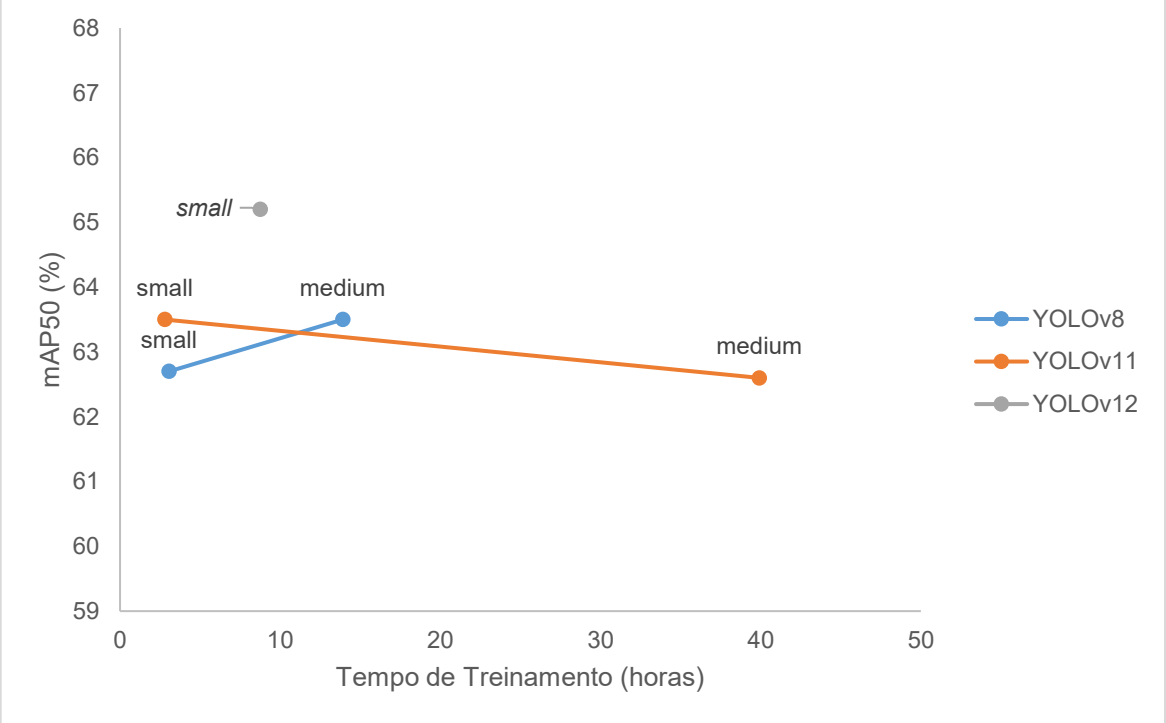
Para este ambiente, o melhor equilíbrio entre TT e mAP50 ocorreu no YOLOv11n, que em apenas 1 hora de treinamento alcançou 89,6% de mAP. Esse valor é 1% maior que o obtido pelo YOLOv11s no ambiente local e com apenas 0,1% de diferença em relação ao YOLOv11m, que demorou cerca de 68 horas para treinar localmente.

Além disso, nota-se que o YOLOv11n apresenta um melhor desempenho em relação ao RF-DETR-Base, pois apesar de obter 0,7% a menos no mAP50, o seu treinamento foi 6x mais rápido.

6.3.2 Banco de Dados 2 (BD2)

Os resultados relativos ao TT e mAP50 do BD2 treinado no ambiente local podem ser visualizados no Gráfico 10.

Gráfico 10 - Relação entre o TT e mAP50 para o BD2 no ambiente local



Fonte: Autoria Própria (2025)

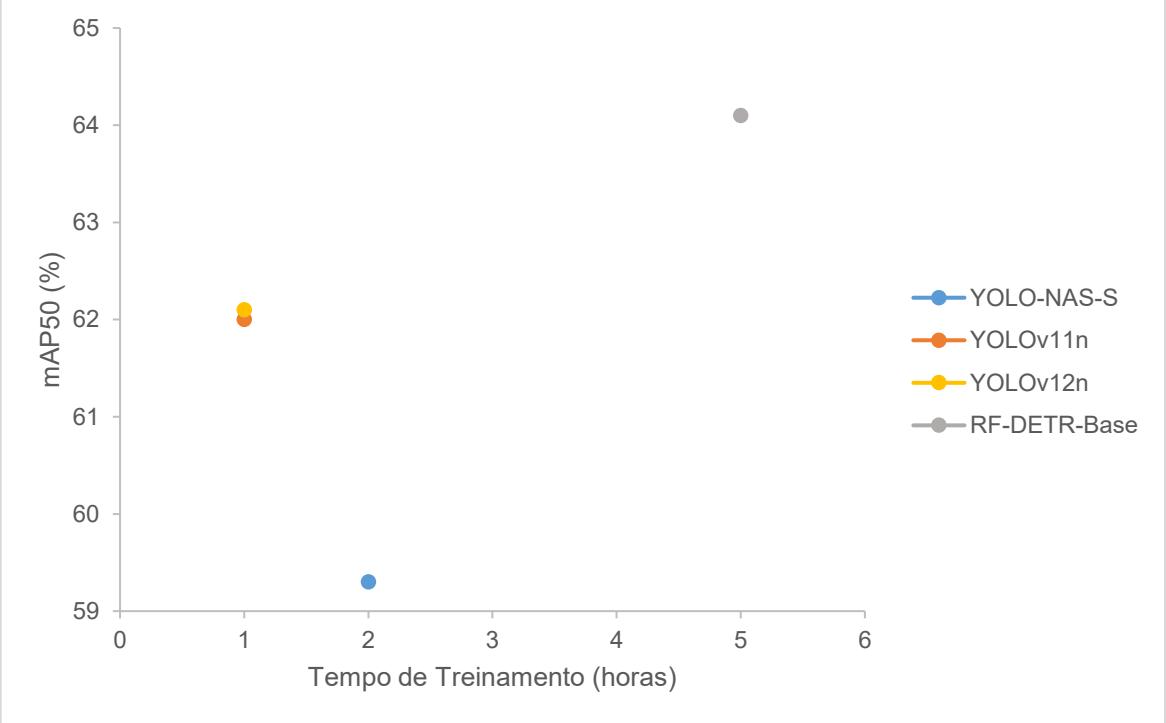
Através do Gráfico 10, a primeira coisa a se notar é o comportamento do YOLOv11, que obteve um mAP menor na versão *medium* quando comparado a versão *small.* Além disso, o mAP da YOLOv11m foi o menor dentre todos os modelos empregados, podendo inferir que essa versão em específico não conseguiu se adaptar ao conjunto de dados.

Também é possível perceber que dentre esses 5 modelos, o YOLOv12s apresenta a melhor eficiência, uma vez que ele apresenta o maior mAP enquanto possui um tempo de treinamento não tão lento (aproximadamente 9 horas). Os melhores resultados para esse modelo novamente podem ser justificados pelos seus mecanismos de atenção.

Dessa forma, os modelos YOLOv12s e YOLOv11s podem ser considerados os modelos com melhor desempenho de treinamento, uma vez que ambos (nos dois bancos de dados) possuem um bom equilíbrio entre mAP50 e TT.

Já para o ambiente *online*, a relação entre o TT e o mAP50 no BD2 é apresentada no Gráfico 11.

Gráfico 11 - Relação entre o TT e mAP50 para o BD2 no ambiente *online*



Fonte: Autoria Própria (2025)

Através do gráfico acima, constata-se que dessa vez todos os modelos apresentaram um TT rápido, onde o modelo mais demorado, o RF-DETR-Base, levou 5 horas para treinar. Dessa forma constata-se que este modelo apresenta o melhor desempenho no ambiente online, equilibrando bem mAP50 com o TT.

6.4 Comparativo entre Ambiente Online e Ambiente Local

Conforme perceptível no tópico anterior, o ambiente online apresenta um tempo relativamente menor para treinar os modelos de DL em comparação ao ambiente local. Além disso, o ambiente *online* apresenta outras vantagens como:

- Integração entre as etapas: Através do Roboflow é possível efetuar a anotação das imagens, pré-processamento, treinamento e avaliação dos modelos em um só lugar;
- Fácil e intuitivo de utilizar: A plataforma Roboflow, apresenta um *design* simples e não necessita da utilização de códigos, sendo fácil e intuitiva de utilizar por pessoas sem familiaridade com a área da programação;
- Acesso estudantil: Estudantes de graduação e pós-graduação podem usufruir de um acesso estudantil (com algumas limitações) apenas

utilizando o e-mail institucional.

Contudo, algumas desvantagens perceptíveis durante o uso da plataforma precisam ser ressaltadas, incluindo:

- Limitações do Acesso Estudantil: Apesar de fornecer o acesso estudantil, existem diversas limitações impostas pela plataforma. Dentre elas, a impossibilidade de utilizar modelos mais robustos (*e.g.* versões *large* e *extra-large*) e visualização incompleta de todos os resultados;
- Suporte limitado para modelos de DL: O Roboflow não fornece suporte a todos os modelos de DL existentes, apenas a versões mais recentes de algumas arquiteturas;
- Controle sob o processo de treinamento e teste: Não é possível efetuar personalização dos modelos e nem modificar os hiper parâmetros de treinamento (*e.g.* número de épocas);
- Necessidade de internet: É necessário estar conectado na internet para utilizar o Roboflow*,* visto que ela é uma plataforma *online.*

As vantagens e desvantagens dos dois ambientes utilizados podem ser observados no Quadro 2.

Quadro 2 – Quadro comparativo: ambiente local e ambiente *online*

	ANACONDA.	roboflow
Gratuito	Sim	Não, porém fornece acesso estudantil (Com limitações)
Integração entre as etapas	Sim, porém é necessário utilizar várias bibliotecas	Sim, em um local só
Fácil e intuitivo de se utilizar	Sim, porém necessário conhecimento prévio em programação	Sim, sem a necessidade de conhecimento prévio
Tempo de Treinamento	Mais lento, porém depende da GPU do computador	Mais rápido
Utilização de modelos robustos	Sim, porém com elevado tempo de treinamento	Sim, porém necessário pagar
Suporte à diversos modelos	Sim	Não
Visualização de todos os resultados	Sim	Sim, porém necessário pagar
Liberdade de personalização do treinamento	Sim	Não
Necessidade de internet	Não	Sim

Fonte: Autoria Própria (2025)

Atrela-se ao exposto que apesar das vantagens do Roboflow, seu uso é limitado, principalmente na análise de resultados de treinamento e suporte a outros modelos. Ressalta-se que existem outros ambientes *online* que podem ser explorados para o treinamento de modelos de DL, como o Google Colab e o Ultralytics Hub.

6.5 Etapa de Teste dos Modelos

Conforme abordado anteriormente, o Roboflow apresenta algumas limitações, principalmente no que tange a liberdade de personalização. Dessa forma, optou-se por realizar a etapa de teste localmente, possuindo assim mais controle sobre as informações. A principal dessas informações é o limiar de confiança, que foi definido como 50%. Assim, o modelo só irá prever *bounding boxes* com o nível de confiança

igual ou superior a 50%, ignorando previsões abaixo dessa porcentagem.

Devido a grande quantidade de modelos treinados, optou-se por escolher apenas dois para essa etapa de teste. Os modelos escolhidos foram o YOLOv11s e o YOLOv12s, devido seus elevados valores nas métricas de avaliação e o bom equilíbrio entre mAP50 e tempo de treinamento.

Primeiramente o teste foi conduzido para os modelos treinados através do BD1, onde verificou-se a capacidade de cada modelo em reconhecer os defeitos. Em seguida, o teste foi efetuado em modelos treinados com o BD2, e sua capacidade de reconhecer o defeito e o nível de severidade simultaneamente foi analisada. Por fim, a validação prática do modelo com os melhores resultados na etapa de teste foi conduzido em um dos ortomosaicos elaborado por Garcia (2025).

6.5.1 Banco de Dados 1 (BD1)

Na Tabela 4 é possível observar a Matriz de Confusão para o modelo YOLOv11s treinado com o BD1. Nesta matriz a linha Classe Real (CR) indica a classe a que a imagem de fato pertence, podendo ser Trinca por Fadiga (TF), Remendo (R), TPV, Buraco (B) e *background* (BG). Já a Classe Predita, corresponde à predição do modelo, também podendo ser TF, R, TPV, B ou BG. Vale ressaltar que as linhas correspondem aos Falsos Negativos (FN) e as colunas aos Falsos Positivos (FP).

Tabela 4 - Matriz de Confusão YOLOv11s (BD1)

Modelo	CR	Classe Predita					Métricas (%)		
		TF	R	TPV	B	BG	Precisão	Recall	F1-score
YOLOV11s	TF	28	0	0	0	2	82,35	93,33	87,50
	R	0	23	0	2	13	95,83	60,53	74,19
	TPV	0	0	10	0	0	100,00	100,00	100,00
	B	6	0	0	18	9	85,71	54,55	66,67
	BG	0	1	0	1	-	-	-	-
	Total						**90,98**	77,10	83,47

Fonte: Autoria Própria (2025)

Para o modelo o YOLOv11s é perceptível um excelente resultado em relação à precisão, onde, além do resultado geral de 90,98%, a classe TPV obteve um valor de 100% nessa métrica. Esse valor é justificado, pois não ocorreram FP, ou seja, todas as *bounding boxes* correspondentes a essas classes foram previstas corretamente.

O remendo também obteve uma precisão elevada (95,83%), com apenas um FP, conforme é possível observar na Figura 30, onde o modelo identificou duas *bounding boxes* ao invés de uma só. Essa sobreposição de *bounding box* também ocorreu em uma das imagens do defeito buraco (Figura 31).

