UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

William D'Abruzzo Martins

Segmentação e Detecção de Buracos em Vias para Sistemas Embarcados Críticos em prol de Veículos Robóticos.

São José do Rio Preto

2025

William D'Abruzzo Martins

Segmentação e Detecção de Buracos em Vias para Sistemas Embarcados Críticos em prol de Veículos Robóticos.

Monografia apresentada como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Financiadora: Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Brasil. Processo nº 2025/03224-4.

Universidade Estadual Paulista "Júlio de Mesquita Filho" - (UNESP)

Instituto de Biociências, Letras e Ciências Exatas - (IBILCE)

Departamento de Ciências de Computação e Estatística

Bacharelado em Ciência da Computação

Orientador: Prof. Dr. Diego Renan Bruno

São José do Rio Preto

2025

M386s

Martins, William
Segmentação e Detecção de Buracos em Vias para Sistemas Embarcados Críticos em prol de Veículos Robóticos. / William Martins. -- São José do Rio Preto, 2025
62 p.

Trabalho de conclusão de curso (Bacharelado - Ciência da Computação) - Universidade Estadual Paulista (UNESP), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto
Orientador: Diego Renan Bruno

1. Inteligência Artificial. 2. Visão Computacional. 3. Veículos Robóticos. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

William D'Abruzzo Martins

Segmentação e Detecção de Buracos em Vias para Sistemas Embarcados Críticos em prol de Veículos Robóticos.

Monografia apresentada como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, junto ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", Câmpus de São José do Rio Preto.

Financiadora: Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Brasil. Processo nº 2025/03224-4.

Banca Examinadora:

Prof. Dr. Diego Renan Bruno
Universidade Estadual Paulista "Júlio de Mesquita Filho"(UNESP)
Instituto de Biociências, Letras e Ciências Exatas (IBILCE)
Orientador

Prof. Dr. Lucas Correia Ribas
Universidade Estadual Paulista "Júlio de Mesquita Filho- (UNESP)
Instituto de Biociências, Letras e Ciências Exatas - (IBILCE)
Departamento de Ciências de Computação e Estatística

Prof. Dr. Wallace Correa de Oliveira Casaca
Universidade Estadual Paulista "Júlio de Mesquita Filho- (UNESP)
Instituto de Biociências, Letras e Ciências Exatas - (IBILCE)
Departamento de Ciências de Computação e Estatística

São José do Rio Preto
19 de novembro de 2025

Agradecimentos

Gostaria de agradecer a todas as pessoas que me ajudaram, não apenas na realização deste trabalho, mas também ao longo de toda a minha formação.

Primeiramente, à minha família — em especial à minha mãe, Fabíola, meu pai, Wilson, ao meu avô, Pascoal, e ao meu tio, Bruno — por sempre acreditarem em mim e investirem na minha educação desde cedo. Sem o apoio e o amor de vocês, este trabalho simplesmente não existiria.

Agradeço também aos professores do Departamento de Ciência da Computação e Estatística da UNESP de São José do Rio Preto, em especial aos professores Dr. Aleardo Manacero Junior e Dra. Renata Spolon Lobato, por me iniciarem na ciência; aos professores Dr. Lucas Correia Ribas e Dr. Wallace Casaca Correia, por me inspirarem com sua paixão pela pesquisa; e ao meu orientador, Dr. Diego Renan Bruno, pela parceria, inspiração e orientação fundamentais durante toda esta jornada e na realização deste trabalho.

Aos meus amigos Gabriel Leoni, Luan Guerra, João Victor (Haru), Guilherme Roma, Túlio Henry, Vinicius Borgue, Ricardo Fares, João Barroso, Tavex, Gabriel Scarano, Lucas Furriel e tantos outros que não caberiam aqui: muito obrigado por tornarem essa caminhada mais leve, divertida e inesquecível. Sem vocês, a jornada teria sido muito mais difícil — e certamente não teria valido tanto a pena.

À Ana Beatriz Silva Zerati, meu agradecimento por tornar minha realidade tão bela que sonhar se tornou desnecessário.

Agradeço à Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) por acreditar no projeto, fomentando-o por meio da bolsa de Iniciação Científica, processo nº 2025/03224-4.

Por fim, deixo meu agradecimento à grande parcela do povo brasileiro que, muitas vezes sem saber, financia a pesquisa e a educação pública no país.

'Repito por pura alegria de viver: a salvação é pelo risco, sem o qual a vida não vale a pena.'
(Clarice Lispector)

Resumo

Este trabalho apresenta o desenvolvimento e a avaliação de modelos de detecção e segmentação de buracos em vias voltados para aplicação em sistemas embarcados críticos e de baixo custo. Foram implementadas e comparadas arquiteturas de aprendizado profundo modernas, com foco na otimização do desempenho por meio de técnicas de quantização e poda de redes neurais. Para a tarefa de detecção, os modelos YOLOv11, RT-DETR e D-FINE foram analisados, sendo o D-FINE o que apresentou melhor desempenho geral devido ao seu módulo de refinamento, enquanto o YOLOv11 destacou-se pela maior velocidade de inferência. Para a segmentação, foram avaliadas arquiteturas genéricas e voltadas a tempo real, demonstrando que o modelo PP-LiteSeg obteve melhor equilíbrio entre acurácia e eficiência, especialmente após a quantização. Além disso, foi desenvolvido um conjunto de dados com perspectiva veicular e anotações automáticas para segmentação, possibilitando experimentos mais próximos das condições reais de direção. Os resultados evidenciam que é possível obter modelos precisos e eficientes para uso embarcado, contribuindo para o avanço de sistemas de assistência à direção e monitoramento autônomo de infraestrutura viária.

Palavras-chave: Detecção de Objetos, Segmentação Semântica, Aprendizado Profundo, Sistemas Embarcados, Otimização.

Abstract

This work presents the development and evaluation of object detection and segmentation models for pothole identification on roads, aimed at applications in low-cost and safety-critical embedded systems. Modern deep learning architectures were implemented and compared, focusing on performance optimization through techniques such as quantization and network pruning. For the detection task, the YOLOv11, RT-DETR, and D-FINE models were analyzed, with D-FINE achieving the best overall performance due to its refinement module, while YOLOv11 stood out for its higher inference speed. For segmentation, both general-purpose and real-time architectures were evaluated, demonstrating that the PP-LiteSeg model achieved the best balance between accuracy and efficiency, especially after quantization. Additionally, a dataset with vehicular perspective and automatically generated segmentation annotations was developed, enabling experiments that better reflect real driving conditions. The results show that it is possible to obtain accurate and efficient models for embedded use, contributing to the advancement of driver-assistance systems and autonomous road infrastructure monitoring.

Keywords: *Object Detection,Semantic Segmentation, Deep Learning, Embedded Systems, Optimization.*

Sumário

Lista de ilustrações **9**

Lista de tabelas **11**

1 INTRODUÇÃO **13**
1.1 Motivação **14**
1.2 Objetivos **15**

2 FUNDAMENTAÇÃO TEÓRICA **16**
2.1 Extração de Características **16**
2.2 Detecção de objetos **17**
2.2.1 YOLO (You Only Look Once) 18
2.2.2 Detectores Baseados em *Transformers* 21
2.2.2.1 Real Time Detector Transformer (RT-DETR) 22
2.2.2.2 D-FINE 23
2.3 Segmentação de objetos **24**
2.3.1 U-Net 26
2.3.2 DeepLab 26
2.3.3 Segformer 28
2.3.4 BiseNet 29
2.3.5 STDC 30
2.3.6 PPLite 32
2.4 Implantação de Redes Neurais **33**
2.4.1 Quantização 34
2.4.1.1 Quantização para FP16 34
2.4.1.2 Quantização para INT8 34
2.4.1.3 Treino com quantização 34
2.4.1.4 Erro de quantização 35
2.4.2 Poda de modelos (Pruning) 35

3 REVISÃO BIBLIOGRÁFICA **37**
3.1 Detecção de buracos em vias **37**
3.2 Segmentação de buracos em vias **38**

4 METODOLOGIA **39**
4.1 Coleta e Preparação do Conjunto de Dados **39**
4.2 Treinamento dos Modelos **40**

4.3 Avaliação e Implantação 41
4.4 Ferramentas Computacionais 42

5 RESULTADOS 43
5.1 Detecção de Objetos 43
5.2 Extratores de Características para Modelos Genéricos de Segmentação 45
5.3 Modelos Específicos para Segmentação em Tempo Real 47

6 CONCLUSÕES 52
6.1 Trabalhos Futuros 52
6.2 Lista de Publicações 53

REFERÊNCIAS 55

Lista de ilustrações

Figura 1 – Predição unificada do modelo Yolo. Para cada grid $S \times S$ prediz um número B de caixas, utilizando o processo de *non maximum supression* para selecionar a melhor caixa durante a inferência. 20

Figura 2 – Arquitetura do DETR. As características da imagem são extraídas por uma CNN e enviadas ao *encoder* e *decoder* das arquiteturas Transformers, que refinam as características e realizam a predição por conjuntos. 22

Figura 3 – Arquitetura geral do modelo RT-DETR. Após a extração de características pelo *backbone*, o bloco AIFI realiza a interação intra-escala do último estágio do *backbone* refinando as características para a fusão pelo bloco CCFF. Após, o Decoder realiza as seleções das *queries* utilizando o algoritmo de incerteza mínima. 23

Figura 4 – D-Fine com módulo FDR, cada caixa é refinada nas camadas subsequentes do decoder e é calculado um *offset* da caixa original. 24

Figura 5 – Exemplo de classificação (a), detecção de objetos (b), segmentação semântica (c) e segmentação de instâncias (d). 25

Figura 6 – Arquitetura U-Net 27

Figura 7 – Ilustração do modelo DeepLab 28

Figura 8 – Arquitetura do modelo Segformer 29

Figura 9 – Diferença entre diferentes tamanhos de campos receptivos. Um campo receptivo maior (amarelo) permite maior adquirir maior contexto para os modelos de segmentação. 29

Figura 10 – Arquitetura do modelo Bisenet 30

Figura 11 – Arquitetura dos blocos STDC. 31

Figura 12 – Visão geral da arquitetura do PP-LiteSeg, adaptada de (PENG et al., 2022). As características da imagem de entrada são inicialmente extraídas pelo *encoder* e agregadas no *Simple Pyramid Pooling Module* (SPPM) para capturar o contexto global. Em seguida, o *Flexible and Lightweight Decoder* (FLD) combina informações semânticas e detalhadas, que são refinadas pelo *Unified Attention Fusion* (UAF). 32

Figura 13 – Esquema de Poda em um modelo de Rede Neural. Neurônios e conexões removidas são indicados por $\times$. 34

Figura 14 – Comparação de *pruning* não estruturado (esquerda) e estruturado (direita). *Pruning* não estruturado tem mais flexibilidade e maior taxa de compreesão, contudo necessita de *hardware* específico para aceleração. *Pruning* estruturado, por sua vez, tem menos flexibilidade mas permite aceleração independente da plataforma. 35

Figura 15 – Metodologia do trabalho proposto e as ferramentas utilizadas em cada etapa. 39
Figura 16 – Anotação automática de caixas delimitadoras para máscaras de segmentação através do modelo Segment Anything Model (RAVI et al., 2024). 41
Figura 17 – Comparação entre a Poda Direta e a Poda Iterativa para razões similares. É possível observar que a Poda Direta mantém resultados mais altos de IoU para razões similares. 50