Figura 30 - Sobreposição de *bounding box* na classe remendo

Fonte: Autoria Própria (2025)

Figura 31 - Sobreposição de *bounding box* na classe buraco

Fonte: Autoria Própria (2025)

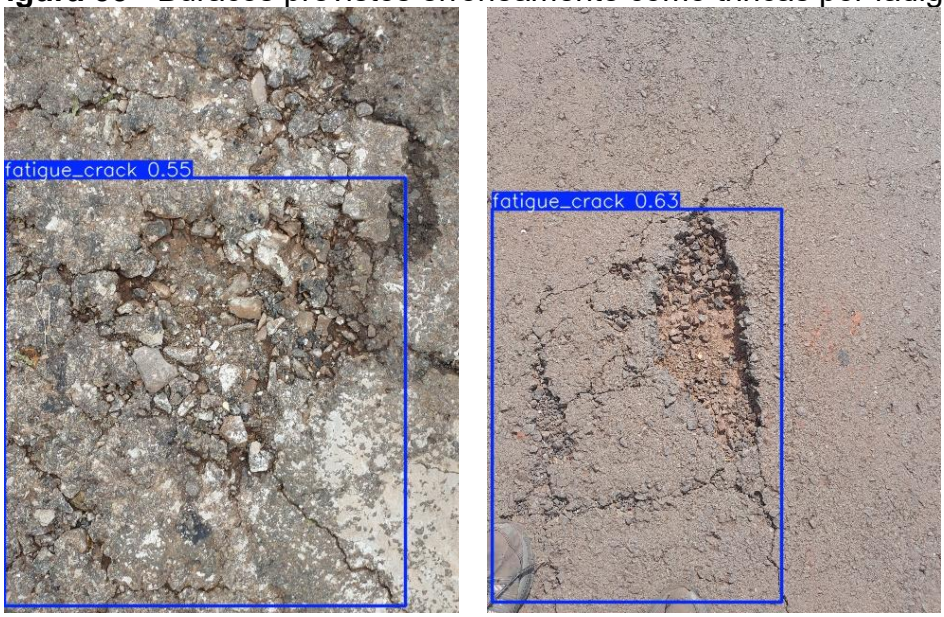
Além do FP apresentado na Figura 31, o modelo também efetuou outras duas

previsões erradas para a classe buraco, que foi prevista em imagens de remendos, conforme é possível observar na Figura 32. Essas previsões impactaram diretamente a métrica precisão do defeito buraco, que foi de 85,71%. O menor valor de precisão, de 82,35%, ocorreu na classe trinca por fadiga, onde o modelo efetuou 6 previsões erradas, prevendo a classe em questão em imagens de buracos (Figura 33).

Figura 32 - Remendos previstos erroneamente como buracos no YOLOv11s

Fonte: Autoria Própria (2025)

Figura 33 - Buracos previstos erroneamente como trincas por fadiga



Fonte: Autoria Própria (2025)

É perceptível que os remendos da Figura 32 possuem uma grande quantidade

de brita aparente, um padrão que o modelo pode ter identificado como exclusivo de buracos. Além disso, a *bounding box* com 72% é marcada em um remendo levemente desgastado, e que além de estar próximo de trincas tem um formato similar ao de um buraco, uma vez que este foi mal executado. Esses fatores também podem ter induzido o modelo a erros.

Já em relação a Figura 33, observa-se buracos que foram previstos erroneamente como trincas por fadiga. Além das imagens apresentadas na Figura 33, outros quatro FPs similares ocorreram, porém em todos os casos as previsões erradas apresentam uma confiança abaixo de 70%, o que pode ser corrigido alterando o limiar de confiança de teste (que para esse trabalho foi definido como 50%).

Para a métrica *recall*, o desempenho do modelo foi consideravelmente menor (77,10%), uma vez que diversos FNs ocorreram. A única exceção foi a TPV, que obteve 100% nesta métrica, pois todas as TPVs presentes nas imagens foram identificadas corretamente. As trincas por fadiga também obtiveram um *recall* elevado, alcançando 93,33% e com a ocorrência de apenas dois FNs (Figura 34).

Figura 34 - Trincas por fadiga não detectadas pelo YOLOv11s

Fonte: Autoria Própria (2025)

Na Figura 34 constata-se que essas trincas por fadiga são muito finas, e sem padrões definidos (severidade baixa), o que pode ter influenciado na ausência de reconhecimento pelo modelo. Vale ressaltar que essas trincas podem ter sido detectadas, porém com um confiança menor que 50%. Para comprovar isso, o teste

necessitaria ser feito com um limiar abaixo de 50%, algo que não é recomendável, pois novos FPs e FNs podem ocorrer.

Treze remendos e nove buracos também não foram detectados pelo modelo. Isso impacta diretamente na métrica *recall*, que apresentou os menores valores nessas classes, 60,53% e 54,55% respectivamente. Além da ausência de detecções, os dois remendos detectados como buracos (Figura 32) e os seis buracos detectados como trinca por fadiga (Figura 33) também influenciaram no valor de recall.

Apesar de apresentar um bom desempenho, com diversas métricas acima de 90%, o modelo YOLOv11s teve seu funcionamento afetado por diversas previsões incorretas realizadas nas classes buracos e remendos, mesmo obtendo 98,1% de mAP50 para a classe buraco na etapa de treinamento.

Para o YOLOv12s treinado no BD1 a matriz de confusão pode ser visualizada na Tabela 5.

Tabela 5 - Matriz de confusão YOLOv12s (BD1)

Modelo	CR	Classe Predita					Métricas (%)		
		TF	R	TPV	B	BG	Precisão	Recall	F1-score
YOLOV12s	TF	26	0	0	0	4	76,47	86,67	81,25
	R	1	32	0	2	2	84,21	86,49	85,33
	TPV	0	0	9	0	1	100,00	90,00	94,74
	B	4	0	0	27	2	84,38	81,82	83,08
	BG	3	6	0	3	0	-	-	-
	Total						86,26	**86,24**	**86,25**

Fonte: Autoria Própria (2025)

Através da Tabela 5, verifica-se que o YOLOv12s obteve resultados de teste ligeiramente melhores que o YOLOv11s, apresentando um *recall* 9% maior e uma *f1-score* aproximadamente 3% maior. Apenas a métrica de precisão apresentou um resultado cerca de 5% menor.

O aumento no *recall* do YOLOv12s é justificado pela eliminação de diversos FNs presentes na coluna BG. O YOLOv11s apresentava treze remendos e nove buracos sem detecção, enquanto o YOLOv12 reduziu ambos os valores para dois. Verificou-se que um buraco e um remendo coincidiram de não serem detectados em ambos os modelos. Esses dois defeitos podem ser visualizados na Figura 35.

Figura 35 - Defeitos não detectados pelo YOLOv11s e pelo YOLOv12s

Fonte: Autoria Própria (2025)

Contudo, apesar da redução do número de detecções ausentes para as classes remendo e buraco, esse valor aumentou para as trincas por fadiga e TPV. Dessa forma, é perceptível que para essas classes o YOLOv11s oferece um melhor desempenho, tanto em precisão, quanto em *recall*.

A queda de precisão é notável para diversas classes, principalmente as trincas por fadiga e para os remendos. O YOLOv12s apesar de apresentar uma boa capacidade de reconhecer imagens com defeitos, ele apresenta uma maior quantidade de FP, dentre eles diversas sombras detectadas como remendo (Figura 36), que não ocorreram no YOLOv11s. Além disso, ressalta-se que os remendos detectados como buracos da Figura 32 também ocorreram no YOLOv12s, assim como detecções erradas de trincas por fadiga em imagens de buracos.

Figura 36 - Sombras detectadas como remendo no YOLOv12s

Fonte: Autoria Própria (2025)

Atrela-se ao exposto que o YOLOv12s, apesar de apresentar uma maior quantidade de FP, demonstrou ser mais eficiente no reconhecimento de defeitos como remendo e buracos. Os resultados alcançados por esse modelo, onde todas as métricas de avaliação apresentaram valores acima de 75%, indicam que o mesmo pode ser aplicado em campo.

6.5.2 Banco de Dados 2 (BD2)

A matriz de confusão do modelo YOLOv11s treinada através do BD2 pode ser vista na Tabela 6. Nesta matriz, além da indicação das classes de defeitos Trinca por Fadiga (TF), Remendo (R), Buraco (B) também é indicado o nível de severidade de cada um, sendo Severidade Alta (SA), Severidade Média (SM) e Severidade Baixa (SB), bem como a classe TPV e o *background* (BG).

Tabela 6 - Matriz de confusão YOLOv11s (BD2)

Modelo	CR		Classe Predita											Métricas (%)		
			TF			R			B			TPV	BG	Pr	Rc	F1
			SA	SM	SB	SA	SM	SB	SA	SM	SB					
YOLOv11s	TF	SA	9	0	0	0	0	0	0	0	0	0	1	56,25	90,00	69,23
		SM	3	2	1	0	0	0	0	0	0	0	4	66,67	20,00	30,77
		SB	0	0	8	0	0	0	0	0	0	0	2	80,00	80,00	80,00
	R	SA	0	0	0	7	0	0	0	0	0	0	5	43,75	58,33	50,00
		SM	0	0	0	8	0	0	0	0	0	0	6	0,00	0,00	0,00
		SB	0	0	0	1	0	8	0	0	0	0	2	80,00	72,73	76,19
	B	SA	1	0	0	0	0	0	4	2	0	0	3	66,67	40,00	50,00
		SM	0	0	0	0	0	0	2	4	0	0	4	50,00	40,00	44,44
		SB	0	0	0	0	0	0	0	2	3	0	5	75,00	30,00	42,86
	TPV		0	0	0	0	0	0	0	0	0	10	0	83,33	100,00	90,91
	BG		3	1	1	0	0	2	0	0	1	2	0	-	-	-
			Total											60,17	**53,11**	53,44

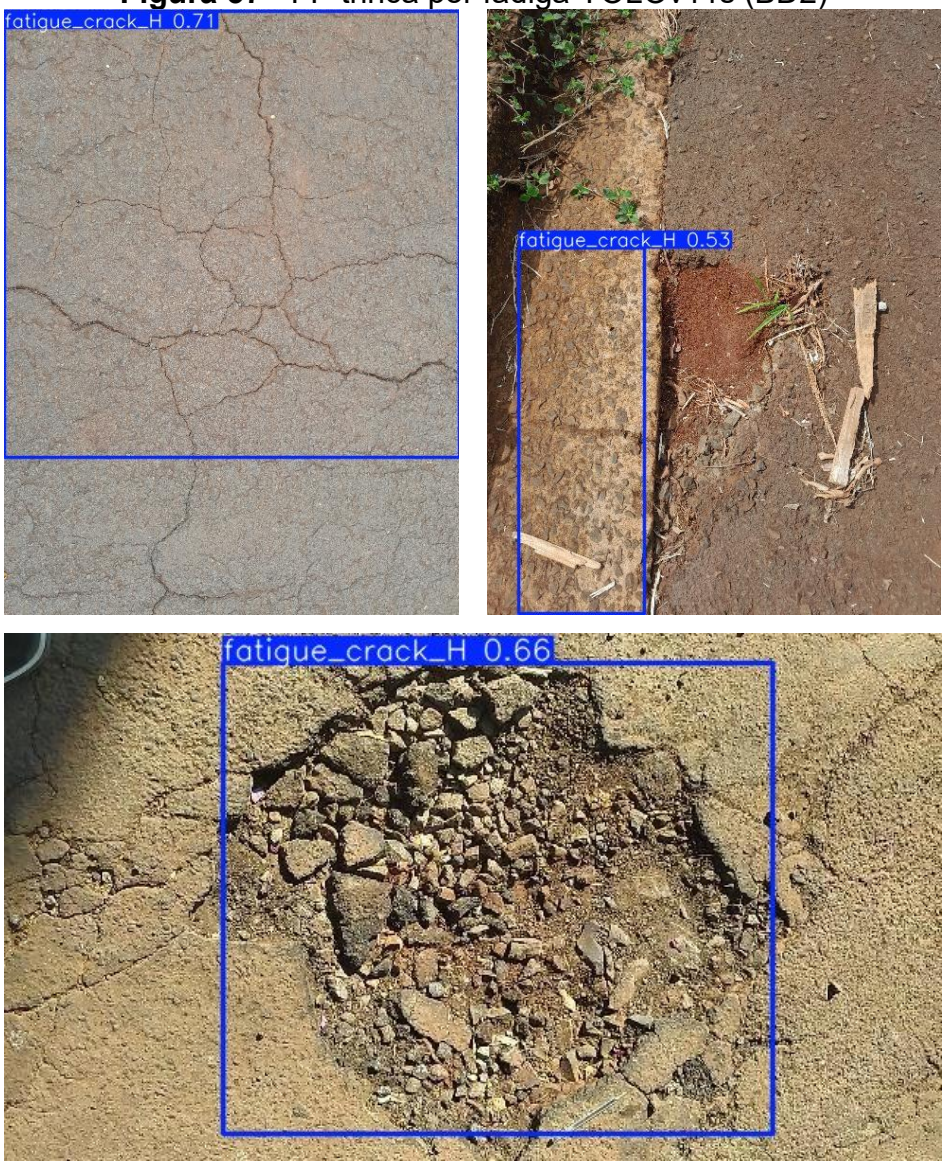
Fonte: Autoria Própria (2025)

Por meio dos valores presentes na Tabela 6, é notável que, assim como no treinamento, o BD2 obteve métricas de avaliação inferiores ao BD1. Assim, novamente, é possível inferir que os modelos de detecção de objetos possuem um melhor desempenho para reconhecer defeitos em pavimentos, mas apresentam dificuldades no reconhecimento do defeito em conjunto com o nível de severidade.

Apesar de uma precisão média de 60,17% e *recall* médio de 53,11%, o modelo YOLOv11s apresentou um bom desempenho, em ambas as métricas, para algumas classes individuais, sendo elas a trinca por fadiga de severidade baixa, com 80% de precisão e *recall* e o remendo de severidade baixa, com 80% de precisão e 72,73% de *recall*. Com esses resultados é possível verificar que o YOLOv11s tem maior facilidade no reconhecimento de defeitos com o nível de severidade baixa.

Em contrapartida, para as demais severidades, o YOLOv11s apresenta uma certa dificuldade de reconhecimento. Para as trincas por fadiga com severidade alta, por exemplo, é notável um alto *recall*, pois poucos FNs foram identificados. Contudo, a precisão de 56,25% indica a ocorrência de diversos FPs (Figura 37), sendo os principais: trincas de severidades média, *background* e buraco de severidade alta.

Figura 37 - FP trinca por fadiga YOLOv11s (BD2)



Fonte: Autoria Própria (2025)

Paras as trincas de fadiga com a severidade média, é perceptível a pouca quantidade de TP, onde dentre as 10 imagens, o modelo identificou apenas 2 corretamente. Verifica-se também que três trincas com esse nível de severidade foram identificadas erroneamente com a severidade alta. Essa constatação sustenta a hipótese de que um BD3 melhoraria o desempenho dos modelos, pois é notável a dificuldade que eles possuem em distinguir nível de severidade parecidos.

Isso fica mais evidente quando o defeito remendo é levado em consideração. A severidade média neste caso, obteve 0% de precisão e 0% de *recall*, ou seja, o modelo não efetuou nenhum acerto. Além disso, dos 14 remendos com severidade média, oito foram detectados erroneamente como severidade alta. Esse fato além de

impactar nas métricas do remendo severidade média, também influenciou no remendo de severidade alta, que obteve apenas 43% de precisão, devido aos FP mencionados, que podem ser visualizadas na Figura 38.

Figura 38 - Remendos severidade média identificados erroneamente

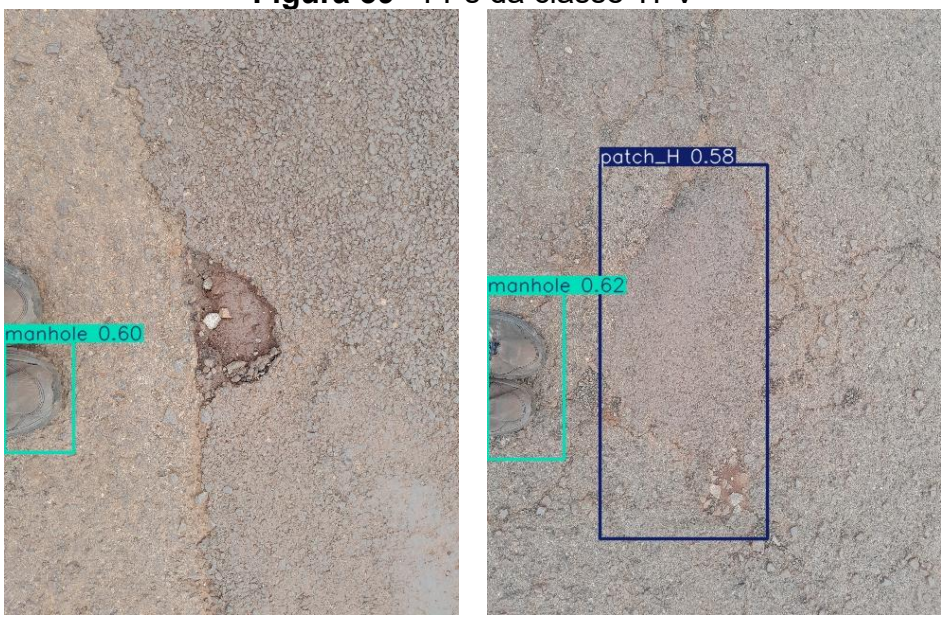
Fonte: Autoria Própria (2025)

O defeito do tipo buraco, conforme já comentado, é um caso à parte, pois é o único dentre os defeitos estudados em que as propriedades geométricas são levadas em consideração para determinação do nível de severidade. Dessa forma, devido à ausência de padrões visuais, o modelo apresenta dificuldades em efetuar as previsões corretas, uma vez que ele não tem acesso as medidas obtidas em campo.

De forma geral, o defeito do tipo buraco apresentou uma boa precisão para a severidade baixa (75%) e valores razoáveis para a severidade média (50%) e alta (66,67%). Porém, todos obtiveram valores abaixo de 50% para a métrica *recall*, o que indica que o YOLOv11s apresenta dificuldade em identificar buracos e seus níveis de severidade, mas quando identifica, geralmente ele está correto.

Por fim, na classe TPV ocorreu uma queda de precisão em relação ao treinamento (onde todas as métricas tiveram o valor de 100%). Essa redução é justificada, pois na etapa de teste o modelo identificou em duas imagens um elemento externo à superfície do pavimento como uma TPV, conforme é notável na Figura 39.

Figura 39 - FPs da classe TPV



Fonte: Autoria Própria (2025)

Em sequência, a matriz de confusão do modelo YOLOv12s, também treinadas com o BD2, pode ser observada na Tabela 7.

Tabela 7 - Matriz de confusão YOLOv12s (BD2)

Modelo	CR		Classe Predita									TPV	BG	Métricas (%)		
			TF			R			B							
			SA	SM	SB	SA	SM	SB	SA	SM	SB			Pr	Rc	F1
YOLOv12s	TF	SA	10	0	0	0	0	0	0	0	0	0	0	71,43	100,00	83,33
		SM	3	1	0	0	0	0	0	0	0	0	6	50,00	10,00	16,67
		SB	0	0	3	0	0	0	0	0	0	0	7	60,00	30,00	40,00
	R	SA	0	0	0	8	0	0	0	0	0	0	4	50,00	66,67	57,14
		SM	0	0	0	6	0	0	0	0	0	0	8	0,00	0,00	0,00
		SB	0	0	0	1	0	9	0	0	0	0	1	100,00	81,82	90,00
	B	SA	0	0	0	0	0	0	7	0	0	0	3	87,50	70,00	77,78
		SM	0	0	0	0	0	0	1	5	0	0	4	71,43	50,00	58,82
		SB	0	0	0	0	0	0	0	2	2	0	6	100,00	20,00	33,33
	TPV		0	0	0	0	0	0	0	0	0	9	1	100,00	90,00	94,74
	BG		3	1	1	2	1	0	0	0	0	0	0	-	-	-
			Total											**69,04**	51,85	**55,18**

Fonte: Autoria Própria (2025)

Através da Tabela 7, nota-se que o YOLOv12s apresentou uma precisão maior, aproximadamente 9%, em comparação ao YOLOv11s. Dessa forma, o YOLOv12s apresenta uma maior capacidade em indicar corretamente *bounding boxes,* onde nenhum FP foi identificado para as classes remendo severidade baixa, buraco

severidade baixa e TPV, ou seja, todas obtiveram 100% de precisão. Outros bons resultados de precisão ocorreram nas classes trinca por fadiga severidade alta (71,43%), buraco severidade alta (87,5%) e buraco severidade média (71,43%).

Em contrapartida aos bons resultados de precisão, o *recall,* foi cerca de 1% menor quando comparado ao YOLOv11s. Esse decréscimo de *recall,* é justificado pelo maior número de ocorrência de FNs, principalmente na classe trinca por fadiga de severidade baixa, que foi de 30% no YOLOv12s, valor 50% menor quando comparado ao YOLOv11s.

Apesar dessa dificuldade do modelo em reconhecer trincas por fadiga com severidade baixa, nota-se uma maior capacidade de detecção de buracos, alcançando 70% na severidade alta e 50% na severidade média. Os buracos com severidade baixa, apesar de um baixo recall (20%) obtiveram a precisão de 100%.

Por fim, novamente percebe-se a dificuldade do modelo em reconhecer o nível de severidade média para os defeitos remendo e trinca por fadiga. Das 10 imagens de trincas de severidade média, apenas 1 foi identificada corretamente, enquanto para o remendo, novamente (de forma semelhante ao YOLOv11s), nenhum remendo de severidade média foi identificado corretamente.

Apesar do seu demasiado tempo de treinamento e da sua baixa capacidade de reconhecer trincas por fadiga de severidade baixa e média, o YOLOv12s apresenta um desempenho elevado para as demais classes, principalmente no que tange os defeitos remendo e buraco. Dessa forma, constata-se que o YOLOv12s é o modelo mais recomendado para aplicação prática.

6.5.3 Validação prática (ortomosaico)

A etapa de validação prática foi realizada utilizando o ortomosaico da UA141 confeccionado por Garcia (2025). Na UA141, a autora constatou, através do uso do software AutoCAD Civil 3D, a presença de 33 trincas por fadiga (8 de severidade baixa e 25 de severidade média), 1 buraco de severidade baixa e 2 remendos (1 de severidade média e 1 de severidade alta). Esses valores serão utilizados como *ground-truth*, possibilitando a comparação entre a quantidade real de defeitos e a quantidade detectada nos experimentos realizados.

No total, foram conduzidos 18 experimentos utilizando o modelo que obteve os melhores resultados na etapa de teste, ou seja, o YOLOv12s. Para cada um dos

experimentos o tamanho dos *slices* e o nível de confiabilidade (Conf.) foram sendo alternados visando compreender o impacto desses parâmetros na detecção. Vale ressaltar que o único parâmetro não variado foi o *overlap*, que para todos os experimentos foi mantido o valor padrão de 0,2.

Além disso, essa etapa foi realizada tanto no modelo capaz de reconhecer apenas defeitos (YOLOv12s-BD1) quanto no capaz de reconhecer defeitos em conjunto com os níveis de severidade (YOLOv12s-BD2). Uma síntese dos 18 experimentos efetuados pode ser observada na Tabela 8.

Tabela 8 – Síntese das informações dos 18 experimentos

YOLOv12s-BD1			YOLOv12s-BD2		
Experimento	*Slice*	Conf.	Experimento	*Slice*	Conf.
E1	640x640	0,5	E10	640x640	0,5
E2	1280x1280	0,5	E11	1280x1280	0,5
E3	1920x1920	0,5	E12	1920x1920	0,5
E4	640x640	0,7	E13	640x640	0,7
E5	1280x1280	0,7	E14	1280x1280	0,7
E6	1920x1920	0,7	E15	1920x1920	0,7
E7	640x640	0,9	E16	640x640	0,9
E8	1280x1280	0,9	E17	1280x1280	0,9
E9	1920x1920	0,9	E18	1920x1920	0,9

Fonte: Autoria Própria (2025)

Os resultados referentes à validação prática do YOLOv12s-BD1 (E1 até E9) podem ser constatados na Tabela 9. Com o intuito de facilitar a visualização da quantidade real de defeitos e das detecções efetuadas durante os experimentos, utilizou-se uma escala de cores Verde-Amarela-Vermelha. Dessa forma, a tonalidade verde representa a baixa quantidade de defeitos, enquanto a vermelha indica quantidades elevadas. A tonalidade amarela, por sua vez, corresponde a valores intermediários entre os extremos.

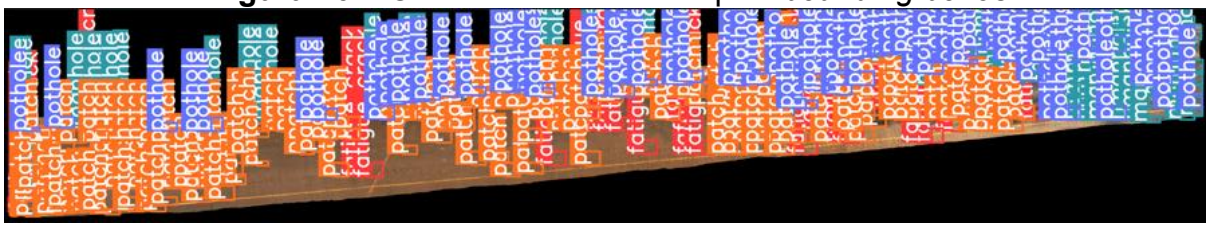
Tabela 9 – Resultados dos Experimentos 1 ao 9

Defeitos	Real	Detectados								
		E1	E2	E3	E4	E5	E6	E7	E8	E9
Trinca por fadiga	33	60	78	55	41	60	45	14	23	11
Buracos	1	86	35	16	61	27	11	13	8	1
Remendos	2	171	37	11	115	22	9	24	7	1
TPVs	0	125	12	6	99	5	1	30	0	0

Fonte: Autoria Própria (2025)

Através da Tabela 9 é notável que o modelo efetuou uma quantidade elevada de detecções em vários experimentos, onde inúmeros FPs foram registrados para todas as classes de defeitos. A pior situação ocorreu em E1, onde o modelo identificou 442 defeitos em uma UA que originalmente apresenta apenas 36. Esse comportamento de E1 é justificado pela utilização de *slices* pequenos em conjunto com uma baixa confiabilidade. Essa combinação, além de efetuar uma grande quantidade de detecções erradas, também impossibilita a análise do pavimento através do ortomosaico, devido ao número de *bounding boxes* geradas (Figura 40).

Figura 40 – Ortomosaico com múltiplas *bounding boxes*



Fonte: Autoria Própria (2025)

No experimento E4, menos detecções foram efetuadas em relação a E1, devido ao aumento da confiabilidade de 0.5 para 0.7. Porém, por ter mantido o *slice* em 640x640, o ortomosaico permaneceu poluído com *bounding boxes*.

A ineficácia de utilizar o *slice* 640x640 pode ser confirmada ao verificar o experimento E7, que mesmo com uma confiabilidade elevada de 0.9, apresentou diversos FPs. Esse fato ocorre porque slices menores fazem o modelo perceber detalhes demais, interpretando erroneamente partes do ortomosaico como classes válidas.

Um exemplo disso são as 30 TPVs detectadas de forma errada. Na Figura 41, é perceptível que essas detecções foram causadas por sombras, que alteram o contraste e a textura da imagem, o que favorece detecções errôneas.

Figura 41 – Sombras detectadas como TPVs

Fonte: Autoria Própria (2025)

Através da Tabela 9, também é perceptível que a utilização conjunta de *slices* grandes e confiabilidades elevadas, torna o modelo mais criterioso. Isso fica evidente em E8 e E9, onde os *slices* de 1280x1280 e 1920x1920 respectivamente, foram utilizados em conjunto com a confiabilidade de 0.9. Essa combinação é efetiva pois enquanto os *slices* grandes preservam mais contexto do ortomosaico, a confiabilidade de 0.9 reduz significativamente a quantidade de FPs.

Enquanto E8 demonstrou melhor reconhecimento de trincas por fadiga, detectando 23 de 33, o E9 apresentou um melhor desempenho para remendos e buracos. No entanto, é fundamental mencionar que ao observar o *output* de E9, percebe-se que as detecções de buracos e remendos realizadas correspondem a FPs. Esse comportamento demonstra que o número baixo de detecções não significa detecções corretas.

Dos FPs presentes em E9, é possível verificar na Figura 42 que uma sombra foi detectada como remendo, demonstrando a fragilidade do modelo para variação de iluminação. Na Figura 43 nota-se que uma parte do meio fio foi detectada como um buraco, mostrando que o modelo não é capaz de reconhecer o limite do pavimento, efetuando detecções fora da área pavimentada. Na Figura 44, verifica-se o problema

de sobreposição de *bounding boxes*, onde uma única trinca por fadiga foi detectada múltiplas vezes.

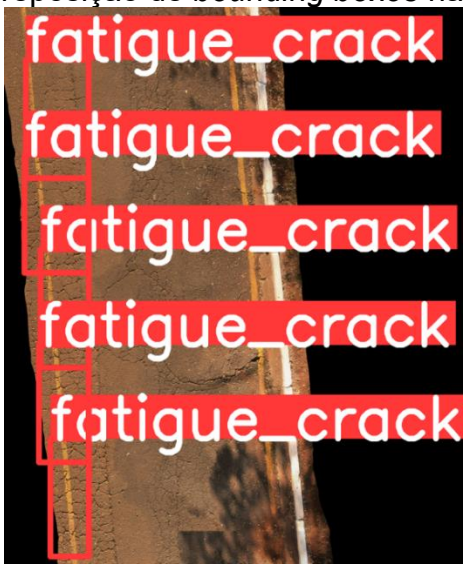
Figura 42 – Sombra detectada como remendo

Fonte: Autoria Própria (2025)

Figura 43 – Parte do meio fio detectada como buraco

Fonte: Autoria Própria (2025)

Figura 44 – Sobreposição de *bounding boxes* na trinca por fadiga



Fonte: Autoria Própria (2025)

Além disso, ressalta-se que, entre os 9 experimentos conduzidos no YOLOv12s-BD1, o modelo reconheceu corretamente apenas um dos remendos de forma parcial em E7 (Figura 45). O segundo remendo presente no ortomosaico, o buraco e diversas das trincas por fadiga não foram reconhecidas em nenhuma dos outros experimentos.