Lista de tabelas

Tabela 1 – Tabela de algoritmos e suas descrições. 18
Tabela 2 – Algoritmos de detecção de objeto baseados em Aprendizado Profundo . 19
Tabela 3 – Configurações dos modelos usados para a tarefa de Detecção de Objetos. 43
Tabela 4 – Desempenho dos modelos de detecção no conjunto de dados proposto. Os valores de FPS foram obtidos a partir do tempo médio de inferência. 43
Tabela 5 – Desempenho do modelo D-FINE em diferentes precisões. Valores entre parênteses indicam a diferença em relação ao baseline FP32. 44
Tabela 6 – Comparação do desempenho de inferência do modelo D-FINE em diferentes precisões. Valores entre parênteses indicam *speedup* em relação ao baseline FP32. 45
Tabela 7 – Desempenho do modelo Unet++ de acordo com o *backbone* utilizado. . 46
Tabela 8 – Desempenho do modelo DeepLab de acordo com diferentes backbones, considerando apenas as métricas principais de segmentação. 46
Tabela 9 – Desempenho do modelo Segformer de acordo com diferentes backbones, considerando apenas as métricas principais de segmentação. 46
Tabela 10 – Comparativo dos melhores resultados para cada arquitetura com seu respectivo *backbone*, incluindo tempo de inferência médio por imagem. 47
Tabela 11 – Comparativo dos modelos com menor tempo de inferência em cada arquitetura, mantendo apenas as métricas principais e tempo de inferência. 47
Tabela 12 – Desempenho dos modelos na resolução 640×640, considerando as métricas principais de segmentação. 48
Tabela 13 – Desempenho dos modelos na resolução 1024×1024, considerando as métricas principais de segmentação. 48
Tabela 14 – Tempo médio de inferência dos modelos nas resoluções 640 × 640 e 1024 × 1024, em ms, com FPS entre parênteses. 49
Tabela 15 – Comparação do desempenho de inferência em diferentes precisões com entrada de 1024 × 1024. Valores entre parênteses indicam aceleração relativa ao baseline FP32. 49
Tabela 16 – Desempenho do modelo PP-LiteSeg a 1024 × 1024 em diferentes precisões. Valores entre parênteses indicam a diferença absoluta em pontos percentuais em relação ao baseline FP32. 49
Tabela 17 – Desempenho e aceleração de inferência do método de Poda Direta em diferentes tamanhos de modelo. 51

Lista de abreviaturas e siglas

ADAS	*Advanced Driver-Assistance Systems*
CNN	*Convolutional Neural Networks*
CNT	Confederação Nacional do Transporte
DETR	*Detector Transformer*
FCN	*Fully Convolutional Neural Network*
RT-DETR	*Real Time Detector Transformer*
ViT	*Vision Transformers*
YOLO	*You Only Look Once*

1 Introdução

A qualidade da infraestrutura viária desempenha um papel fundamental no suporte às atividades econômicas e na garantia da segurança pública. Irregularidades, buracos e danos nas estradas não apenas impactam negativamente a indústria automotiva, reduzindo a vida útil dos veículos, mas também representam riscos significativos à segurança pública, aumentando a chance de acidentes (SAFYARI; MAHDIANPARI; SHIRI, 2024).

No Brasil, a situação é ainda mais crítica. De acordo com uma pesquisa da Confederação Nacional do Transporte (CNT), divulgada em 2021, a cada mil quilômetros de rodovias pavimentadas, identificam-se, em média, 16 pontos críticos, sendo a maioria desses caracterizada por buracos de dimensões superiores ao diâmetro de um pneu.

Embora os investimentos em infraestrutura viária tenham aumentado, o ritmo de reparo de buracos nas estradas ainda é insuficiente para acompanhar a deterioração das vias. Como consequência, a responsabilidade por evitar acidentes recai principalmente sobre os motoristas. No entanto, fatores humanos, como embriaguez, sonolência e distrações, permanecem como as principais causas de acidentes de trânsito (BRUNO, 2020).

Diante desse cenário, a indústria automobilística tem investido cada vez mais em tecnologias de assistência ao motorista, incluindo sistemas capazes de detectar danos nas vias (SAFYARI; MAHDIANPARI; SHIRI, 2024). No Brasil, o programa Rota 2030 busca impulsionar o desenvolvimento do setor automobilístico, incentivando, entre outras iniciativas, a adoção de tecnologias assistivas à direção (Ministério do Desenvolvimento, Indústria, Comércio e Serviços, 2024). Assim, os Sistemas Avançados de Assistência ao Motorista (ADAS) desempenham um papel fundamental na promoção de uma condução mais segura, auxiliando os condutores a responder adequadamente a diferentes situações no trânsito (PAUL et al., 2016).

A detecção de buracos para Sistemas Avançados de Assistência ao Motorista (ADAS) é realizada principalmente por dois métodos: análise de imagens 2D, utilizando câmeras, e sensores 3D baseados em tecnologia *laser* (LYDON et al., 2020). Embora os sensores 3D ofereçam alta precisão na captura da topografia das vias, sua implementação é onerosa devido ao alto custo dos dispositivos. Em contrapartida, métodos baseados em imagens 2D apresentam uma alternativa mais acessível, proporcionando resultados com boa acurácia e potencial para processamento em tempo real.

Os métodos de detecção de buracos 2D, por sua vez, são divididos em duas principais categorias: Algoritmos de Visão Computacional e Aprendizado de Máquina (ASAD et al., 2022a).Métodos de Visão Computacional aplicados na literatura incluem técnicas como *Local Binary Pattern* (LBP), classificação de textura, técnicas limiares de histograma, Transformada Discreta de Wavelet (DWT) e entre outras (DHIMAN; KLETTE, 2020).

Apesar da alta acurácia, os algoritmos de visão computacional apresentam um custo elevado, o que limita sua viabilidade para aplicações em tempo real. Além disso, seu desempenho é comprometido em condições climáticas adversas, como chuva intensa, neblina e baixa luminosidade, reduzindo sua confiabilidade na detecção de irregularidades viárias (MA et al., 2022). Essas limitações tornam esses métodos inadequados para ADAS, que exigem respostas rápidas e consistentes em diferentes cenários ambientais. Nesse contexto, abordagens baseadas em Aprendizado de Máquina, especialmente aquelas fundamentadas em Aprendizado Profundo, surgem como alternativas mais robustas e adaptáveis.

Algoritmos tradicionais de aprendizado de máquina foram aplicados no uso de detecção de buracos em vias, como *Support Vector Machines* (SVM), Regressão Logística, Bayes, *K-Nearest Neighbors* e *Random Forests* (EGAJI et al., 2021). Porém, o estado da arte para problemas de visão computacional envolve o uso de Redes Neurais Profundas, em especial Redes Neurais Convolucionais (CNNs). Na abordagem envolvendo Redes Neurais Profundas, trabalhos da literatura foram realizados envolvendo o uso de CNNs e Redes Residuais para detecção e segmentação de objetos (JAKUBEC et al., 2023; DHIMAN; KLETTE, 2020)

1.1 Motivação

Apesar das diversas soluções disponíveis, poucos trabalhos na literatura são adaptados para ADAS veiculares. Isso ocorre porque muitas abordagens envolvem soluções computacionalmente custosas e utilizam conjuntos de dados que não refletem adequadamente as condições reais do ponto de vista do veículo. Diante desse cenário, este projeto tem como foco a implementação de modelos de aprendizado profundo para detecção e segmentação de objetos, abordando as seguintes lacunas identificadas na literatura:

- **Disponibilidade limitada de conjuntos de dados práticos:** Muitos estudos dependem de conjuntos de dados contendo imagens que não são adequadas para aplicações de ADAS veiculares. Essas imagens são frequentemente capturadas a partir de uma perspectiva aérea, diretamente acima dos buracos, em vez da visão a partir de um veículo. Essa discrepância afeta negativamente o desempenho do modelo durante a inferência embarcada, destacando a necessidade de conjuntos de dados alinhados com cenários reais de direção.

- **Falta de otimização para sistemas embarcados de baixo custo:** Pesquisas existentes frequentemente utilizam modelos de detecção computacionalmente intensivos, incompatíveis com o processamento em tempo real em dispositivos embarcados de baixo custo. A alta latência de inferência desses modelos os torna impraticáveis para aplicações em tempo real, evidenciando a necessidade de soluções leves e eficientes, adaptadas a ambientes com restrições de hardware.

1.2 Objetivos

O objetivo principal desse trabalho é avaliar modelos do estado-da-arte de detecção e segmentação de objetos aplicados ao contexto de buracos em vias em sistemas ADAS, avaliando-o quanto à sua aplicação em sistemas de borda de baixo custo. De maneira específica, os objetivos desse trabalho contemplam:

- Automatizar a tarefa de rotulação de imagens, unindo redes de detecção como pré-processamento para segmentação;
- Treinar e avaliar modelos de detecção de objeto para a tarefa de detecção de buracos em vias;
- Treinar e avaliar modelos de segmentação de objeto para a tarefa de segmentação de buracos em vias;
- Adaptar modelos para seu uso em dispositivos embarcados de baixo nível por meio do uso de técnicas como quantização, poda de modelos, *Transfer Learning*.

2 Fundamentação Teórica

Nesse capítulo será apresentada a fundamentação teórica que deu base ao desenvolvimento do trabalho. Na primeira seção será tratado o uso de Redes Neurais Artificiais para extração de características, posteriormente será tratado sobre o uso das Redes Neurais Artificiais para tarefa de detecção de objetos e segmentação de objetos. Por fim, será tratado sobre técnicas de implantação de redes neurais como quantização e poda de modelos.

2.1 Extração de Características

A extração de características em imagens é o processo de identificar e capturar informações relevantes e distintas que descrevem o conteúdo visual de uma imagem. Essa etapa é fundamental em tarefas como classificação, detecção e segmentação, pois permite que algoritmos interpretem e diferenciem padrões visuais de forma eficiente e precisa.

As Redes Neurais Convolucionais (Convolutional Neural Networks — CNNs) constituem uma variação das Redes Neurais Profundas desenvolvidas especificamente para o processamento e análise de imagens, desenvolvidas por LeCun (LECUN et al., 1989). Essa arquitetura foi concebida para lidar com três características intrínsecas das imagens: a alta dimensionalidade dos dados, a correlação espacial local entre os pixels e a invariância a transformações geométricas, como translações e rotações (PRINCE, 2023).

A primeira característica refere-se à alta dimensionalidade das imagens, uma vez que, por exemplo, uma resolução típica de 224×224 pixels com três canais de cor (RGB) resulta em mais de 150 mil entradas. Em redes neurais tradicionais totalmente conectadas, isso implicaria em bilhões de parâmetros a serem aprendidos, o que torna o treinamento computacionalmente inviável. A segunda característica está relacionada à correlação local entre pixels vizinhos: regiões adjacentes tendem a conter informações significativas, como bordas, texturas ou padrões. Redes neurais convencionais não exploram essa noção de proximidade, tratando todos os neurônios como independentes, o que inviabiliza o reconhecimento eficiente de padrões locais. Por fim, a terceira característica diz respeito à invariância a transformações geométricas — o significado semântico de uma imagem geralmente se mantém inalterado sob pequenas mudanças de posição, escala ou rotação. Por exemplo, uma árvore continua sendo reconhecida como tal mesmo que esteja deslocada alguns pixels para a esquerda. Capturar essas propriedades em arquiteturas tradicionais exigiria um volume extremamente elevado de dados para que o modelo generalizasse adequadamente.