Figura 45 – Único remendo detectado corretamente

Fonte: Autoria Própria (2025)

Esse resultado evidencia que apesar das variações no tamanho de *slice* e de confiabilidade, o modelo apresenta limitações significativas na detecção de defeitos em ortomosaicos. Duas hipóteses podem ser levantadas para justificar a baixa qualidade dos resultados observados.

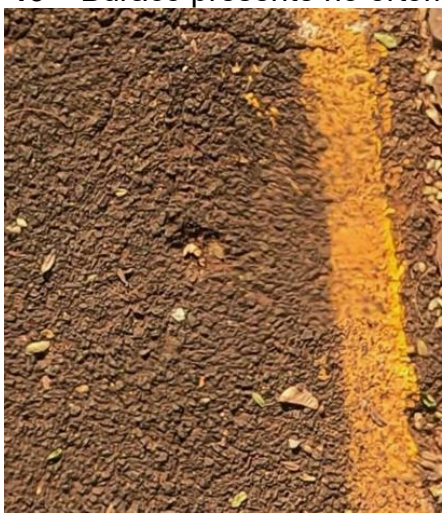
A primeira hipótese consiste em um problema clássico do *machine learning* chamado de mudança de domínio (*domain shift*). No presente trabalho os modelos foram treinados em imagens de smartphones tiradas do solo (*source domain*). Porém, esta validação prática foi executada em um ortomosaico confeccionado através de vídeos de uma GoPro posicionada a 2 metros do solo (*target domain*). Nagananda *et al.* (2021), destacam que os modelos treinados em imagens retiradas do solo não generalizam bem com imagens aéreas por uma série de fatores.

Dentre esses fatores, o ponto de vista é o mais crucial. O ponto de vista do ortomosaico é maior, cobrindo toda uma UA, ou seja, os defeitos presentes na pista ocupam menos pixels no ortomosaico do que nas imagens utilizadas para treinamento, o que impacta diretamente na capacidade de detecção do modelo. A utilização de técnicas de *domain adaptation* e a inclusão de imagens aéreas no banco de dados de treinamento podem trazer resultados melhores.

A segunda hipótese reside na relação do tamanho dos defeitos presentes no ortomosaico e do tamanho dos *slices* propostos. Os remendos e as trincas por fadiga longas não cabem nos *slices*, impossibilitando assim o seu reconhecimento completo

devido à falta de contexto. Já o buraco, devido ao seu tamanho (Figura 46), ocupa poucos pixels no ortomosaico, necessitando da utilização de *slices* pequenos. Porém, como constatado anteriormente, esse *slices* geram muitas *bounding boxes* e impedem a visualização do pavimento. A utilização de outros valores de *slices* além dos propostos, incluindo *slices* retangulares que cobrem a extensão longitudinal da pista, pode solucionar os problemas mencionados.

Figura 46 – Buraco presente no ortomosaico



Fonte: Autoria Própria (2025)

Para os nove experimentos executados no YOLOv12s-BD2 (E10-E18), foi confeccionada a Tabela 10 que, assim como a Tabela 9, apresenta a escala de cores Verde-Amarela-Vermelha. Através da Tabela 10, é possível verificar a quantidade real e a quantidade detectada, nos experimentos E10-E18, dos defeitos Trincas por Fadiga (TF), Buracos (B) e Remendos (R) e seus níveis de severidade.

Tabela 10 – Resultados dos Experimentos 10 ao 18

Defeitos	Real	Detectados								
		E10	E11	E12	E13	E14	E15	E16	E17	E18
TF_Baixa	8	6	0	0	1	0	0	0	0	0
TF_Média	25	7	0	0	1	0	0	0	0	0
TF_Alta	0	2	6	1	0	2	0	0	0	0
B_Baixa	1	4	0	0	0	0	0	0	0	0
B_Média	0	0	0	0	0	0	0	0	0	0
B_Alta	0	104	12	0	35	0	0	0	0	0
R_Baixa	0	22	5	1	1	1	0	0	0	0
R_Média	1	0	0	0	0	0	0	0	0	0
R_Alta	1	13	2	3	1	0	1	0	0	0
TPVs	0	16	0	0	3	0	0	0	0	0

Fonte: Autoria Própria (2025)

Por meio da Tabela 10 é possível verificar que nos experimentos E16, E17 e E18 o modelo não efetuou nenhuma detecção, indicando a sua incapacidade de realizar detecções com a confiabilidade acima de 90% em ortomosaicos, possivelmente pelas baixas métricas de avaliação obtidas na etapa de treinamento.

Nota-se também, que de forma geral esse modelo não se adaptou bem com nenhum dos *slices* propostos, efetuando poucas detecções nos *slices* 1280x1280 (E11 e E14) e 1920x1920 (E12 e E15) e muitas detecções no *slice* de 640x640 (E10 e E13). Esse comportamento mantém o padrão observado nos experimentos realizados no BD1, com *slices* pequenos favorecendo FPs e *slices* grandes reduzindo significativamente o número de detecções.

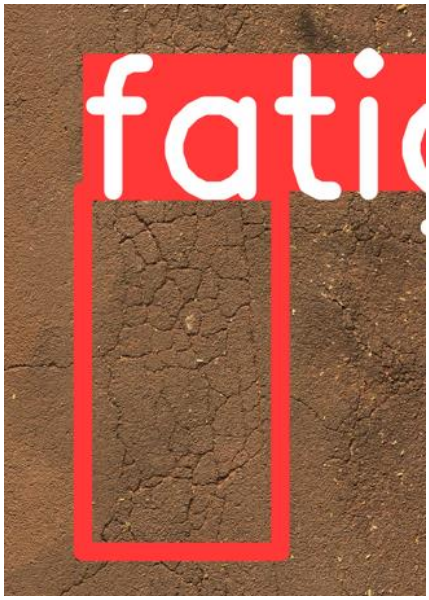
Além disso, foi constatado que tanto o buraco, quanto o remendo de severidade alta não foram identificados corretamente em nenhum do experimentos. O remendo de severidade média por sua vez, foi reconhecido parcialmente nos experimentos E12 e E14, porém com o nível de severidade baixa (Figura 47). Fato similar ocorreu com as trincas por fadiga de severidade média e baixa, que em muitos casos foram reconhecidas como severidade alta (Figura 48).

Figura 47 – Remendos detectados em E12 e E14

E12 E14

Fonte: Autoria Própria (2025)

Figura 48 – Trinca por fadiga de severidade média detectada como severidade alta



Fonte: Autoria Própria (2025)

Essa discrepância entre o nível de severidade predito pelo modelo e o nível de severidade real fortalecem a discussão levantada no tópico anterior acerca da subjetividade da norma ASTM D64333 e a necessidade de treinar um terceiro banco de dados. Além disso, os resultados adquiridos nos experimentos E10-E18 reforçam as hipóteses levantadas anteriormente.

Por fim, é válido ressaltar que apesar dos resultados apresentados serem insatisfatórios do ponto de vista prático, a detecção em ortomosaicos empregando o método SAHI é pouco explorada na literatura, especialmente na detecção de defeitos. Com o presente trabalho, verificou-se que a combinação YOLO+SAHI apresenta potencial, porém é fundamental que mais experimentos sejam efetuados, variando o tamanho dos *slices*, a confiabilidade, o *overlap* e empregando outros modelos de detecção de objetos.

6.6 Aplicativo Desenvolvido: Detector de Defeitos em Pavimentos (Beta)

O aplicativo desenvolvido, intitulado Detector de Defeitos em Pavimentos (DDP), no momento se encontra na versão beta, com algumas funcionalidades indisponíveis, mas que em breve serão implementadas. O DDP é um *software* de código aberto (*open source*), cujo código fonte e executável para instalação estão disponíveis publicamente no repositório GitHub (https://github.com/atilamarconcine).

Para execução do DDP, foram definidos os seguintes requisitos:

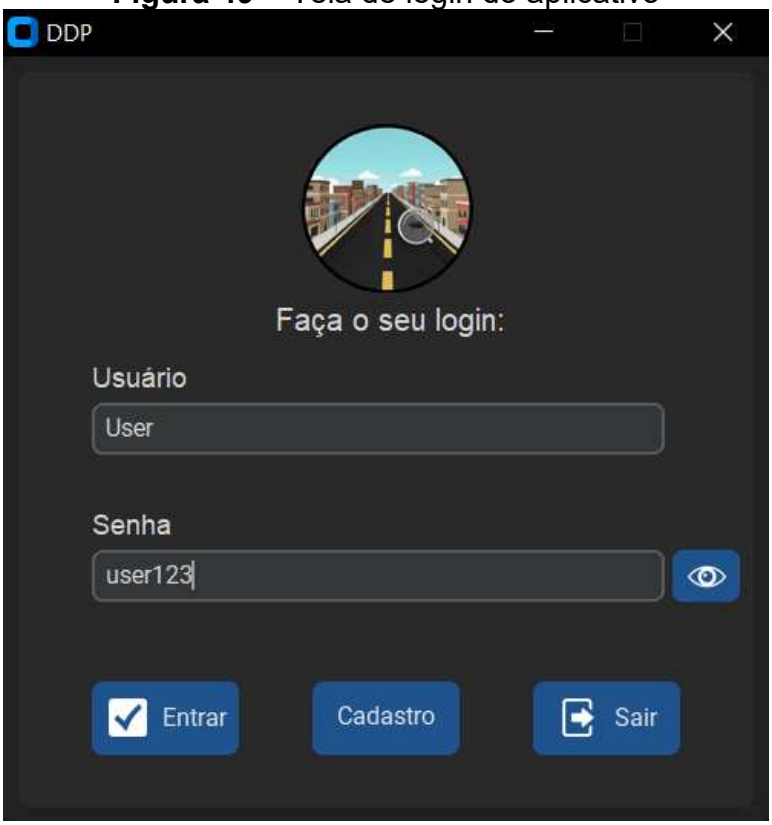
- **Sistema Operacional:** Windows 10 ou 11 (64bits);

- **Armazenamento:** 2 GB de espaço livre em disco;
- **Processador (CPU) (Mínimo):** Intel Core i5 (6ª geração), AMD Ryzen 3 (série 2000) ou equivalentes;
- **Processador (CPU) (Recomendado):** Intel Core i5 (10ª geração), AMD Ryzen 5 (série 3000) ou superior;
- **Memória RAM:** 8 GB (Mínimo), 16 GB (Recomendado).

Ressalta-se que na versão atual o aplicativo utiliza exclusivamente a CPU para realizar a detecção dos defeitos. Devido a elevada carga computacional gerada por esse processo, congelamentos momentâneos podem ocorrer na interface. Para contornar esse problema, as próximas versões do DDP terão suporte a GPU (placas NVIDIA com arquitetura CUDA).

Ao executar o aplicativo, o usuário é apresentado a uma tela de login (Figura 49), sendo possível digitar o usuário e a senha. Na versão beta, o preenchimento dessas informações é padrão para todos os usuários (Usuário: User e Senha: user123). Isso será modificado nas versões seguintes, onde os usuários terão a possibilidade de fazer um cadastro, preenchendo informações pessoais e gerando um nome de usuário e uma senha própria. Além disso, pessoas com cadastros diferentes poderão utilizar o *software* no mesmo *hardware*.

Figura 49 – Tela de login do aplicativo



Fonte: Autoria Própria (2025)

Ao inserir corretamente as informações de login o usuário é redirecionado para a interface gráfica principal do aplicativo, que consiste em uma janela de 900x600 composta de três *frames* principais (Figura 50). O *frame* 1 exibe o nome do aplicativo, o *frame* 2 consiste no menu de opções, possuindo 7 botões com diferentes funcionalidades. Já o *frame* 3, inicialmente, não exibe nenhum conteúdo, porém, é o único que varia durante o uso do aplicativo, conforme será apresentado posteriormente.

Figura 50 – Tela principal do aplicativo

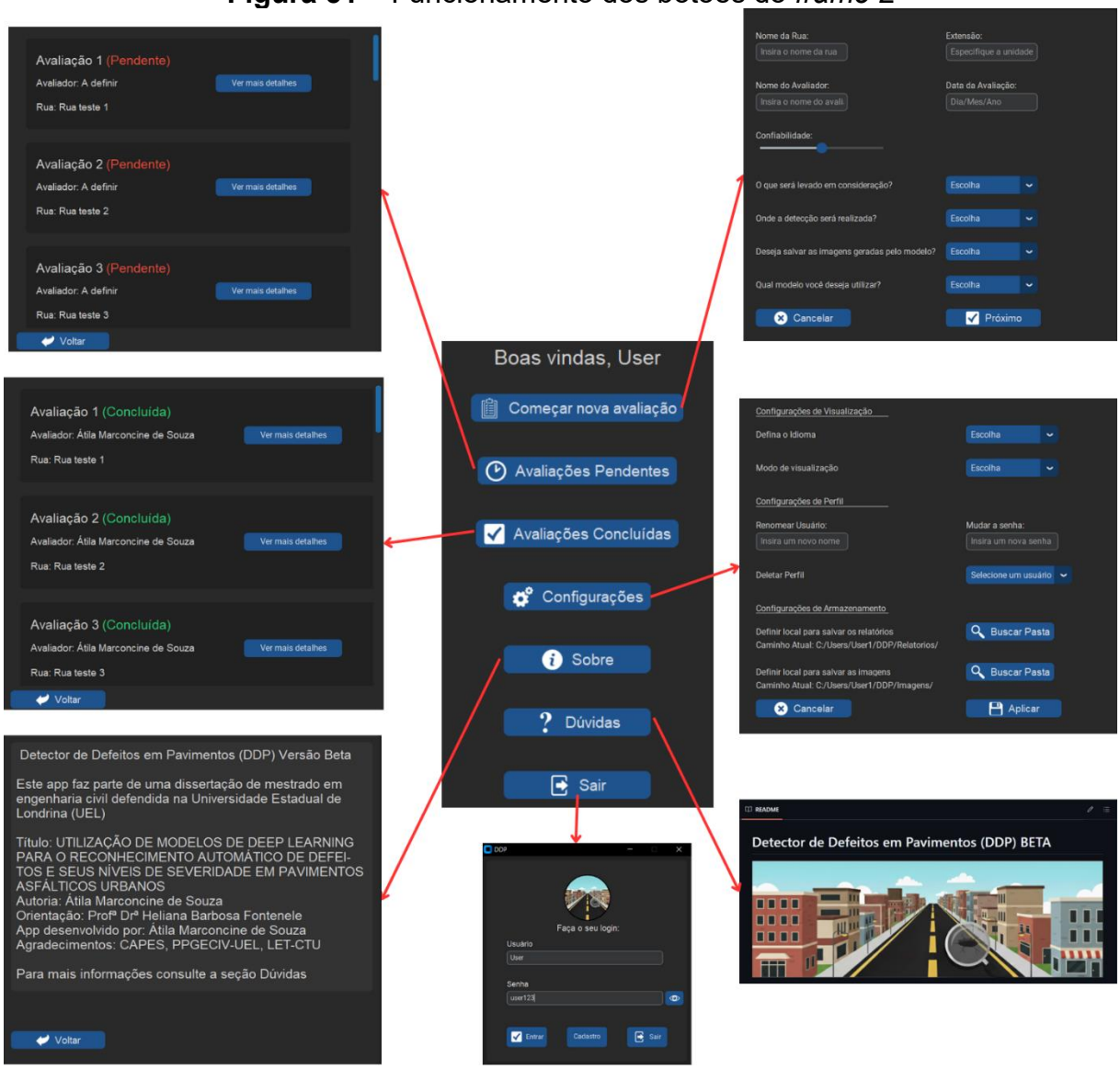
Fonte: Autoria Própria (2025)

Na Figura 50, é possível observar que o *frame* 2 possui 7 botões. O botão B1 é responsável por iniciar o procedimento de avaliação no *frame* 3. Os botões B2 e B3 são responsáveis por exibir, respectivamente, as avaliações pendentes e as avaliações concluídas. O botão B4, apresenta diversas configurações que o usuário pode modificar.