Para lidar com as particularidades das imagens, as Redes Neurais Convolucionais incorporam três conceitos fundamentais: interações esparsas, compartilhamento de parâmetros e

representações equivariantes (GOODFELLOW; BENGIO; COURVILLE, 2016). As interações esparsas abordam a alta dimensionalidade das imagens ao restringir as conexões entre os neurônios a pequenas regiões locais da entrada. Isso permite que a rede detecte apenas características relevantes, como bordas e padrões, por meio do uso de convoluções que funcionam como filtros aplicados sobre a imagem (YOUNESI et al., 2024). O compartilhamento de parâmetros facilita o reconhecimento de padrões locais, uma vez que um mesmo conjunto de pesos (filtro) é utilizado em diferentes regiões da imagem. Essa abordagem é particularmente eficaz devido à recorrente semelhança entre pixels vizinhos, que frequentemente descrevem o mesmo objeto ou estrutura. Por fim, as representações equivariantes, proporcionadas pela operação de convolução, conferem à rede a capacidade de reconhecer objetos independentemente de pequenas variações geométricas, como translações.

Nessa monografia, foram utilizadas diversas variações de Redes Neurais Convolucionais para extração de características, como ResNet, MobileNet, EfficientNet, HRNet e outras arquiteturas híbridas que utilizam-se também de *Vision Transformers* (VASWANI et al., 2017) como a arquitetura MobileViT.

2.2 Detecção de objetos

Dentro de Visão Computacional, detecção de objetos é uma tarefa que consiste em localizar, dentro de uma imagem, objetos de uma mesma classe. Diferente de Classificação de Imagens, que apenas determina se um objeto está presente ou não, a tarefa de detecção de objetos consiste em identificar a posição exata dos objetos por meio de caixas delimitadoras. A tarefa de detecção de objetos serve como base para outras, como Segmentação de objetos e Rastreamento de objetos, além de aplicações no mundo real, como direção autônoma, robótica, inspeção industrial, videomonitoramento e outros.

Soluções para a tarefa de detecção de objetos podem ser divididas em dois períodos históricos: algoritmos tradicionais (antes de 2014) e algoritmos baseados em Aprendizado Profundo (depois de 2014) (ZOU et al., 2023).

Entre os algoritmos tradicionais, estão o Viola Jones (VIOLA; JONES, 2001), Histograma de Gradientes Orientados (DALAL; TRIGGS, 2005) e *Deformable Part-Based Model* (FELZENSZWALB; MCALLESTER; RAMANAN, 2008), conforme apresentado na Tabela 1.

Já nas abordagens baseadas em Aprendizado Profundo, existem três principais categorias de detectores: detectores de um estágio, detectores de dois estágios e detectores baseados em *Vision Transformers*, conforme apresentado na Tabela 2. Os detectores de dois estágios geram um conjunto de proposições de objetos no primeiro estágio e, em seguida, refinam e classificam essas proposições no segundo estágio. Por outro lado, os detectores de um estágio eliminam a etapa intermediária de proposição de objetos e realizam a detecção e classificação simultaneamente, tornando o processo mais rápido, embora geralmente com uma

Tabela 1 – Tabela de algoritmos e suas descrições.

Algoritmo	**Ano**	**Descrição**
Viola Jones	2001	Detecção em tempo real de faces. Uma janela deslizante percorre toda imagem verificando se existe uma face na janela.
Histograma de Gradientes Orientados	2005	Utiliza a distribuição dos gradientes de intensidade em regiões da imagem, obtendo informações de borda e textura. Foi amplamente utilizado para detecção de pedestres.
Deformable Part-Based Models	2008	Cada objeto é representado como uma composição de partes deformáveis. Durante o treinamento, o modelo aprende a identificar e decompor o objeto em suas partes constituintes, enquanto, na inferência, o objetivo é reconhecer o objeto a partir da composição dessas partes.

Fonte: Elaborada pelo autor.

leve perda de precisão (ZAIDI et al., 2021).

Mais recentemente, os detectores baseados em *Vision Transformers* vêm ganhando destaque por empregarem mecanismos de autoatenção dos *Transformers* em vez das convoluções tradicionais para a detecção de objetos. Essas abordagens demonstram um desempenho superior em relação aos métodos baseados em CNNs, especialmente em cenários com grande variabilidade de contexto e relações espaciais complexas. No entanto, esse avanço vem acompanhado de um aumento significativo na complexidade computacional e em tempos de treinamento mais longos (SHEHZADI et al., 2023).

2.2.1 YOLO (You Only Look Once)

O YOLO (You Only Look Once) (REDMON et al., 2016) é um modelo de detecção de objetos amplamente reconhecido por equilibrar alta acurácia e velocidade. Seu diferencial está na forma como a tarefa de detecção é abordada: o YOLO reformula o problema como uma tarefa de regressão de estágio único, ao contrário dos detectores em duas etapas, como os RCNN, que primeiro geram propostas de regiões e, em seguida, classificam-nas separadamente.

Tabela 2 – Algoritmos de detecção de objeto baseados em Aprendizado Profundo

Categoria	**Algoritmo**	**Ano**
Detectores de dois estágios	R-CNN	2014
	SPP-NET	2014
	Fast R-CNN	2015
	Faster R-CNN	2015
Detectores de um estágio	YOLO	2015
	SSD	2015
	RetinaNet	2017
	CenterNet	2019
Detectores baseados em *Transformers*	DETR	2020
	Deformable DETR	2020
	CO-DETR	2022
	RT-DETR	2023

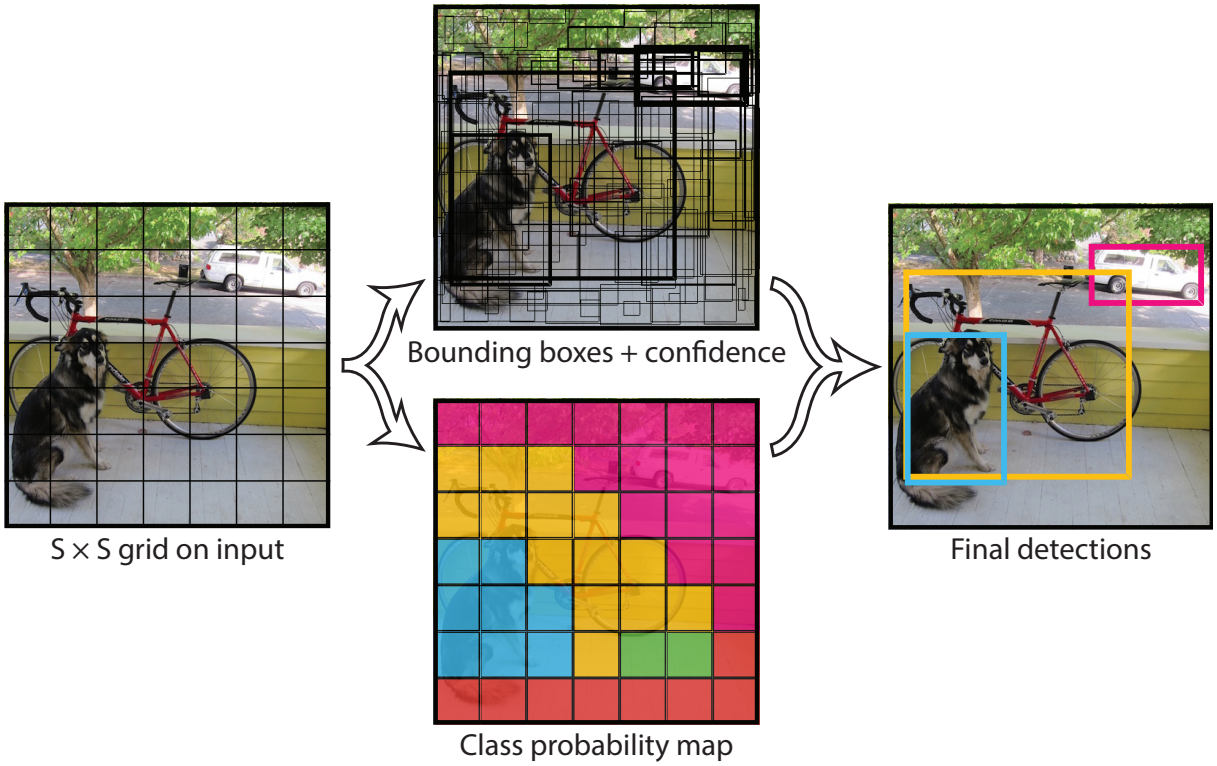
Fonte: Elaborada pelo autor.

Essa reformulação faz com que a imagem passe apenas uma vez na rede, sendo extremamente rápido.

O modelo realiza predições considerando a imagem inteira, estimando simultaneamente caixas delimitadoras e probabilidades de classe em diferentes regiões. Inicialmente, a entrada é dividida em uma **grade** $S \times S$, na qual cada célula cuja região central contém um objeto é responsável por predizer B caixas delimitadoras, acompanhadas de uma pontuação de confiança. Além disso, cada célula prevê a distribuição de probabilidades sobre todas as classes para os objetos potenciais naquela região. Entre as B predições geradas, a caixa delimitadora selecionada é aquela que apresenta o **maior IoU** em relação à anotação de referência. Na Figura 1 é ilustrado o processo unificado de predição do YOLO.

A oitava versão do YOLO trouxe melhorias significativas ao adotar uma abordagem de detecção anchor-free (livre de âncoras), na qual o modelo localiza diretamente o centro dos objetos e, a partir dessa informação, estima sua posição e dimensões. Nos métodos tradicionais de detecção de objetos, utilizavam-se anchor boxes — caixas pré-definidas que serviam como aproximações iniciais das caixas reais (ground truth) e eram refinadas durante o treinamento por meio de funções de perda. No entanto, esse tipo de abordagem apresentava uma sensibilidade elevada ao tamanho, proporção e quantidade dessas âncoras (TIAN et al., 2022), exigindo ajustes manuais de hiperparâmetros. Ao eliminar as anchor boxes, o modelo se torna mais simples e eficiente, além de acelerar o processo de pós-processamento, especialmente durante a aplicação da técnica de Supressão Não Máxima (Non-Maximum Suppression – NMS).

Figura 1 – Predição unificada do modelo Yolo. Para cada grid $S \times S$ prediz um número B de caixas, utilizando o processo de *non maximum supression* para selecionar a melhor caixa durante a inferência.



Fonte: (REDMON et al., 2016)

Com o objetivo de tornar o modelo ainda mais adequado para aplicações em tempo real, a décima versão do YOLO elimina a etapa de pós-processamento conhecida como Non-Maximum Suppression (NMS), que anteriormente impactava a latência na inferência (WANG et al., 2024). Para isso, os autores propõem uma abordagem de treinamento livre de NMS (NMS-Free), baseada em um esquema de atribuição dupla de rótulos (*dual label assignment*), utilizando duas cabeças distintas na arquitetura. Essa estratégia melhora tanto a eficiência quanto o desempenho durante a inferência:

One-to-many: Cada objeto no rótulo (*ground truth*) pode ser associado a múltiplas predições. Neste caso, a seleção da melhor predição é feita com o uso do NMS na etapa de pós-processamento. Essa abordagem fornece uma supervisão mais robusta durante o treinamento, porém compromete a velocidade de inferência devido à aplicação do NMS.

One-to-one: Cada objeto real é associado a uma única predição. Isso elimina a necessidade de NMS durante a inferência, resultando em maior velocidade. No entanto, essa estratégia tende a oferecer uma supervisão mais fraca, o que pode impactar negativamente a acurácia.

Durante o treinamento, o modelo utiliza ambas as cabeças, permitindo que a rede extraia o máximo de informações supervisionadas da abordagem *one-to-many*. Já na inferência,

apenas a cabeça *one-to-one* é utilizada, proporcionando maior velocidade sem comprometer significativamente o desempenho.

A décima primeira versão do YOLO aprimora ainda mais as capacidades em tempo real de seu antecessor, introduzindo novos módulos arquiteturais (KHANAM; HUSSAIN, 2024). Dentre eles, destacam-se:

O bloco CSPSA, que reforça a atenção espacial nos mapas de características;

O bloco C3k2, que melhora a agregação de características;

O módulo C2PSA, que também trabalha com atenção espacial, mas com foco especial na detecção de objetos pequenos ou parcialmente ocultos.

Essas melhorias contribuem para o aumento da acurácia na detecção, especialmente de objetos pequenos, e para a redução do tempo de inferência, mantendo a eficiência do modelo em cenários de alta demanda computacional.

2.2.2 Detectores Baseados em *Transformers*

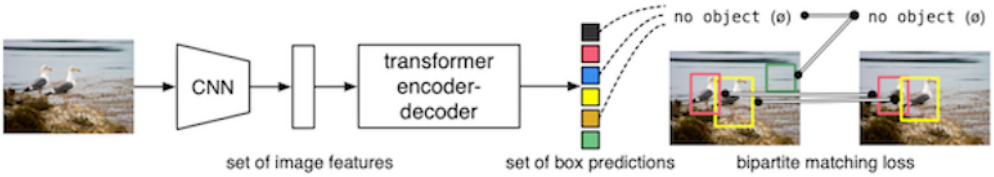
A arquitetura Transformer revolucionou o campo do Processamento de Linguagem Natural ao introduzir o mecanismo de atenção (attention mechanism), que permite ao decodificador de texto concentrar-se em partes específicas da entrada e atribuir pesos distintos à importância de cada elemento em relação aos demais. Esse mecanismo torna possível capturar dependências de longo alcance e relações contextuais complexas dentro dos dados (VASWANI et al., 2017).

Com base nessa mesma fundação, os Transformers Visuais (Visual Transformers – ViT) foram propostos para modelar, de forma simultânea, as relações entre todas as regiões de uma imagem (DOSOVITSKIY et al., 2021). Essa abordagem possibilita uma compreensão global da cena visual, ao considerar interações entre diferentes partes da imagem, mesmo que estejam distantes entre si espacialmente.

Em 2020, o grupo de pesquisa do Facebook publicou o modelo DEtection TRansformer (DETR) (CARION et al., 2020), um modelo que combinava Redes Neurais Convolucionais com Mecanismos de Atenção. O grande diferencial do modelo é o fato de ele não trabalhar nem com proposições de regiões (como os detectores de dois estágios) e nem com predição do centro do objeto através de anchor boxes ou grid (como modelos do Yolo), mas sim predizendo todos os objetos de uma vez e treinando com uma função de perda que realiza um emparelhamento bipartido único entre a caixa predita e a caixa real.

Dado N predições, com N sendo maior que o número de objetos na imagem, o modelo encontra o emparelhamento ótimo que minimiza o custo de correspondência entre o conjunto de caixas reais y e o conjunto de caixas preditas $\hat{y}$ buscando uma partição dos N elementos $\sigma \in \mathfrak{S}_N$.

Figura 2 – Arquitetura do DETR. As características da imagem são extraídas por uma CNN e enviadas ao *encoder* e *decoder* das arquiteturas Transformers, que refinam as características e realizam a predição por conjuntos.



Fonte: (CARION et al., 2020)

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i=1}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) \tag{2.1}$$

Após encontrar a partição, aplica a função de perda total *Hungarian Loss* (KUHN, 1955) que é a Cross Entropy loss combinada com a regressão da caixa.

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + 1_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}(i)}) \right] \tag{2.2}$$

Conforme apresentado na Figura 2, o modelo é composto por uma CNN como *backbone* que extrai as características e um *encoder* e *decoder* básicos da arquitetura Transformer (VASWANI et al., 2017).

Baseado nesse detector clássico, outros modelos baseados na arquitetura Transformer surgem para aplicações em tempo real, como o RT-DETR (ZHAO et al., 2024) e o D-FINE (PENG et al., 2024).

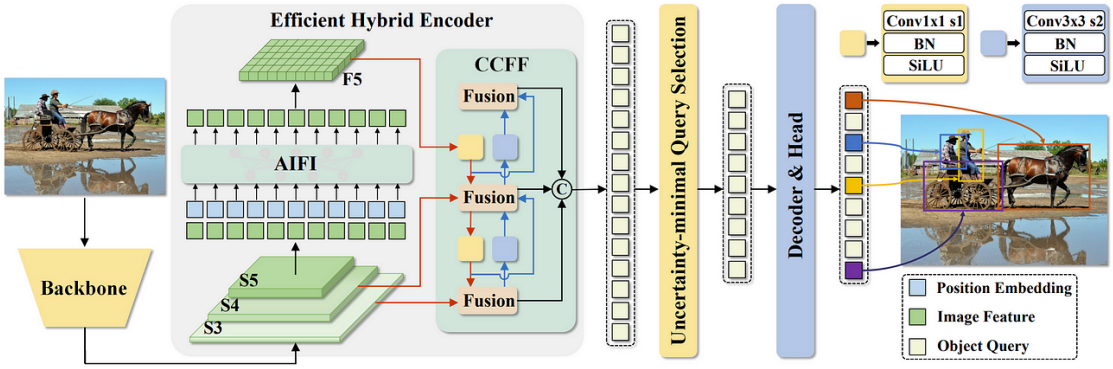
2.2.2.1 Real Time Detector Transformer (RT-DETR)

A principal contribuição do RT-DETR consiste em adaptar a arquitetura do DETR para aplicações em tempo real, buscando manter um bom nível de acurácia alinhado a uma velocidade de inferência compatível com tais exigências. As duas principais inovações dessa nova arquitetura estão na proposição de um encoder híbrido, voltado à fusão eficiente de características, e na introdução de um novo mecanismo no decoder, projetado para gerar caixas delimitadoras com maior precisão. A arquitetura geral do RT-DETR está ilustrada na Figura 3.

O encoder híbrido é composto por dois módulos principais: o AIFI (*Attention-based Intra-scale Feature Interaction*) e o CCFF (*Cross-Scale Feature Fusion*). O módulo AIFI realiza interações intraescalares no último estágio do *backbone*, onde se encontram as características semanticamente mais relevantes. Já o CCFF é responsável pela fusão de *features* extraídas em escalas adjacentes, permitindo uma representação mais rica e coerente entre diferentes níveis de abstração. Por fim, o *decoder* melhora as *queries* geradas pelo DETR utilizando o

conceito de seleção de *queries* através da incerteza mínima, que consiste em considerar tanto a confiança da classe quanto a confiança da caixa de predição.

Figura 3 – Arquitetura geral do modelo RT-DETR. Após a extração de características pelo *backbone*, o bloco AIFI realiza a interação intra-escala do último estágio do *backbone* refinando as características para a fusão pelo bloco CCFF. Após, o Decoder realiza as seleções das *queries* utilizando o algoritmo de incerteza mínima.



Fonte: (ZHAO et al., 2024)

2.2.2.2 D-FINE

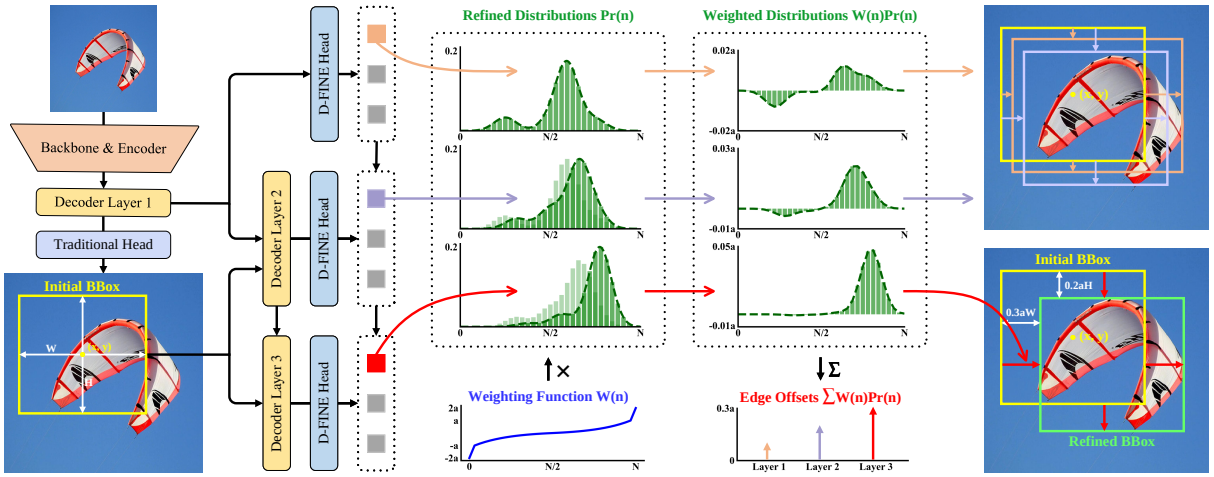
Seguindo a proposta do RT-DETR de adaptar arquiteturas baseadas em *Transformers* para tarefas de detecção em tempo real, Peng *et al.* apresentaram o modelo D-FINE (PENG et al., 2024). Essa arquitetura alcança eficiência temporal ao introduzir duas contribuições principais: o módulo FDR (*Fine-Grained Distribution Refinement*) e uma técnica de auto-destilação leve denominada GO-LSD (*Global Optimal Label-guided Self-Distillation*).

O módulo FDR busca refinar de forma iterativa as distribuições de probabilidade responsáveis pela regressão das caixas delimitadoras. Diferente de abordagens tradicionais que predizem diretamente as coordenadas dos vértices, o FDR modela cada lado da caixa — superior, inferior, esquerdo e direito — como uma distribuição discreta de probabilidades. Assim, para cada uma das quatro bordas, o modelo aprende uma distribuição $Pr^l(n)$, onde n representa os deslocamentos (ou *offsets*) possíveis em relação à posição inicial da borda, e $W(n)$ é uma função de ponderação associada a cada deslocamento.

Durante o refinamento, as camadas subsequentes do *decoder* ajustam as distribuições com base na configuração inicial da caixa $d^0 = \{t, b, l, r\}$, de forma que o deslocamento refinado em cada estágio l é calculado como:

$$d^l = d^0 + \{H, H, W, W\} \cdot \sum_{n=0}^{N} W(n) Pr^l(n)$$

Figura 4 – D-Fine com módulo FDR, cada caixa é refinada nas camadas subsequentes do decoder e é calculado um *offset* da caixa original.



Fonte: (PENG et al., 2024)

Onde H e W correspondem à altura e à largura da caixa, respectivamente. Na Figura 4 está presente o funcionamento do modelo com o módulo FDR. Dessa maneira, o módulo FDR permite que o modelo aprenda correções mais precisas e contínuas para as caixas delimitadoras, reduzindo erros de localização e mantendo a eficiência necessária para aplicações em tempo real.