Ressalta-se que na presente versão, as abas acessadas através dos botões B2, B3 e B4 possuem suas respectivas interfaces gráficas definidas. Contudo, suas funcionalidades de processamento e armazenamento de informações ainda estão em fase de desenvolvimento. Em versões futuras do aplicativo, a biblioteca de banco de dados local SQlite3 será implementada, permitindo persistência de dados necessária para armazenar localmente as preferências de configuração do usuário, bem como gerenciar as avaliações completas e as avaliações pendentes.

Os demais botões estão configurados totalmente, na qual o botão B5, exibe no *frame* 3 uma breve descrição do aplicativo, o botão B6 redireciona o usuário para o GitHub e o botão B7 volta novamente para tela de login. A Figura 51 indica o que ocorre ao pressionar cada um dos sete botões.

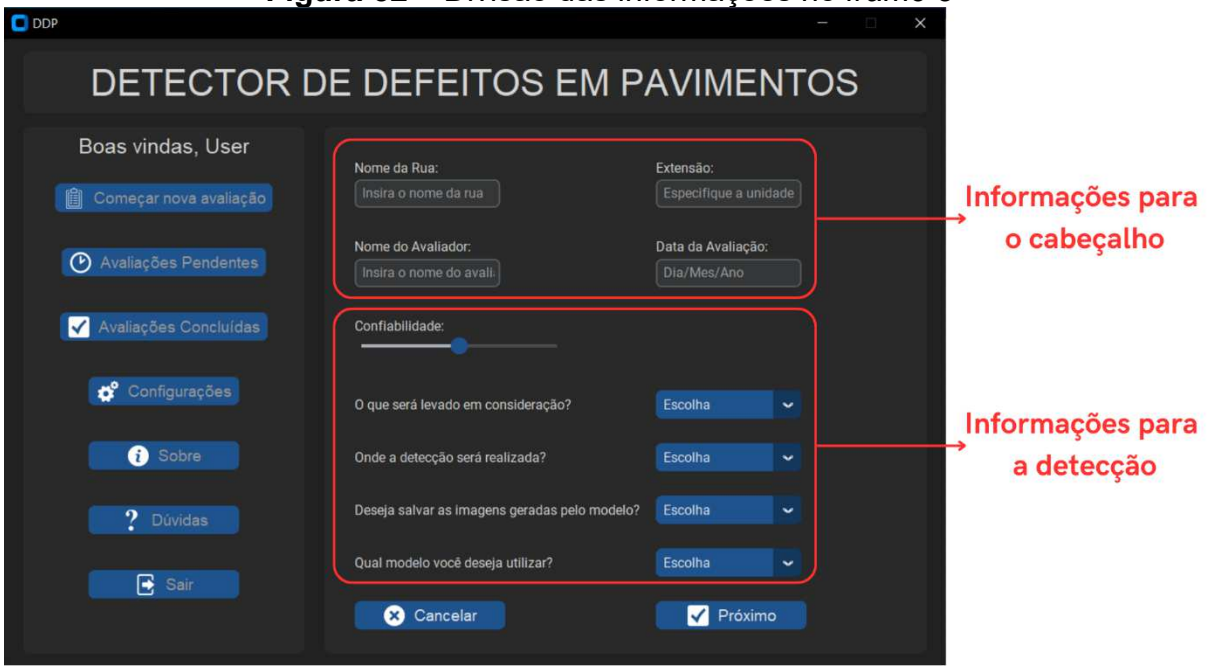
Figura 51 – Funcionamento dos botões do *frame* 2



Fonte: Autoria Própria (2025)

O botão B1 é responsável por iniciar uma nova avaliação. Quando clicado, os botões do *frame* 2 são bloqueados e o *frame* 3 exibe um conjunto de informações a serem preenchidas pelo usuário. Essas informações podem ser divididas em informações destinadas para o cabeçalho (que serão armazenadas e utilizadas na confecção do relatório) e informações destinadas para a detecção. Na Figura 52 é possível visualizar essa divisão de informações.

Figura 52 – Divisão das informações no *frame* 3



Fonte: Autoria Própria (2025)

As informações de cabeçalho consistem no preenchimento de informações através de entradas (*Entry*), onde o usuário deve digitar o nome da rua, sua extensão, o nome do avaliador e a data de avaliação. Já as informações de detecção são compostas por uma barra deslizante (*Slider*) e 4 menus de opções (*Option Menu*). Cada um desses elementos são explicados a seguir:

- **Confiabilidade**: Barra deslizante que varia de 0 a 1. O modelo irá mudar as detecções exibidas de acordo com o valor escolhido (*e.g.* se o valor selecionado for 0.8, apenas detecções com 80% de confiança serão exibidas). O valor padrão de confiabilidade é 0.5.
- **Menu de Opções 1:** Local para selecionar o tipo de detecção. O usuário deve escolher se o modelo levará em consideração apenas os defeitos ou os defeitos e seus níveis de severidade.
- **Menu de Opções 2:** Menu para escolher o arquivo em que será realizada a detecção. Na presente versão, apenas a detecção em imagens (.jpg, .png, .jpeg) está disponível, sendo possível efetuá-la de duas formas: manual e automática. Essas duas formas de detecção serão abordadas na sequência.
- **Menu de Opções 3:** Neste menu o usuário tem a opção de escolher se deseja salvar no computador as imagens com as *bounding box*

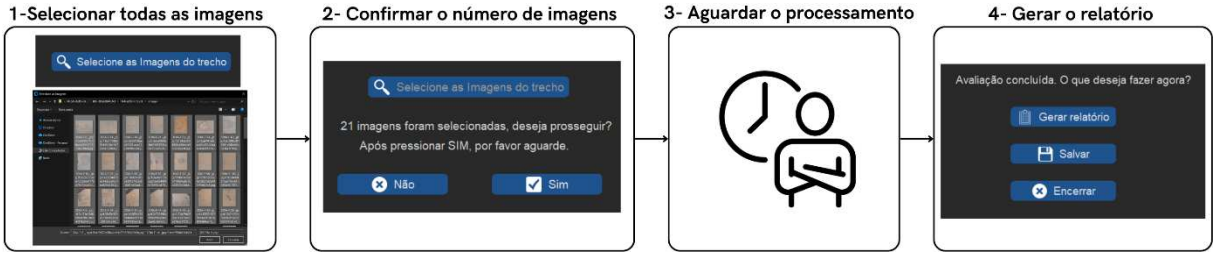
detectadas pelo modelo;

- **Menu de Opções 4:** Por fim, o quarto menu de opções consiste na seleção do modelo a ser utilizado na avaliação. No momento o DDP disponibiliza apenas os modelos YOLOv12 e YOLOv11. Ressalta-se que esses modelos foram os que obtiveram as melhores métricas de avaliação na etapa de treinamento.

Dentre as informações necessárias para prosseguir com a avaliação, a seleção do arquivo em que a detecção será efetuada (Menu de Opções 2) é a única informação que redireciona o usuário para modos diferentes: o modo manual e o modo automático.

No modo automático, o usuário seleciona todas as imagens do trecho que deseja avaliar. O modelo detecta os defeitos presentes e finaliza a avaliação automaticamente. A desvantagem desse modo, é que o modelo irá incluir na avaliação tanto a detecções corretas, quanto as incorretas (*i.e.* FP e FN) O fluxo dev funcionamento deste modo é mostrado na Figura 53.

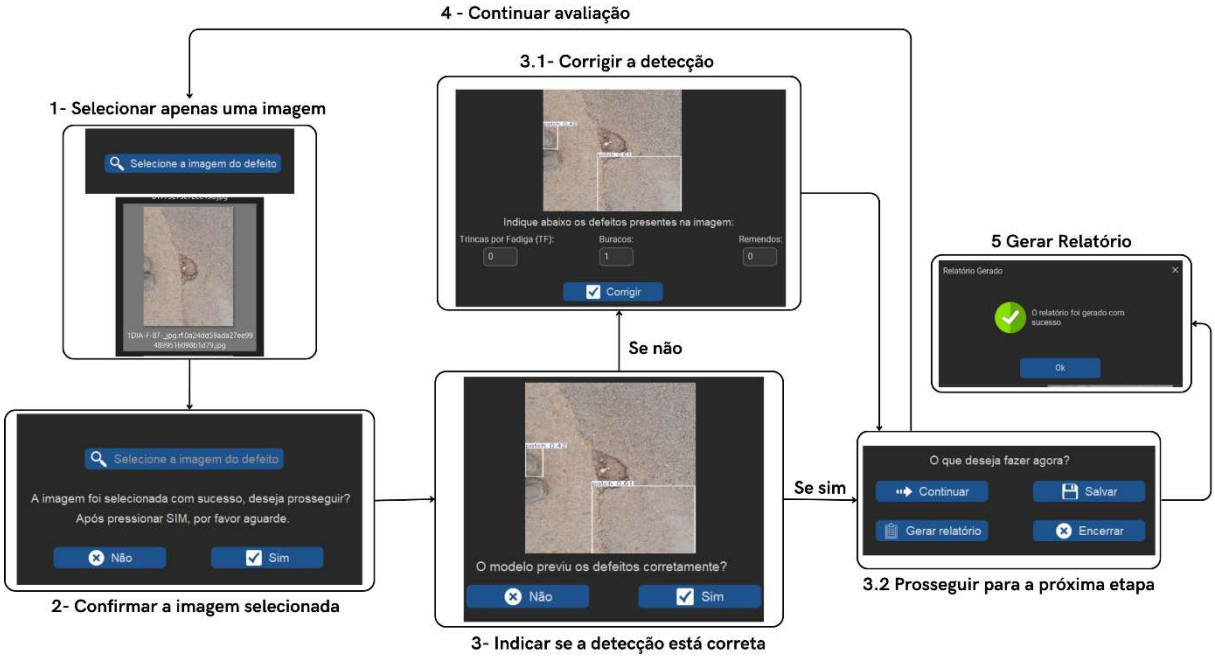
Figura 53 – Funcionamento do modo automático



Fonte: Autoria Própria (2025)

Já no modo manual, o usuário possui um maior controle da avaliação, pois caso o modelo erre a detecção na imagem selecionada, é possível efetuar uma correção manual, evitando assim a presença de FP e FN. Para isso, apenas uma imagem é avaliada por vez. É possível observar mais detalhes do funcionamento deste modo na Figura 54.

Figura 54 – Funcionamento do modo manual

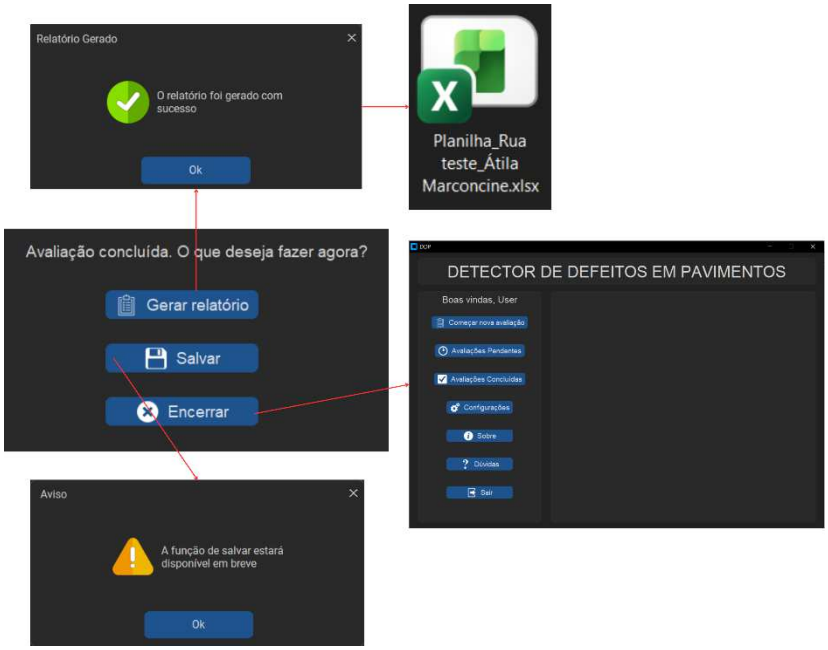


Fonte: Autoria Própria (2025)

É importante ressaltar que a correção manual realizada pelo usuário é pontual e não implica em nenhuma melhoria de aprendizagem do modelo. Ou seja, se a mesma imagem com detecções errôneas for submetida novamente, o modelo selecionado cometerá os mesmos erros.

Ao finalizar a avaliação automática o *frame* 3 exibe três botões: Gerar relatório, salvar e encerrar (vide Figura 53). Já o modo manual apresenta um quarto botão, intitulado continuar, que redireciona o usuário para inserir uma nova imagem (vide Figura 54). O funcionamento dos demais botões é possível ser observado na Figura 55.