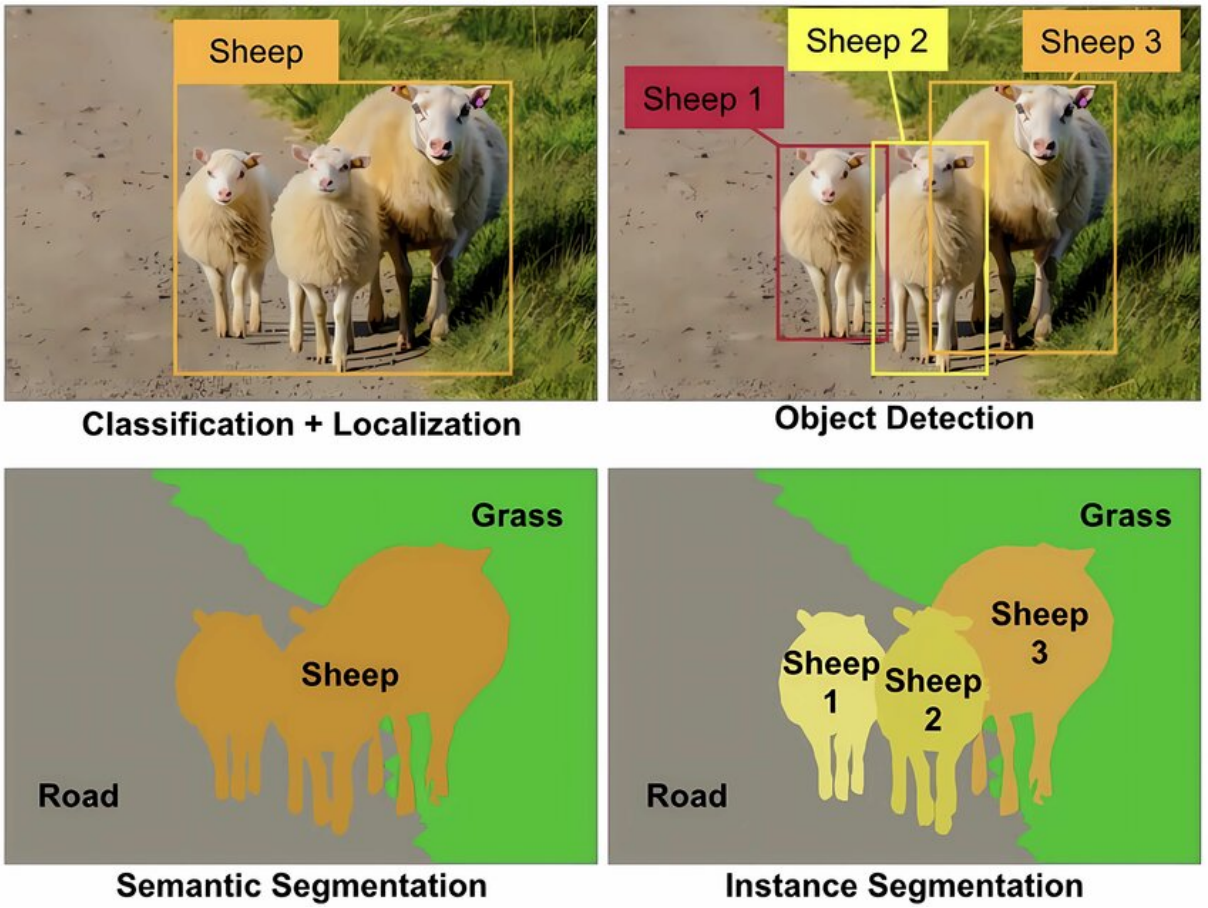
O módulo de destilação **GO-LSD** (*Global Optimal Localization Self-Distillation*) realiza uma auto-destilação de localização entre as distribuições refinadas da última camada e aquelas das camadas mais rasas. Para isso, o método aplica o *Hungarian Matching* em cada camada, a fim de encontrar a correspondência ótima entre o conjunto de predições intermediárias e o conjunto de predições finais refinadas. Em seguida, a perda de destilação de localização é calculada apenas sobre esses pares alinhados de forma ótima. Esse processo força as camadas mais rasas a imitarem a precisão de localização da camada final, resultando em uma melhoria significativa na qualidade das representações intermediárias e na convergência geral do detector.

2.3 Segmentação de objetos

Diferente da Detecção de Objetos, que apenas localiza e delimita objetos por meio de caixas delimitadoras, a Segmentação de Objetos vai além ao identificar com precisão os contornos dos objetos na imagem. Essa tarefa classifica cada pixel individualmente, determinando se pertence ao objeto ou ao plano de fundo, resultando em uma identificação mais detalhada, conforme apresentado na Figura 5.

A segmentação de objetos é dividido em três ramos: (WANG et al., 2022)

Figura 5 – Exemplo de classificação (a), detecção de objetos (b), segmentação semântica (c) e segmentação de instâncias (d).



Fonte: (FERNáNDEZ; RODRíGUEZ, 2023)

- **Segmentação Semântica**: Classifica cada pixel da imagem como pertencente a uma determinada classe, sem diferenciar instâncias individuais de um mesmo objeto.
- **Segmentação de Instâncias**: Identifica e diferencia cada objeto individualmente dentro da mesma classe, sem necessariamente considerar o contexto global da cena.
- **Segmentação Panóptica**: Combina a segmentação semântica e a segmentação por instâncias, fornecendo uma visão abrangente da cena. Enquanto o fundo e as áreas homogêneas são segmentados semanticamente, cada objeto distinto recebe uma identificação única, permitindo uma representação detalhada tanto dos elementos individuais quanto do contexto geral.

Assim como na detecção de objetos, a segmentação também pode ser realizada por meio de algoritmos tradicionais ou abordagens baseadas em Aprendizado Profundo. Os métodos tradicionais utilizam técnicas como detecção de bordas, análise de gradientes e modelagem de formas para segmentar objetos em uma imagem, apresentando a vantagem de

exigir baixo custo computacional. No entanto, esses métodos tendem a falhar em cenários mais complexos, como imagens com iluminação variável, baixa resolução ou alto nível de ruído. Para superar essas limitações, redes neurais profundas passaram a ser amplamente utilizadas na segmentação, permitindo a criação de modelos mais robustos e adaptáveis a diferentes condições visuais, melhorando significativamente a precisão da segmentação em aplicações do mundo real.

Nesta monografia, foram avaliados diferentes modelos de segmentação semântica aplicados à tarefa de segmentação de buracos em vias, incluindo as arquiteturas U-Net, DeepLab, Segformer, PPLiteSeg, STDC, BiSeNet e PIDNet.

2.3.1 U-Net

Proposta por Olaf Ronneberger, Philipp Fischer e Thomas Brox (RONNEBERGER; FISCHER; BROX, 2015), a rede U-Net é criada para segmentação de imagens biomédicas como uma extensão do modelo FCN (LONG; SHELHAMER; DARRELL, 2015). A rede FCN adiciona ao *backbone* camadas sucessivas em que as operações de *pooling* são trocadas por *upsampling* a fim de retornar à resolução original e gerar os mapas de segmentação. Essas camadas sucessivas são chamadas de **caminho expansivo**, enquanto o *backbone* é chamado de **caminho contrativo**.

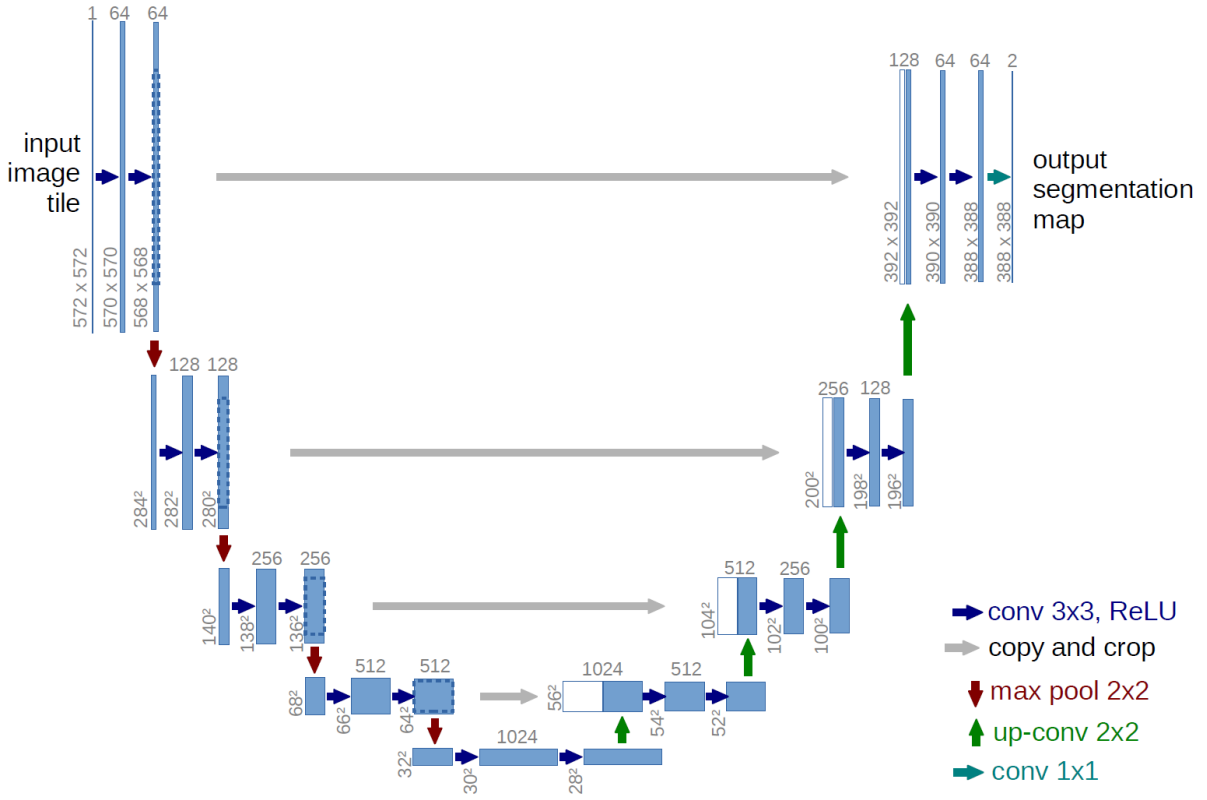
Uma diferença em relação ao FCN é que, na U-Net, o *upsampling* utiliza um grande número de canais de ativação, o que permite propagar informações de contexto para as camadas de maior resolução. Isso torna o caminho expansivo simétrico ao caminho de contração, formando uma estrutura em forma de "U", como apresentado na Figura 6. A U-Net resolve o problema de perda de informação presente nas FCNs ao adicionar conexões de atalho (*skip connections*) entre cada par de camadas correspondentes de *downsampling* e *upsampling*. Essas conexões têm se mostrado eficazes na recuperação de detalhes de alta precisão.

O modelo U-Net foi desenvolvido para aplicações em imagens médicas. Diferentemente das imagens naturais, as imagens médicas exigem segmentações de alta precisão, especialmente em termos de IoU (*Intersection over Union*), uma vez que erros podem comprometer significativamente a experiência do usuário em ambientes clínicos. A rede U-Net++ (ZHOU et al., 2018) melhora a U-Net adicionando conexões de atalhos densas, nas quais as *features* do caminho contrativo passam sobre um bloco de convolução denso que os enriquece de forma a deixar o nível semântico próximo às *features* do caminho expansivo.