Figura 55 – Funcionamento dos botões disponíveis após encerramento da avaliação

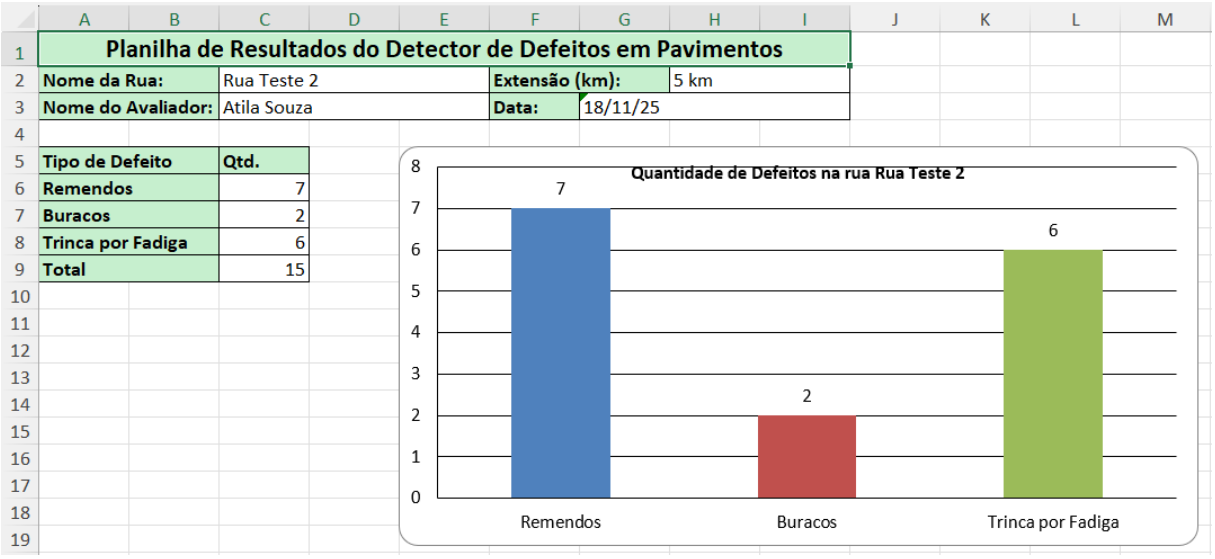


Fonte: Autoria Própria (2025)

Conforme a Figura 55, ao clicar no botão de salvar, um aviso é exibido indicando a indisponibilidade desta função na versão atual. Futuramente esse botão servirá para armazenar as avaliação pendentes e concluídas, que poderão ser acessadas através dos botões B4 e B5 do *frame* 2. O botão de encerrar, por sua vez, é responsável por remover o conteúdo do *frame* 3, finalizando a avaliação em andamento e possibilitando clicar novamente nos botões do *frame* 2. Por fim, ao clicar no botão de gerar relatório um aviso é exibido confirmando a confecção do arquivo da avaliação.

O arquivo gerado consiste em uma planilha de excel (.xlsx) que pode ser encontrada na pasta em que o programa foi instalado. Na planilha é exibido um cabeçalho com as informações inseridas anteriormente no *frame* 3, uma tabela com o número de defeitos detectados pelo modelo e um gráfico de barra confeccionado a partir da tabela. Um exemplo de planilha pode ser visualizado na Figura 56.

Figura 56 – Exemplo de planilha gerada *(output)*



Fonte: Autoria Própria (2025)

Caso o usuário tenha selecionado a opção de reconhecer os defeitos e o nível de severidade no menu de opções 1, além das informações relativas aos defeitos também serão exibidas informações dos níveis de severidade, adicionado assim no arquivo mais uma tabela e mais um gráfico em barra. Um exemplo de planilha gerada para defeitos e níveis de severidade pode ser observada na Figura 57.

Figura 57 – Exemplo de planilha gerada para defeitos e severidades

Fonte: Autoria Própria (2025)

É importante enfatizar a importância desse tipo de *software* na avaliação de pavimentos e na gerência de pavimentos de forma geral. O DDP além de empregar modelos de *deep learning* para acelerar o processo de reconhecimento de defeito e determinação do seu nível de severidade, possui uma interface gráfica simples e intuitiva, podendo ser utilizada por qualquer pessoa sem a necessidade de

conhecimento prévio em programação. Ademais, por ser um aplicativo de *desktop*, grande parte da avaliação (com exceção da coleta da imagens) pode ser feita em escritório, eliminando assim a exposição dos avaliadores ao tráfego de veículos e condições climáticas adversas.

No Brasil, existem outros *softwares* que utilizam algoritmos de *deep learning* para auxiliar na detecção de defeitos em pavimentos, como o Highway AI, desenvolvido por Freitas (2022), o DNIT-ICM, proposto pelo DNIT (2023), e o Pothole Web App, elaborado por Barbosa, Rosa e Rieder (2024).

Destaca-se que, dentre os aplicativos mencionados, apenas o DDP leva em consideração o nível de severidade dos defeitos e possui código aberto de fácil acesso via Github. Porém, comparado às aplicações existentes, o DDP possui três limitações principais:

- **Local de Uso:** Por ter sido desenvolvido com a biblioteca CustomTkinter, o DDP é um software exclusivo de *desktop*, inviabilizando seu uso em dispositivos móveis (*e.g.* smartphones e tablets). Ademais, o *software* foi compilado e empacotado especificamente para o ambiente Windows (64 bits), sendo incompatível com os sistemas operacionais Linux ou macOS.
- **Utilização de mapas:** A utilização de mapas é uma alternativa interessante para indicar o local em que o defeito detectado pelo modelo está localizado. Porém, o DDP não possui esta função, sendo o nome da rua (no cabeçalho) a única indicação geográfica no momento.
- **Detecção em vídeos:** Os modelos empregados no DDP foram configurados apenas para o reconhecimento de defeitos em imagens, não sendo possível a utilização de vídeos, que são mais rápidos e práticos de adquirir.

Por fim, ressalta-se que todas as limitações listadas acima serão contornadas em versões futuras do *software*.

7 CONCLUSÕES

Com base na aplicação do *framework* proposto e no exposto ao longo deste trabalho, é possível constatar que todos os modelos empregados apresentam um bom desempenho no reconhecimento de defeitos presentes em pavimentos asfálticos urbanos e seus níveis de severidade.

No BD1, os melhores resultados de treinamento ocorreram no RF-DETR-Base, obtendo um mAP50 de 90,3% e um mAP50-95 de 65,4%, demonstrando superioridade em relação as arquiteturas YOLO mais recentes. Já no BD2, quando o nível de severidade é levado em consideração, todas as métricas de avaliação apresentam valores inferiores em relação ao BD1. Contudo, apesar disso, os resultados foram satisfatórios, com destaque para o YOLOv12s, que alcançou o maior mAP50 (65,2%) e o maior recall (65,5%).

Na análise dos resultados de treinamento por classes, verifica-se que no BD1, os maiores valores de AP50 ocorreram nas classes TPVs e buracos, atingindo respectivamente, 100% e 98,1%. Já os demais defeitos obtiveram resultados inferiores, porém satisfatórios, com o remendo alcançando o AP50 de 88,1% e a trinca por fadiga o AP50 de 76,4%.

Já no BD2, foi constatado um padrão nos resultados dos defeitos trinca por fadiga e remendo em todos os modelos: o nível de severidade baixa apresenta maior AP50 em relação aos demais níveis. Para esses dois defeitos, a determinação do nível de severidade tende a ser mais visual e subjetiva, uma vez que suas propriedades geométricas não são levadas em consideração. Esse fato ocasiona inconsistências no processo de anotação e, consequentemente, estas impactam negativamente o aprendizado dos modelos. A união das severidades média e alta em uma única classe, através de um terceiro banco de dados, pode contornar esse problema.

Em contrapartida, para o defeito buraco, o comportamento exibido no BD2 é diferente das demais classes. Para esse defeito os modelos não apresentaram um padrão de resultados, uma vez que que para determinação do seu nível de severidade são necessárias a área e a profundidade, informações que não foram fornecidas aos modelos. Além disso, a profundidade é uma informação geométrica da terceira dimensão, e os modelos empregados foram treinados com imagens 2D. Dessa forma, faz-se necessário o uso de outras metodologias como o LiDAR, a fotogrametria e/ou algoritmos de *monocular depth estimation* para a determinação correta do nível de

severidade deste defeito.

Ao levar o tempo de treinamento em consideração, constatou-se que no BD1 o YOLOv11s obteve a melhor eficiência quando treinado no ambiente local, enquanto no ambiente online o melhor desempenho ocorreu no YOLOv11n. Já no BD2, os resultados mais expressivos no ambiente local e ambiente online ocorreram respectivamente nos modelos YOLOv12s e RF-DETR-Base.

Ao comparar o ambiente online com o ambiente local, é perceptível que o ambiente local é mais vantajoso e recomendado para utilização, uma vez que o ambiente online possui diversas limitações, dentre elas: impossibilidade de utilizar modelos robustos, visualização incompleta de resultados, indisponibilidade de modelos mais antigos, ausência de personalização de modelos e modificação de hiper parâmetros.

Devido às limitações mencionadas, a etapa de teste foi conduzida apenas no ambiente local, através dos modelos com melhor desempenho, ou seja, YOLOv11s e YOLOv12s. Nessa etapa, constata-se a superioridade do modelo YOLOv12s, que obteve o maior equilíbrio entre métricas no BD1 (86,26% de precisão, 86,24% de recall, 86,25% de f1-score) e o melhor desempenho no BD2 (69,04% de precisão, 51,85% de recall e 55,18% de f1-score).

Na etapa de validação prática, a técnica SAHI em conjunto com o modelo YOLOv12s apresenta potencial para o reconhecimento de defeitos em ortomosaicos. Porém, mais experimentos necessitam ser conduzidos, uma vez que o tópico em questão é pouco explorado na literatura e os resultados obtidos não foram satisfatórios.

Por fim, o aplicativo proposto, o Detector de Defeitos em Pavimentos (DDP), apresenta-se como uma ferramenta importante na gerência de pavimentos, permitindo que a avaliação da malha viária seja efetuada de forma automática, rápida, simples e eficaz, através de uma interface gráfica intuitiva. Em relação a outras aplicações existentes, o DDP possui algumas limitações que serão implementadas em versões futuras.

REFERÊNCIAS

AFAQ, Saahil; RAO, Smitha. Significance Of Epochs On Training A Neural Network. **International Journal of Scientific & Technology Research**, [*s. l.*], vol. 9, no. 06, p. 1–4, 2020.

AKYON, Fatih Cagatay; ALTINUC, Sinan Onur; TEMIZEL, Alptekin. Slicing Aided Hyper Inference and Fine-Tuning for Small Object Detection. *In*: , 2022. **2022 IEEE International Conference on Image Processing (ICIP)**. [*S. l.*]: IEEE, 2022. p. 966–970.

ALBUQUERQUE, Tairoe Paz e. **Índice de Condição Baseado em Defeitos Superficiais para Gerência de Pavimentos Urbanos**. 2017. 140 f. Dissertação - Universidade Federal da Paraíba, João Pessoa, 2017.

ALI, Luqman *et al.* Bibliometric Analysis and Review of Deep Learning-Based Crack Detection Literature Published between 2010 and 2022. **Buildings**, [*s. l.*], vol. 12, no. 4, 2022.

ALOM, Md Zahangir *et al.* A state-of-the-art survey on deep learning theory and architectures. **Electronics (Switzerland)**, [*s. l.*], vol. 8, no. 3, 2019.

ALQETHAMI, Sara *et al.* RoadNet: Efficient Model to Detect and Classify Road Damages. **Applied Sciences (Switzerland)**, [*s. l.*], vol. 12, no. 22, p. 1–15, 2022.

ASTM D6433. **Standard Practice for Roads and Parking Lots Pavement Condition Index Surveys**. [*S. l.*]: ASTM International, 2024.

BADRINARAYANAN, Vijay; KENDALL, Alex; CIPOLLA, Roberto. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *In*: , 2017. **IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI 2017)**. [*S. l.: s. n.*], 2017. p. 2481–2495.

BARBOSA, Jean Cássio Peres; ROSA, Francisco Dalla; RIEDER, Rafael. Pothole Detection Web App: uma abordagem para detecção de buracos em pavimentos asfálticos utilizando YOLO. **Revista Brasileira de Computação Aplicada**, [*s. l.*], vol. 16, no. 3, p. 25–36, 2024.

BENTO, Carlos Henrique Rodrigues *et al.* Avaliação do uso do LIDAR em smarthphones para classificação de buracos. *In*: , 2025, Goiânia, Goiás, Brasil. **39º ANPET - Congresso de Pesquisa e Ensino em Transportes**. Goiânia, Goiás, Brasil: [*s. n.*], 2025. p. 1–5.

BUSLAEV, Alexander *et al.* Albumentations: Fast and flexible image augmentations. **Information (Switzerland)**, [*s. l.*], vol. 11, no. 2, p. 1–20, 2020.

CAREY JR, W. N.; IRICK, P. E. The Pavement Serviceability - Performance Concept. **Highway Research Board Bulletin**, [*s. l.*], no. 250, p. 40–58, 1960.

CARION, Nicolas *et al.* **End-to-End Object Detection with Transformers**. [*S. l.*], 2020. Available at: https://arxiv.org/abs/2005.12872v3. Accessed at: 6 Jun. 2025.

CHEN, Kai *et al.* Hybrid task cascade for instance segmentation. *In*: , 2019. **IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2019)**. [*S. l.: s. n.*], 2019. p. 4969–4978.

CHEN, Qiang *et al.* **LW-DETR: A Transformer Replacement to YOLO for Real-Time Detection**. [*S. l.*], 2024. Available at: https://arxiv.org/abs/2406.03459. Accessed at: 6 Jun. 2025.

DANIELESKI, Maria Luiza. **Proposta de metodologia para avaliação superficial de pavimentos urbanos: aplicação à rede viária de Porto Alegre**. 2004. 187 f. Dissertação - Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, 2004.

DECIAI. **YOLONAS**. [*S. l.*], 2023. Available at: https://github.com/Deci-AI/super-gradients/blob/master/YOLONAS.md. Accessed at: 28 May 2025.

DNIT. **Avaliação objetiva da superfície de pavimentos flexíveis e semi-rígidos - Procedimento**. Rio de Janeiro: DNIT, 2003.

DNIT. **Avaliação subjetiva da superfície de pavimentos flexíveis e semi-rígidos - Procedimento**. Rio de Janeiro: DNIT, 2003.

DNIT. **ICM**. [*S. l.*], 2023. Available at: https://www.gov.br/dnit/pt-br/assuntos/infraestrutura-rodoviaria/icm. Accessed at: 19 Nov. 2025.