2.3.2 DeepLab

A rede DeepLab (CHEN et al., 2017) foi desenvolvida para lidar com três grandes dificuldades na Segmentação Semântica: redução da resolução das características, presença de objetos em múltiplas escalas e baixa precisão nas bordas. A arquitetura e seus módulos são

Figura 6 – Arquitetura U-Net



Fonte: (RONNEBERGER; FISCHER; BROX, 2015)

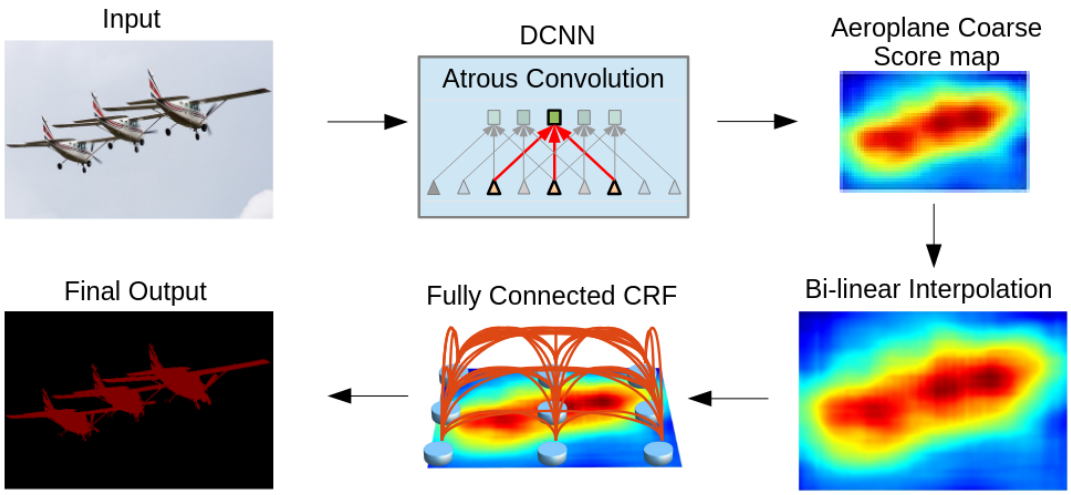
apresentados na Figura 7.

O primeiro problema decorre das combinações de *pooling* e *downsampling* presentes nos modelos *backbones* de extração de características, que acabam reduzindo a resolução espacial das representações e dificultando uma segmentação precisa. Para lidar com isso, o DeepLab utiliza convoluções dilatadas, definidas pela equação $y[i] = \sum_k x[i + rk] \cdot w[k]$, onde r é a taxa de dilatação, isto é, o número de pixels saltados entre os elementos do filtro. Essa abordagem permite aumentar arbitrariamente o campo de visão da rede, apenas inserindo $r-1$ zeros entre elementos consecutivos do filtro, o que possibilita ampliar a entrada sem adicionar mais parâmetros ou custo computacional.

O segundo problema é resolvido inspirado no *Spatial Pyramid Pooling*, utilizando múltiplas convoluções dilatadas em paralelo, com cada uma com diferentes taxas de amostragem. Essa estratégia permite que a rede capture informações contextuais em diferentes escalas simultaneamente.

Por fim, o terceiro problema — a baixa acurácia nas bordas — é tratado por meio do uso de *Conditional Random Fields* (CRFs) (KRäHENBüHL; KOLTUN, 2011). Essa técnica combina as pontuações de classes geradas por classificadores multiclasse com informações de baixo nível, capturadas a partir das interações locais entre pixels e contornos ou entre

Figura 7 – Ilustração do modelo DeepLab



Fonte: (CHEN et al., 2017)

superpixels. Com isso, é possível refinar as predições da segmentação, alinhando melhor os contornos dos objetos às suas bordas reais na imagem.

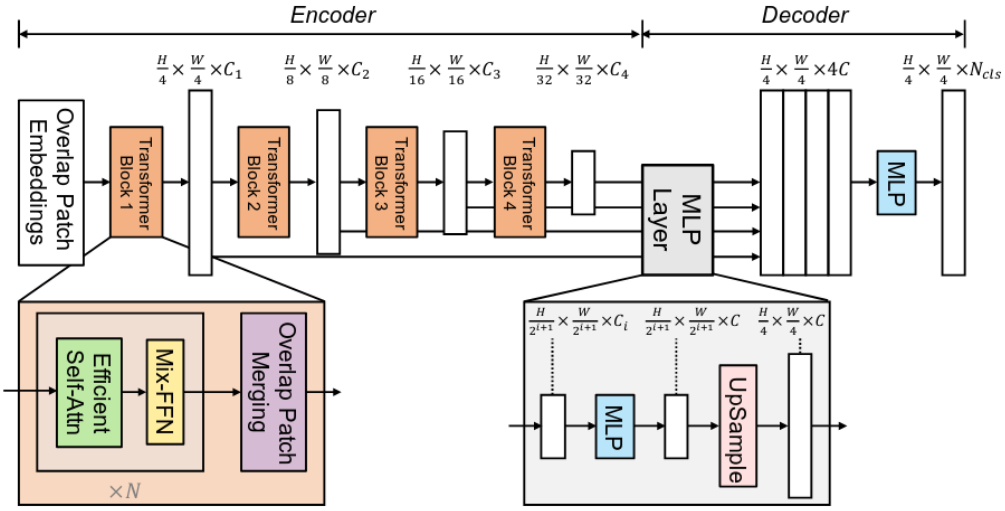
2.3.3 Segformer

Segformer é uma arquitetura Transformer para segmentação semântica que utiliza um *backbone* hierárquico sem codificação posicional, capaz de gerar simultaneamente características detalhadas em alta resolução e representações mais grosseiras em baixa resolução (XIE et al., 2021). O modelo também conta com um *decoder* leve, composto inteiramente por MLPs, que aproveita as representações geradas pelo Transformer ao agregar informações de diferentes camadas. A arquitetura do modelo é apresentado na Figura 8.

O *encoder* gera representações em múltiplos níveis e escalas a partir de uma imagem de entrada. Essas representações combinam características grosseiras em alta resolução com detalhes refinados em baixa resolução, obtidas por meio da fusão de *patches* (patch merging). O modelo também emprega um MIX-FFN, composto por uma convolução 3×3 e um MLP, que integra a autoatenção com o contexto local, fornecendo indiretamente informações posicionais ao Transformer.

O *decoder* é composto exclusivamente por camadas MLP. Primeiramente, as representações em múltiplos níveis geradas pelo *encoder* passam por uma camada MLP para unificar a dimensão dos canais. Em seguida, essas representações são reamostradas para a mesma resolução espacial e concatenadas. Uma terceira camada MLP é utilizada para fundir as características concatenadas, e uma última camada MLP é responsável pela predição final do mapa de segmentação.

Figura 8 – Arquitetura do modelo Segformer

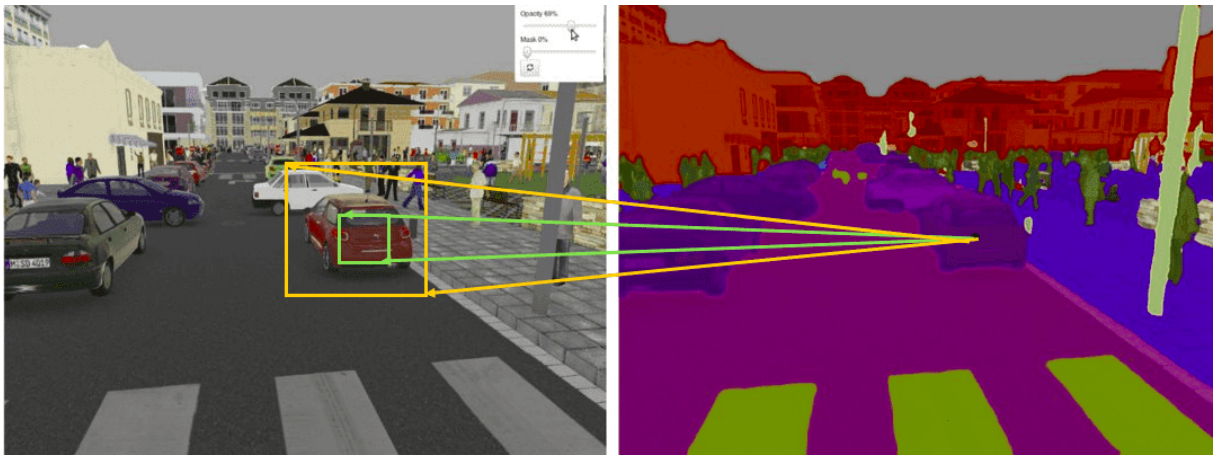


Fonte: (XIE et al., 2021)

2.3.4 BiseNet

BiseNet é um modelo proposto para segmentação semântica em tempo real que consite majoritariamente em dois caminhos: Caminho Espacial e Caminho Contextual (YU et al., 2018). A principal ideia que leva ao surgimento do modelo é balancear um campo receptivo largo para capturar maior contexto do objeto (Figura 9) e ao mesmo tempo manter informações espaciais.

Figura 9 – Diferença entre diferentes tamanhos de campos receptivos. Um campo receptivo maior (amarelo) permite maior adquirir maior contexto para os modelos de segmentação.

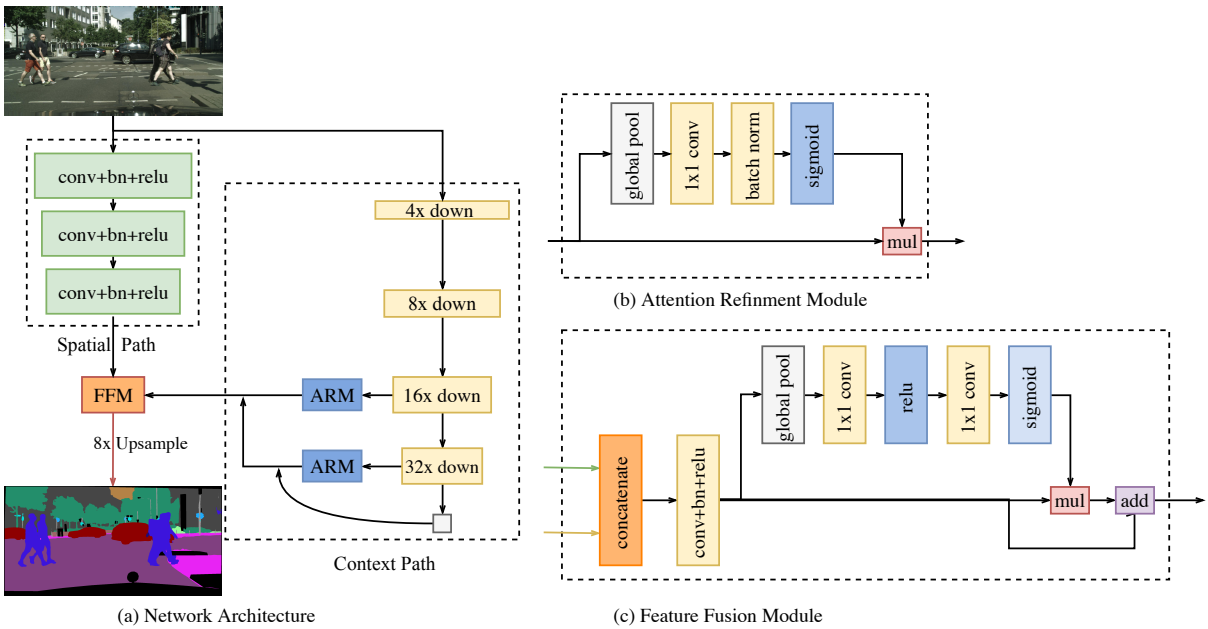


Fonte: (ADALOGLOU, 2020)

Para manter uma representação espacial detalhada, o Caminho Espacial (Spatial Path) realiza apenas algumas operações de downsampling, de modo a preservar alta resolução

e capturar informações espaciais ricas da cena. Por outro lado, o Caminho Contextual (Context Path) amplia o campo receptivo empregando o backbone da rede (como ResNet ou Xception), combinado com Global Average Pooling para obter informações globais. Essas representações contextuais são refinadas por meio do Attention Refinement Module (ARM) e posteriormente integradas ao Caminho Espacial através do Feature Fusion Module (FFM), garantindo equilíbrio entre detalhes espaciais e contexto global. A arquitetura geral do modelo é ilustrada na Figura 10.

Figura 10 – Arquitetura do modelo Bisenet



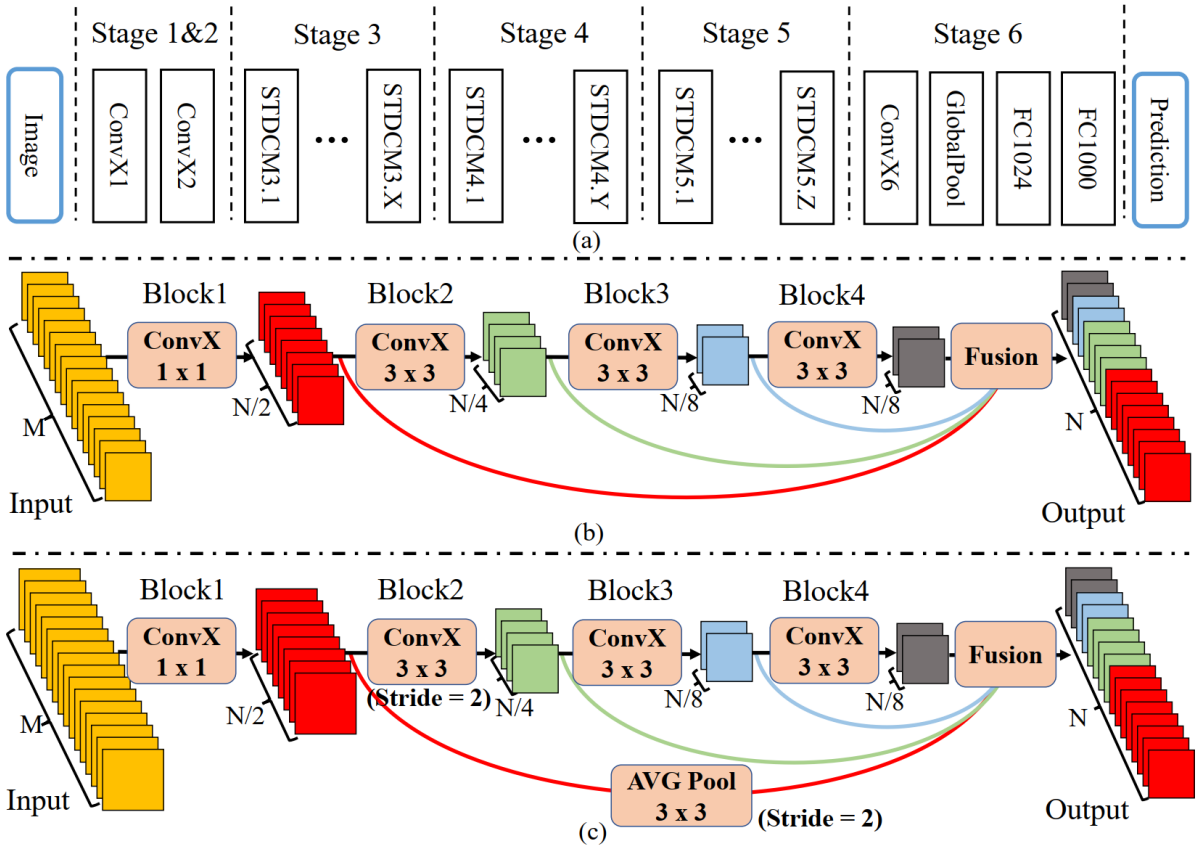
Fonte: (YU et al., 2018)

Posteriormente, os autores propuseram uma melhoria no modelo BiSeNet com o objetivo de reduzir a complexidade computacional (YU et al., 2021). Para isso, o modelo utiliza um backbone leve dividido em dois ramos: o Ramo de Detalhes, equivalente ao Caminho Espacial, e o Ramo Semântico, equivalente ao Caminho Contextual. As características extraídas em ambos os ramos são então combinadas por meio de uma camada de agregação, resultando em uma fusão eficiente das informações detalhadas e contextuais.

2.3.5 STDC

O modelo STDC é uma adaptação do BiSeNet projetada para segmentação semântica em tempo real, oferecendo uma inferência ainda mais rápida (FAN et al., 2021). Os autores unificam os dois caminhos do BiSeNet (Espacial e Contextual) e propõem um backbone próprio, baseado em blocos STDC. Além disso, o modelo reutiliza o Attention Refinement Module (ARM) e o Feature Fusion Module (FFM), incorporando também um módulo de detalhamento das características, que preserva informações espaciais importantes para a segmentação.

Figura 11 – Arquitetura dos blocos STDC.



Fonte: (FAN et al., 2021)

O módulo **STDC** (*Short-Term Dense Concatenate*) é composto por blocos **ConvX**, nos quais cada unidade contém uma convolução, seguida por normalização em lote (*Batch Normalization*) e função de ativação ReLU. A primeira convolução utiliza filtro 1×1 para reduzir a dimensionalidade, enquanto as convoluções subsequentes empregam filtros 3×3 para capturar padrões espaciais mais complexos. As saídas intermediárias de cada estágio são concatenadas de forma densa, permitindo melhor reutilização de características e enriquecimento das representações. A saída final de um bloco STDC pode ser representada como:

$$x_{\text{out}} = F(x_1, x_2, \ldots, x_n),$$

em que F corresponde ao operador de fusão das saídas parciais x_i. Essa estrutura promove um equilíbrio entre eficiência computacional e capacidade de extração de características, como ilustrado na Figura 11.

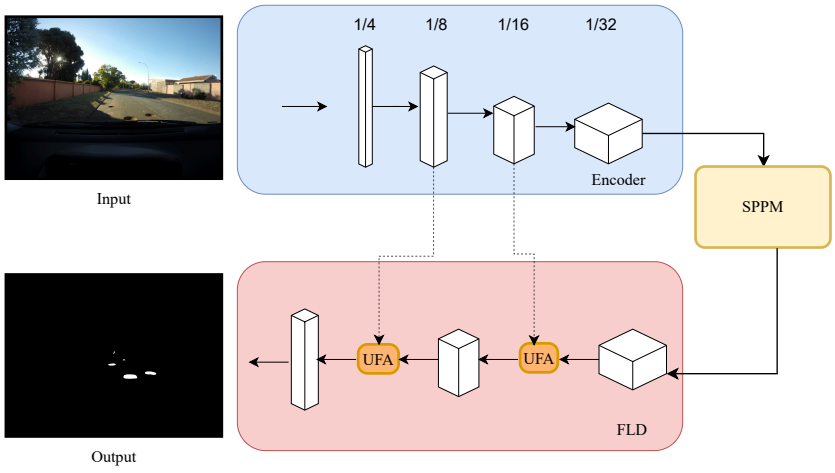
O *decoder* utiliza os blocos STDC supracitados como *backbone* e adota o mesmo conceito do Caminho Contextual do BiseNet para capturar informações globais de contexto. Além disso, atmbém incorpora dois módulos fundamentais: o *Attention Refinement Module* (ARM), responsável por refinar mapas de características e destacar regiões mais relevantes, e o *Feature Fusion Module* (FFM), que realiza a fusão eficiente entre informações contextuais

e detalhes espaciais preservados. Complementarmente, um módulo de detalhamento (*Detail Guidance Module*) auxilia na preservação de bordas e contornos finos, garantindo maior precisão nas regiões de transição entre classes. Essa combinação de componentes torna o STDC-Seg uma arquitetura altamente eficiente, alcançando um equilíbrio entre acurácia e velocidade de inferência, adequado para aplicações em tempo real.

2.3.6 PPLite

O **PP-LiteSeg** (PENG et al., 2022) é um modelo de segmentação proposto pela Baidu Inc., baseado em uma arquitetura *encoder–decoder*, que introduz dois componentes principais: o *Flexible and Lightweight Decoder* (FLD) e o *Unified Attention Fusion Module* (UAFM). Além disso, o modelo incorpora uma versão aprimorada do *Pyramid Pooling Module*, originalmente proposto por Zhao *et al.* (ZHAO et al., 2017), denominada *Simple Pyramid Pooling Module* (SPPM). A arquitetura geral do PP-LiteSeg é apresentada na Figura 12.

Figura 12 – Visão geral da arquitetura do PP-LiteSeg, adaptada de (PENG et al., 2022). As características da imagem de entrada são inicialmente extraídas pelo *encoder* e agregadas no *Simple Pyramid Pooling Module* (SPPM) para capturar o contexto global. Em seguida, o *Flexible and Lightweight Decoder* (FLD) combina informações semânticas e detalhadas, que são refinadas pelo *Unified Attention Fusion* (UAF).



Fonte: Elaborada pelo autor, adaptado de (PENG et al., 2022)

O **Flexible and Lightweight Decoder (FLD)** propõe uma estratégia para equilibrar a distribuição de custo computacional entre o *encoder* (extrator de características) e o *decoder* (responsável pela predição de segmentação). Diferente de modelos tradicionais, que mantêm o mesmo número de canais ao longo de todo o *decoder* — prática que acarreta sobrecarga computacional desnecessária e cria gargalos —, o FLD reduz gradualmente o número de canais das camadas de alto nível ao mesmo tempo em que aumenta a resolução espacial. Esse

desenho possibilita *upsampling* eficiente e predições em nível de pixel, alcançando um bom equilíbrio entre precisão e custo computacional.

O **Unified Attention Fusion (UAF)** tem como objetivo otimizar a fusão entre características de baixo e alto nível, etapa crucial para melhorar a acurácia da segmentação (LONG; SHELHAMER; DARRELL, 2015). Para isso, o módulo combina dois mecanismos de atenção complementares: a atenção espacial, que gera pesos para destacar a relevância relativa de cada pixel no mapa de características, e a atenção por canal, que atribui pesos conforme a importância de cada canal de características. A integração desses mecanismos enriquece as representações extraídas e amplia a capacidade do modelo de capturar tanto detalhes espaciais quanto o contexto semântico de alto nível.

2.4 Implantação de Redes Neurais

Modelos de Redes Neurais exigem uma grande quantidade de processamento devido ao elevado número de parâmetros e operações com ponto flutuante envolvidas em sua execução, o que impacta diretamente a latência do processamento e os requisitos de memória para armazenamento do modelo. Essas características afetam a implantação dos modelos no mundo real, especialmente em casos em que os modelos devem ser implantados em dispositivos embarcados, como no caso de ADAS, automação industrial, vigilância, robótica e entre outros. Devido a isso, diversas técnicas surgem com o propósito de adaptar modelos de Redes Neurais para sistemas com hardware restrito, descritos em três principais técnicas (ROTH et al., 2019):

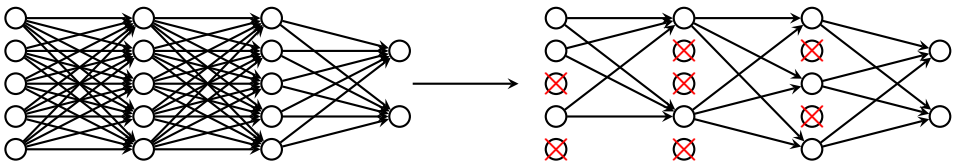
- **Quantização**: A quantização consiste em reduzir o número de bits utilizados para representar os pesos e as ativações das camadas, geralmente diminuindo de 32 bits em ponto flutuante para 16 bits em ponto flutuante ou para inteiros de 8 bits. Além de reduzir o consumo de memória, também melhora o tempo de inferência ao utilizar operações aritméticas mais baratas.