DNIT. **Manual de gerência de pavimentos**. Rio de Janeiro: DNIT, 2011.

DNIT. **Manual De Restauração De Pavimentos Asfálticos**. Rio de Janeiro: DNIT, 2006.

DNIT. **NORMA DNIT 005/2003 - TER Defeitos: Defeitos nos pavimentos flexíveis e semi-rígidos Terminologia**. Rio de Janeiro: DNIT, 2003.

DNIT. **NORMA DNIT 154/2010: Pavimentação asfáltica-Recuperação de defeitos em pavimentos asfálticos - Especificação de serviço**. Rio de Janeiro: DNIT, 2010.

DOMINGUES, Felippe Augusto Aranha. **MID: Manual de Identificação de Defeitos de Revesttimentos Asfálticos de Pavimentos**. São Paulo: Camargo Campos, 1993.

DONG, Jiaxiu *et al.* CBAM-Optimized Automatic Segmentation and Reconstruction System for Monocular Images with Asphalt Pavement Potholes. **IEEE Transactions on Intelligent Transportation Systems**, [*s. l.*], vol. 25, no. 8, p. 10313–10330, 2024.

DONG, Qiao *et al.* Classification of pavement climatic regions through unsupervised and supervised machine learnings. **Journal of Infrastructure Preservation and Resilience**, [*s. l.*], vol. 2, no. 1, 2021.

DWYER, Brad. **When Should I Auto-Orient My Images?**. [*S. l.*], 2020. Available at: https://blog.roboflow.com/exif-auto-orientation/. Accessed at: 2 Jun. 2025.

FERRARI, Enzzo Costa *et al.* Classificação Do Defeito Buraco a Partir De Modelos 3D. *In*: , 2023, Santos, São Paulo, Brasil. **37º ANPET - Congresso de Pesquisa e**

Ensino em Transportes. Santos, São Paulo, Brasil: [*s. n.*], 2023. p. 1–12.

FREITAS, Gabriel Tavares de Melo. **SOFTWARE PARA IDENTIFICAÇÃO DE DEFEITOS NA SUPERFÍCIE DE PAVIMENTOS RODOVIÁRIOS UTILIZANDO DEEP LEARNING**. 2022. 109 f. Dissertação - Universidade Federal do Ceará, Fortaleza, 2022.

GANESHAN, D; SHARIF, M S; APEAGYEI, A. Road Deterioration detection: A Machine Learning-Based System for Automated Pavement Crack Identification and Analysis. *In*: , 2023. **2023 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies, 3ICT 2023**. [*S. l.*]: Institute of Electrical and Electronics Engineers Inc., 2023. p. 188–194.

GARCIA, Carolina. **PROCEDIMENTO DE AQUISIÇÃO DE IMAGENS DO PAVIMENTO E AVALIAÇÃO SEMIAUTOMATIZADA DE DEFEITOS (PAIPA)**. 2025. 104 f. Dissertação - Universidade Estadual de Londrina, Londrina, 2025.

GIMENES, Larissa de Abreu. **Fotogrametria à curta distância com uso de smartphone para avaliação da condição de pavimentos flexíveis urbanos**. 2025. 120 f. Dissertação - Universidade Estadual de Londrina, Londrina, 2025.

GIRSHICK, Ross. Fast R-CNN. *In*: , 2015. **IEEE International Conference on Computer Vision (ICCV 2015)**. [*S. l.: s. n.*], 2015. p. 1440–1448.

GIRSHICK, Ross *et al.* Rich feature hierarchies for accurate object detection and semantic segmentation. *In*: , 2014. **IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2014)**. [*S. l.: s. n.*], 2014. p. 580–587.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep learning**. [*S. l.*]: MIT Press, 2016.

GOPALAKRISHNAN, Kasthurirangan. Deep learning in data-driven pavement image analysis and automated distress detection: A review. **Data**, [*s. l.*], vol. 3, no. 3, 2018.

HAAS, R.; HUDSON, W. R.; ZANIEWSKI, J. **Modern Pavement Management**. Malabar: Krieger Publishing Company, 1994.

HAFIZ, Abdul Mueed; BHAT, Ghulam Mohiuddin. A survey on instance segmentation: state of the art. **International Journal of Multimedia Information Retrieval**, [*s. l.*], vol. 9, no. 3, p. 171–189, 2020.

HASSAN, M U *et al.* Predictive Maintenance of Norwegian Road Network Using Deep Learning Models. **Sensors**, [*s. l.*], vol. 23, no. 6, 2023.

HE, Kaiming *et al.* Deep residual learning for image recognition. *In*: , 2015. **IEEE Computer Society Conference on Computer Vision and Pattern Recognition**. [*S. l.: s. n.*], 2015. p. 770–778.

HE, Kaiming *et al.* Mask R-CNN. *In*: , 2017. **2017 IEEE International Conference on Computer Vision (ICCV)**. [*S. l.*]: IEEE, 2017. p. 2980–2988.

HINTON, G. E.; SALAKHUTDINOV, R. R. **Reducing the Dimensionality of Data with Neural NetworksSCIENCE**. [*S. l.: s. n.*], 2006.

HSIEH, Yung-An; YANG, Zhongyu; TSAI, Yi-Chang James. Convolutional neural network for automated classification of jointed plain concrete pavement conditions. **COMPUTER-AIDED CIVIL AND INFRASTRUCTURE ENGINEERING**, [*s. l.*], vol. 36, no. 11, SI, p. 1382–1397, 2021.

IBRAHIM, A *et al.* Flexible Pavement Crack's Severity Identification and Classification using Deep Convolution Neural Network. **Journal of Mechanical Engineering**, [*s. l.*], vol. 18, no. 2, p. 193–201, 2021.

JAFARI, Fereshteh; MORADI, Kamran; SHAFIEE, Qobad. Shallow Learning vs. Deep Learning in in Engineering Applications. *In*: ERTUGRUL, Omer Faruk; GUERRERO, Josep M.; YILMAZ, Musa (eds.). **Shallow Learning vs. Deep Learning: a pratical guide for machine learning**. [*S. l.*]: SPRINGER, 2024. p. 29–76.

JEGHAM, Nidhal *et al.* **YOLO Evolution: A Comprehensive Benchmark and Architectural Review of YOLOv12, YOLOv11 and Their Previous Versions**. [*S. l.*], 2025. Available at: http://arxiv.org/abs/2411.00201. Accessed at: 26 May 2025.

JOHNSON, Justin. **Lecture 1: Introduction to Deep Learning for Computer Vision**. [Vídeo]: Michigan Online, 2019.

KANG, Junhyung *et al.* A Survey of Deep Learning-Based Object Detection Methods and Datasets for Overhead Imagery. **IEEE Access**, [*s. l.*], vol. 10, p. 20118–20134, 2022.

KHANAM, Rahima; HUSSAIN, Muhammad. **YOLOv11: An Overview of the Key Architectural Enhancements**. [*S. l.*], 2024. Available at: http://arxiv.org/abs/2410.17725. Accessed at: 28 May 2025.

KRISHNA, Sajja Tulasi; KALLURI, Hemantha Kumar. Deep learning and transfer learning approaches for image classification. **International Journal of Recent Technology and Engineering**, [*s. l.*], vol. 7, no. 5, p. 427–432, 2019.

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. *In*: , 2012. **25th Conference on Neural Information Processing Systems (NeurIPS 2012)**. [*S. l.: s. n.*], 2012. p. 1097–1105.

KUMAR, Yogesh; KUMAR, Pankaj. Comparative study of YOLOv8 and YOLO-NAS for agriculture application. *In*: , 2024. **Proceedings - 11th International Conference on Signal Processing and Integrated Networks, SPIN 2024**. [*S. l.*]: IEEE, 2024. p. 72–77.

LECUN, Yann *et al.* Gradient-Based Learning Applied to Document Recognition. **Proceedings of the IEEE**, [*s. l.*], vol. 86, no. 11, p. 2278–2324, 1998.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **Nature**, [*s. l.*], vol. 521, no. 7553, p. 436–444, 2015.

LIN, Tsung Yi *et al.* Focal Loss for Dense Object Detection. *In*: , 2017. **IEEE**

International Conference on Computer Vision (ICCV 2017). [*S. l.: s. n.*], 2017. p. 2980–2988.

LIN, Tsung-Yi *et al.* Microsoft COCO: Common Objects in Context. *In*: LECTURE NOTES IN COMPUTER SCIENCE (INCLUDING SUBSERIES LECTURE NOTES IN ARTIFICIAL INTELLIGENCE AND LECTURE NOTES IN BIOINFORMATICS). [*S. l.: s. n.*], 2014. vol. 8693 LNCS, p. 740–755.

LIU, Wei *et al.* SSD: Single shot multibox detector. *In*: , 2016. **European Conference on Computer Vision (ECCV 2016)**. [*S. l.: s. n.*], 2016. p. 21–37.

LIU, Fangyu; LIU, Jian; WANG, Linbing. Asphalt pavement fatigue crack severity classification by infrared thermography and deep learning. **Automation in Construction**, [*s. l.*], vol. 143, p. 104575, 2022.

LIU, Shiqing; ZHANG, Haoyu; JIN, Yaochu. A survey on computationally efficient neural architecture search. **Journal of Automation and Intelligence**, [*s. l.*], vol. 1, no. 1, p. 100002, 2022.

LLOPIS-CASTELLÓ, David *et al.* Automatic Classification and Quantification of Basic Distresses on Urban Flexible Pavement through Convolutional Neural Networks. **Journal of Transportation Engineering, Part B: Pavements**, [*s. l.*], vol. 147, no. 4, p. 1–8, 2021.

LÓPEZ, Osval Antonio Montesinos; LÓPEZ, Abelardo Montesinos; CROSSA, José. **Multivariate Statistical Machine Learning Methods for Genomic Prediction**. [*S. l.*]: SPRINGER, 2022.

LOPRENCIPE, Giuseppe; PANTUSO, Antonio. A specified procedure for distress identification and assessment for urban road surfaces based on PCI. **Coatings**, [*s. l.*], vol. 7, no. 5, 2017.

MCCULLOCH, Warren S.; PITTS, Walter. A Logical Calculus of the Ideas Immanent In Nervous Activity. **Bulletin of Mathematical Biophysics**, [*s. l.*], vol. 5, p. 115–133, 1943.

MILLER, John S.; BELLINGER, Willian Y. **Distress Identification Manual for the Long-Term Pavement Performance Program**. [*S. l.: s. n.*], 2014.

NAGANANDA, Navya *et al.* Benchmarking Domain Adaptation Methods on Aerial Datasets. **Sensors**, [*s. l.*], vol. 21, no. 23, p. 8070, 2021.

NELSON, Joseph. **You Might Be Resizing Your Images Incorrectly**. [*S. l.*], 2020. Available at: https://blog.roboflow.com/you-might-be-resizing-your-images-incorrectly/. Accessed at: 2 Jun. 2025.

NIU, Zhaoyang; ZHONG, Guoqiang; YU, Hui. A review on the attention mechanism of deep learning. **Neurocomputing**, [*s. l.*], vol. 452, p. 48–62, 2021.

PÁEZ, Edgar Misael Arévalo. **Índice de condição do pavimento (ICP) para aplicação em sistemas de gerência de pavimentos urbanos**. 2015. 151 f. Dissertação - Escola de Engenharia de São Carlos, São Carlos, 2015.

PARDEDE, Johanson Pardomuan; LEKSMONO, Suryo Putrano; AJI, Ridho Bayu. Analysis of Pavement Damage on Jakarta–Merak Toll Road Using PCI Approach. **Journal of Hunan University Natural Sciences**, [*s. l.*], vol. 52, no. Volume 52, Issue 5, p. 4–14, 2025.

PERAKA, Naga Siva Pavani; BILIGIRI, Krishna Prapoorna; KALIDINDI, Satyanarayana N. Development of a Multi-Distress Detection System for Asphalt Pavements: Transfer Learning-Based Approach. **Transportation Research Record**, [*s. l.*], vol. 2675, no. 10, p. 538–553, 2021.

PERAKA, N S P; BILIGIRI, K P; KALIDINDI, S N. Framework for Pothole Detection, Quantification, and Maintenance System (PDQMS) for Smart Cities. *In*: , 2020. **Lecture Notes in Civil Engineering: Proceedings of the 9th International Conference on Maintenance and Rehabilitation of Pavements—Mairepav9**. [*S. l.*]: Springer, 2020. p. 913–922.

PRADEEP, T. *et al.* Reliability and Prediction of Embedment Depth of Sheet pile Walls Using Hybrid ANN with Optimization Techniques. **Arabian Journal for Science and Engineering**, [*s. l.*], vol. 47, no. 10, p. 12853–12871, 2022.

RAINIO, Oona; TEUHO, Jarmo; KLEN, Riku. Evaluation metrics and statistical tests for machine learning. **Scientific Reports**, [*s. l.*], vol. 14, p. 14, 2024.

RANYAL, E; SADHU, A; JAIN, K. Automated pothole condition assessment in pavement using photogrammetry-assisted convolutional neural network. **International Journal of Pavement Engineering**, [*s. l.*], vol. 24, no. 1, 2023.

RANYAL, Eshta; SADHU, Ayan; JAIN, Kamal. Enhancing pavement health assessment: An attention-based approach for accurate crack detection, measurement, and mapping. **Expert Systems with Applications**, [*s. l.*], vol. 247, 2024.

REDMON, Joseph *et al.* You Only Look Once: Unified, Real-Time, Object Detection. *In*: , 2016. **Proceedings ofthe IEEE conference on computer vision and pattern recognition**. [*S. l.: s. n.*], 2016. p. 779–788.

REDMON, Joseph; FARHADI. **YOLOv3: An Incremental Improvement**. [*S. l.*], 2018. Available at: https://arxiv.org/abs/1804.02767. Accessed at: 19 Mar. 2025.

REN, Shaoqing *et al.* Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *In*: , 2015. **Advances in Neural Information Processing Systems 28 (NIPS 2015)**. [*S. l.: s. n.*], 2015. p. 91–99.

ROBERTS, R; INZERILLO, L; MINO, G Di. Exploiting low-cost 3D imagery for the purposes of detecting and analyzing pavement distresses. **Infrastructures**, [*s. l.*], vol. 5, no. 1, 2020.

ROBICHEAUX, Peter *et al.* **RF-DETR: A SOTA Real-Time Object Detection Model**. [*S. l.*], 2025. Available at: https://blog.roboflow.com/rf-detr/. Accessed at: 6 Jun. 2025.

RONNEBERGER, Olaf; FISCHER, Philipp; BROX, Thomas. U-Net: Convolutional Networks for Biomedical Image Segmentation. *In*: , 2015. **Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015)**. [*S. l.: s. n.*], 2015. p. 234–241.

RUSSELL, Stuart; NORVIG, Peter. **Artificial Intelligence: A Modern Approach**. 4. ed. Hoboken: Pearson, 2021.

SALVIATTO, Vitor Hugo. **Ferramenta de Avaliação da Condição de Pavimentos Flexíveis Urbanos Baseada em uma Análise Multicritério**. 2021. 122 f. Dissertação - Universidade Estadual de Londrina, Londrina, 2021.

SARKER, Iqbal H. Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions. **SN Computer Science**, [*s. l.*], vol. 2, no. 6, p. 1–20, 2021.

SARKER, Iqbal H. Machine Learning: Algorithms, Real-World Applications and Research Directions. **SN Computer Science**, [*s. l.*], vol. 2, no. 3, p. 1–21, 2021.

SHAO, Shuai *et al.* Objects365: A Large-Scale, High-Quality Dataset for Object Detection. *In*: , 2019. **2019 IEEE/CVF International Conference on Computer Vision (ICCV)**. [*S. l.*]: IEEE, 2019. p. 8429–8438.

SHELHAMER, Evan; LONG, Jonathan; DARRELL, Trevor. Fully Convolutional Networks for Semantic Segmentation. **IEEE Conference on Computer Vision and Pattern Recognition (CVPT 2015)**, [*s. l.*], p. 3431–3440, 2015.

SHOLEVAR, Nima; GOLROO, Amir; ESFAHANI, Sahand Roghani. Machine learning techniques for pavement condition evaluation. **Automation in Construction**, [*s. l.*], vol. 136, no. March, p. 104190, 2022.

SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. *In*: , 2015. **3rd International Conference on Learning Representations, ICLR 2015**. [*S. l.: s. n.*], 2015. p. 1–14.

SONG, Weidong *et al.* Automatic Pavement Crack Detection and Classification Using Multiscale Feature Attention Network. **IEEE ACCESS**, [*s. l.*], vol. 7, p. 171001–171012, 2019.

SOUZA, Átila Marconcine de; CESTARI, Vinicius Fier; FONTENELE, Heliana Barbosa. Classification of pothole distress severity in asphalt pavements using YOLOv8. **DYNA**, [*s. l.*], vol. 92, no. 238, p. 47–56, 2025.

SOUZA, Átila Marconcine de; FONTENELE, Heliana Barbosa. **Determination of Pavement Distress Severity Using Machine Learning: A Systematic Review**. Transportation Research Record: Submetido, 2025.

SPENCER, Billie F.; HOSKERE, Vedhus; NARAZAKI, Yasutaka. Advances in Computer Vision-Based Civil Infrastructure Inspection and Monitoring. **Engineering**, [*s. l.*], vol. 5, no. 2, p. 199–222, 2019.

TERVEN, Juan; CÓRDOVA-ESPARZA, Diana Margarita; ROMERO-GONZÁLEZ, Julio Alejandro. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. **Machine Learning and Knowledge Extraction**, [*s. l.*], vol. 5, no. 4, p. 1680–1716, 2023.

TIAN, Yunjie; YE, Qixiang; DOERMANN, David. **YOLOv12: Attention-Centric Real-**

Time Object Detectors. [*S. l.*], 2025. Available at: https://arxiv.org/abs/2502.12524. Accessed at: 26 May 2025.

TORRALBA, Antonio; ISOLA, Phillip; FREEMAN, William T. **Foundations of Computer Vision**. [*S. l.*]: MIT Press, 2024.

TRAN, Thai Son *et al.* A two-step sequential automated crack detection and severity classification process for asphalt pavements. **International Journal of Pavement Engineering**, [*s. l.*], vol. 23, no. 6, p. 2019–2033, 2020.

TRAN, Van Phuc *et al.* One stage detector (RetinaNet)-based crack detection for asphalt pavements considering pavement distresses and surface objects. **JOURNAL OF CIVIL STRUCTURAL HEALTH MONITORING**, TIERGARTENSTRASSE 17, D-69121 HEIDELBERG, GERMANY, vol. 11, no. 1, p. 205–222, 2021.

TREVOR, Hanson *et al.* Using Remote Sensing to Measure Pavement Surface Condition and Distress: A New Brunswick Pilot Test. *In*: , 2024. **Proceedings ofthe Canadian Society for Civil Engineering Annual Conference 2023**. [*S. l.: s. n.*], 2024. p. 1–14.

ULTRALYTICS. **Ultralytics YOLOv11**. [*S. l.*], 2024. Available at: https://docs.ultralytics.com/models/yolo11/. Accessed at: 9 May 2025.

ULTRALYTICS. **Ultralytics YOLOv8**. [*S. l.*], 2023. Available at: https://docs.ultralytics.com/models/yolov8/. Accessed at: 9 May 2025.

VALIPOUR, Parisa Setayesh *et al.* Automatic pavement distress severity detection using deep learning. **Road Materials and Pavement Design**, [*s. l.*], vol. 25, no. 8, p. 1830–1846, 2023.

VERBRAEKEN, Joost *et al.* A Survey on Distributed Machine Learning. **ACM Computing Surveys**, [*s. l.*], vol. 53, no. 2, 2020.

WANG, Chien-Yao; YEH, I-Hau; LIAO, Hong-Yuan Mark. **YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information**. [*S. l.*], 2024. Available at: https://arxiv.org/abs/2402.13616. Accessed at: 26 May 2025.

WEI, Jing *et al.* Machine learning in materials science. **InfoMat**, [*s. l.*], vol. 1, no. 3, p. 338–358, 2019.

XU, Yayin *et al.* Machine learning in construction: From shallow to deep learning. **Developments in the Built Environment**, [*s. l.*], vol. 6, no. April 2020, p. 100045, 2021.

ZANCHETTA, Fábio. **Sistema De Gerência De Pavimentos Urbanos: Avaliação De Campo, Modelo De Desempenho E Análise Econômica**. 2017. 234 f. Tese de Doutorado - Universidade de São Paulo, São Carlos, 2017.

ZHANG, Hao *et al.* **DINO : DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection**. [*S. l.*], 2022. Available at: https://arxiv.org/abs/2203.03605. Accessed at: 6 Jun. 2025.

ZHAO, Xia *et al.* A review of convolutional neural networks in computer vision. **Artificial Intelligence Review**, [*s. l.*], vol. 57, no. 4, p. 1–43, 2024.

ZHAO, Yian *et al.* **DETRs Beat YOLOs on Real-time Object Detection**. [*S. l.*], 2024. Available at: https://arxiv.org/abs/2304.08069. Accessed at: 6 Jun. 2025.

ZHENG, Lele *et al.* Deep learning-based intelligent detection of pavement distress. **Automation in Construction**, [*s. l.*], vol. 168, no. PA, p. 105772, 2024.

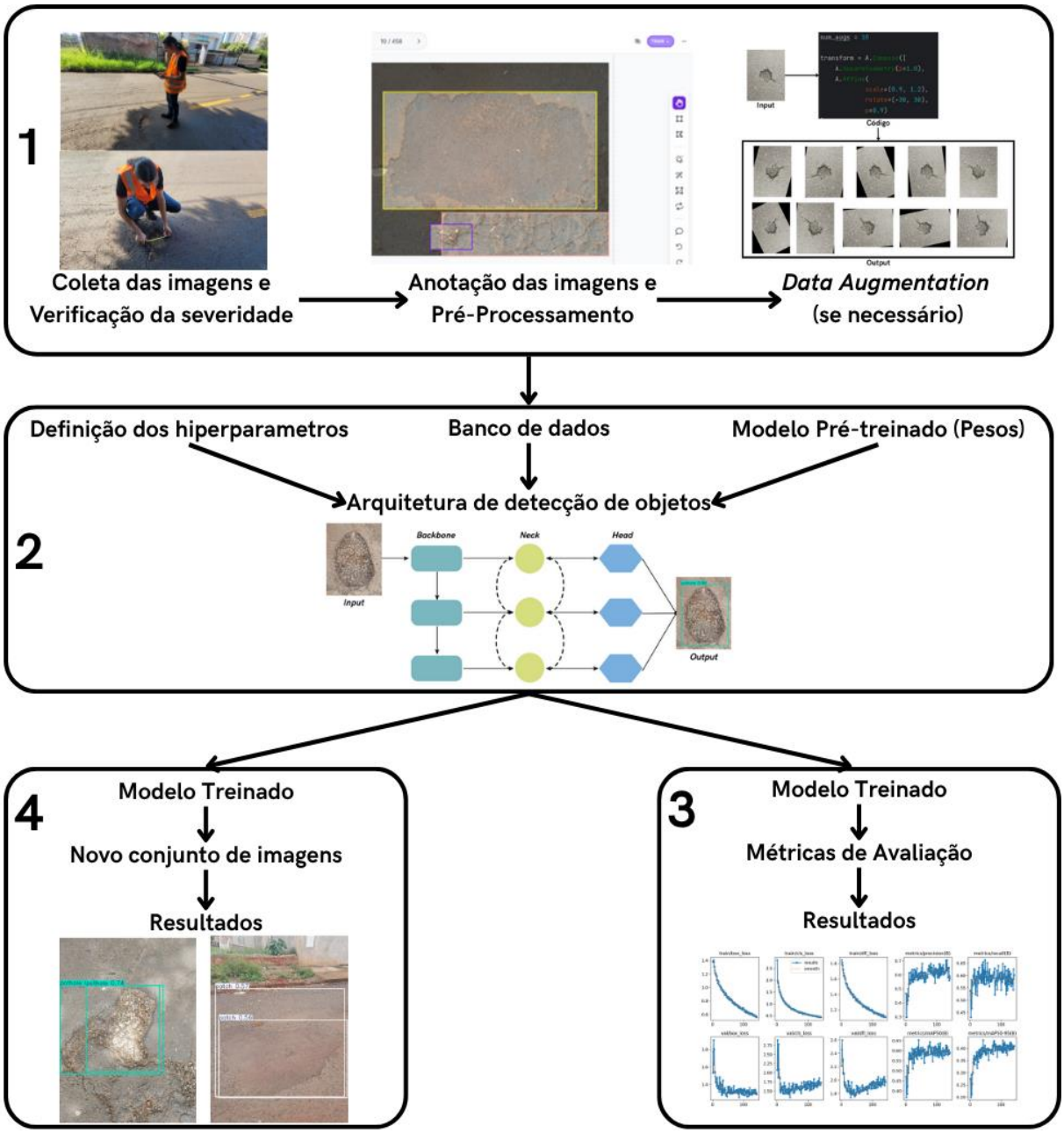
ZHONG, Guoqiang; LING, Xiao; WANG, Li Na. From shallow feature learning to deep learning: Benefits from the width and depth of deep architectures. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, [*s. l.*], vol. 9, no. 1, p. 1–14, 2019.

ZHU, Xizhou *et al.* Deformable DETR: Deformable Transformes for End-to-End Object Detection. *In*: , 2021. **ICLR 2021 - The Ninth International Conference on Learning Representations**. [*S. l.: s. n.*], 2021. p. 1–16.

APÊNDICES

APÊNDICE A

Framework ilustrado do método

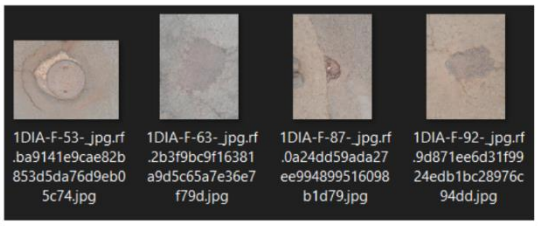


Legenda: O número 1 indica a etapa de confecção do banco de dados, onde é necessário efetuar a coleta e a determinação do nível de severidade em campo (bancos de dados disponíveis online também podem ser utilizado). Com as imagens coletadas, efetuam-se as etapas de anotação, pré-processamento e expansão do banco de dados (caso necessário); O número 2 corresponde a etapa de treinamento, na qual hiperparâmetros são definidos para o banco de dados confeccionado. Pesos de modelos pré-treinados também podem ser utilizados (transferência de aprendizagem); com o modelo treinado é possível efetuar as etapas 3 (validação) e 4 (teste). Na etapa de validação métricas de avaliação são empregadas para verificar o desempenho do modelo. Por fim, na etapa de teste, o modelo é submetido a um novo conjunto de imagens não utilizadas na etapa de treinamento. Para o teste também podem ser empregados vídeos e ortomosaicos.

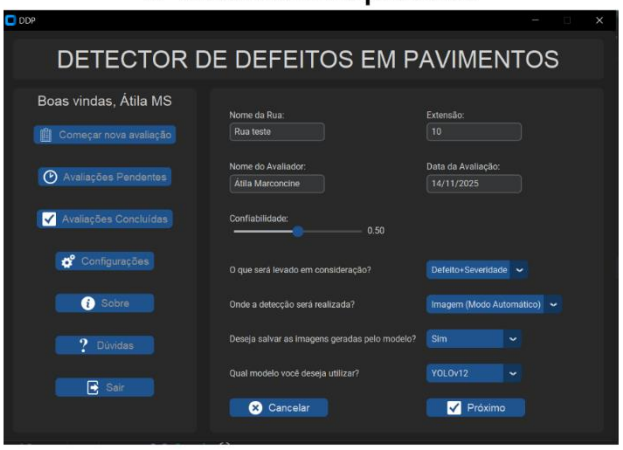
APÊNDICE B

Framework ilustrado do aplicativo

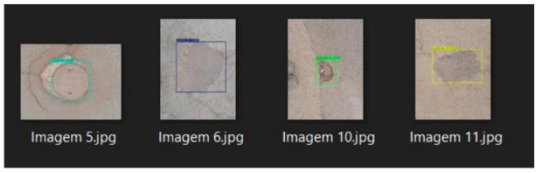
1- *Input*:
Imagens de pavimentos com defeitos



2- Interface do aplicativo



3- Detecção dos defeitos e níveis de severidade



4- *Output*:
Planilha com Tabelas e Gráficos

Planilha de Resultados do Detector de Defeitos em Pavimentos			
Nome da Rua:	Rua teste	Extensão (km):	10
Nome do Avaliador:	Átila Marconcine	Data:	14/11/2025

Tipo de Defeito	Severidade	Qtd.
Trinca por Fadiga	Baixa	5
Trinca por Fadiga	Média	2
Trinca por Fadiga	Alta	9
Remendo	Baixa	0
Remendo	Média	0
Remendo	Alta	6
Buraco	Baixa	1
Buraco	Média	0
Buraco	Alta	1

Tipo de Defeito	Qtd. Total
Remendos	6
Buracos	2
Trinca por Fadiga	16
Total	24