- **Poda de modelos**: Consiste na remoção seletiva de componentes da arquitetura do modelo, desde pesos individuais até camadas ou neurônios inteiros, como ilustrado na Figura 13. Essa técnica não apenas reduz a complexidade computacional e o tamanho do modelo, tornando-o mais eficiente para execução em sistemas embarcados, mas também pode atuar como uma forma de regularização, fornecendo uma melhor generalização do modelo (SRIVASTAVA et al., 2014).

- **Eficiência de estrutura**: Refere-se a um conjunto de técnicas voltadas para a construção de arquiteturas de Redes Neurais mais eficientes desde o início, ao invés de se concentrar apenas na compressão de modelos como ocorre na quantização e na poda. O principal exemplo são as convoluções separáveis em profundidade apresentadas no

MobileNet (HOWARD et al., 2017), além de outras otimizações presentes nos modelos apresentados nas seções 2.2 e 2.3.

Figura 13 – Esquema de Poda em um modelo de Rede Neural. Neurônios e conexões removidas são indicados por $\times$.



Fonte: Elaborada pelo autor.

2.4.1 Quantização

Quantização envolve utilizar tipos de dados de menor precisão para representar os pesos e as ativações de um modelo de rede neural. A quantização ocorre para dois principais tipos: FP16 e INT8 (SHAFIQUE; MUNIR; KONG, 2023).

2.4.1.1 Quantização para FP16

O processo de quantizar os pesos em *floats* de 32 bits (FP32) para *floats* de 16 bits (FP16) é simples e direto, uma vez que ambos são dados com ponto flutuante.

2.4.1.2 Quantização para INT8

A quantização para inteiros de 8 bits (INT8) é mais desafiadora, pois os valores devem ser representados com apenas 256 níveis discretos. Esse processo requer o mapeamento de valores contínuos para um intervalo finito $[\alpha, \beta]$, onde entradas fora desse intervalo são truncadas para o limite mais próximo. A determinação de α e β é realizada por meio de um procedimento de calibração (WU et al., 2020). Um dos métodos é a calibração por percentil (MCKINSTRY et al., 2019), que estima o intervalo de quantização executando um subconjunto representativo do conjunto de dados na rede e selecionando o percentil 99,99 das ativações observadas como limite máximo.

2.4.1.3 Treino com quantização

O Treino com quantização é um método que considera o impacto da quantização durante o processo de treinamento. O modelo é treinado com operações conscientes da quantização, que imitam o processo de quantização durante o treino. Dessa maneira, o cálculo do gradiente é feito considerando os pesos quantizados, de forma que o modelo evite mínimos estreitos na função de perda e convirja para mínimos mais largos e planos, menos sensíveis à discretização dos pesos (WU et al., 2020). Isso permite que o modelo aprenda a manter um

bom desempenho mesmo na representação quantizada, resultando em uma precisão superior em comparação à quantização realizada após o treinamento.

2.4.1.4 Erro de quantização

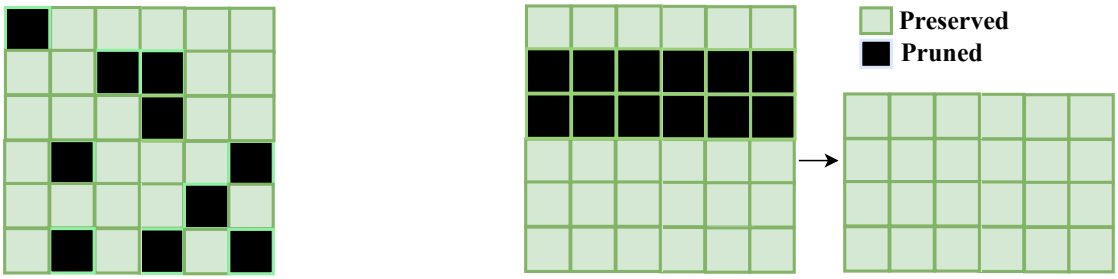
A acurácia do modelo diminui com o uso de quantização devido a erros de arredondamento, ao utilizar menos bits para representar um número, e a erros de corte, que ocorrem ao mapear valores de alta precisão para um intervalo discreto e limitado, como no caso de representações em INT8.

2.4.2 Poda de modelos (Pruning)

A poda de redes neurais é uma técnica de compressão de modelos que reduz o tamanho e a complexidade das redes neurais ao remover sistematicamente pesos ou unidades de ativação menos significativos (CUN; DENKER; SOLLA, 1989). Duas abordagens principais são comumente empregadas (Figura 14): poda não estruturada (*unstructured*), que remove pesos individuais abaixo de um determinado limiar (HAN et al., 2015), e poda estruturada (*structured*), que elimina estruturas inteiras, como filtros, canais ou camadas (ANWAR; HWANG; SUNG, 2017).

A poda não estruturada oferece maior flexibilidade e pode alcançar maiores taxas de compressão, já que qualquer peso pode ser removido. No entanto, é desafiadora de acelerar na prática, pois exige hardware especializado para explorar de forma eficiente representações esparsas (KUZMIN et al., 2023). Já a poda estruturada remove estruturas computacionais bem definidas, tornando mais fácil obter aceleração real em hardware de uso geral, embora com custo de flexibilidade reduzida em comparação à poda não estruturada.

Figura 14 – Comparação de *pruning* não estruturado (esquerda) e estruturado (direita). *Pruning* não estruturado tem mais flexibilidade e maior taxa de compreesão, contudo necessita de *hardware* específico para aceleração. *Pruning* estruturado, por sua vez, tem menos flexibilidade mas permite aceleração independente da plataforma.



Fonte: Elaborada pelo autor.

No método de *group pruning* proposto no algoritmo DepGraph (FANG et al., 2023), o processo de poda é guiado pelas dependências estruturais entre as camadas. Primeiramente,

constrói-se um grafo de dependências para organizar as camadas em grupos de acordo com suas inter-relações. Dentro de cada grupo, um escore de importância $I(w)$ é atribuído a cada parâmetro, tipicamente utilizando a norma L2. A importância do grupo $I(g)$ é então calculada pela agregação dos escores de seus parâmetros. Para induzir esparsidade em nível de grupo, realiza-se uma fase de treinamento esparso, que gradualmente leva parâmetros pouco relevantes a valores próximos de zero. Após essa etapa, escores relativos de importância são utilizados para identificar quais grupos podem ser removidos de forma segura da rede, reduzindo a complexidade do modelo sem comprometer o desempenho.

3 Revisão Bibliográfica

Este capítulo apresenta uma revisão de trabalhos relacionados da literatura, com foco na detecção e segmentação de buracos em vias. O conteúdo está organizado em duas seções principais: a primeira aborda estudos voltados para a detecção de buracos, enquanto a segunda explora pesquisas direcionadas à segmentação de buracos.

3.1 Detecção de buracos em vias

Gupta et al. (GUPTA et al., 2020) propõem uma abordagem inovadora para a detecção de buracos utilizando imagens térmicas, empregando versões modificadas do ResNet34 com SSD e do ResNet50 com RetinaNet. Esses modelos alcançaram altas taxas de precisão, de $74,93\%$ e $91,15\%$, respectivamente. No entanto, não foram fornecidas informações sobre o tempo de inferência, o que deixa incerta sua aplicabilidade em tempo real. Embora os resultados sejam promissores, a dependência de imagens térmicas limita a viabilidade da solução para aplicações de baixo custo, já que a maioria das câmeras acessíveis não possui essa funcionalidade.

Maeda et al. (MAEDA et al., 2018) apresentam um conjunto de dados composto por 9053 imagens anotadas para detectar uma ampla variedade de danos em vias, capturadas por uma câmera de smartphone montada próxima ao veículo. A avaliação foi realizada utilizando o modelo SSD com as arquiteturas InceptionV2 e MobileNet como backbones. Devido ao foco na detecção de múltiplos tipos de danos, o *RECALL* para detecção de buracos foi baixo: $0,03$ para o InceptionV2 e $0,02$ para o MobileNet. O tempo de inferência do modelo SSD com MobileNet no smartphone foi de 100 ms, tornando-o adequado para detecção em tempo real. No entanto, o desempenho do modelo é insatisfatório para a detecção exclusiva de buracos, como evidenciado pelas baixas taxas de *recall*, provavelmente devido ao desequilíbrio entre classes e à dificuldade em identificar uma ampla gama de danos.

A. Abu-Raddaha et al. (ABU-RADDAHA; EL-SHAIR; RAWASHDEH, 2024) utilizam o modelo YOLOv5 para detecção de buracos em um conjunto de dados contendo 3777 imagens, empregando uma técnica de transformação de perspectiva para ajustar o ponto de vista das imagens. O método proposto atinge um mAP50 de $68,8\%$ e um *recall* médio de $38,5\%$ com a versão pequena do YOLO. Contudo, o modelo utiliza imagens com resolução de $800x800$, o que, somado à sobrecarga da transformação de perspectiva, aumenta o tempo de inferência. Os próprios autores ressaltam a necessidade de otimizações para tornar o método viável para uso em tempo real.

Outros trabalhos apresentam detecção de buraco em um contexto de tempo real.

Asad *et al.* (ASAD et al., 2022b) utilizaram modelos leves de detecção de objetos implantados em um Raspberry Pi 4B e posteriormente os otimizaram exportando para o OpenVINO, visando melhor desempenho em hardware embarcado. De forma semelhante, Musa *et al.* (MUSA et al.,) empregaram *knowledge distillation*, uma técnica amplamente conhecida para redução de parâmetros de modelos, a fim de implementar CNNs profundas para classificação de imagens de buracos em sistemas embarcados

3.2 Segmentação de buracos em vias

Diversos estudos na literatura exploram a aplicação de segmentação semântica para detecção de buracos. Fan *et al.* (FAN et al., 2020) propuseram uma transformação de disparidade para melhorar a discriminação entre regiões danificadas e não danificadas, combinada com uma estratégia de adaptação de domínio adversarial para converter imagens RGB de estradas em representações de disparidade. Entretanto, sua abordagem foi projetada principalmente para inspeção de vias, e não para sistemas de assistência ao condutor (ADAS), uma vez que se baseia em imagens aéreas ao invés da perspectiva veicular necessária em aplicações de ADAS. Em trabalhos subsequentes, os autores incorporaram ainda uma *Graph Attention Layer* para otimizar as representações de características para segmentação semântica, alcançando melhorias significativas de desempenho (FAN et al., 2021). Contudo, ambas as abordagens permanecem focadas na inspeção de vias com imagens aéreas, em vez de sistemas embarcados voltados para ADAS.

Katsamenis *et al.* (YANG et al., 2023) empregaram a arquitetura baseada em *transformers* SegFormer para segmentação semântica de buracos. De forma semelhante, métodos como ERCU-Net (TRIPATHI; INDU; KUMAR, 2024), MAFNet (FENG et al., 2022), e o framework proposto por Akagic *et al.* (AKAGIC; BUZA; OMANOVIC, 2017) obtiveram resultados promissores. No entanto, essas abordagens continuam focadas principalmente em inspeção de vias com imagens aéreas.

Masihullah *et al.* (MASIHULLAH et al., 2021) propuseram um framework acoplado baseado em atenção para segmentação conjunta de estrada e buracos, construído sobre a arquitetura DeepLabV3+. Seu framework alcançou 93.6% de IoU para segmentação de estradas e 73.8% de IoU para segmentação de buracos. Entretanto, a avaliação para detecção de buracos foi realizada em um conjunto de dados relativamente pequeno, com apenas 237 imagens, o que limita a generalização dos resultados.

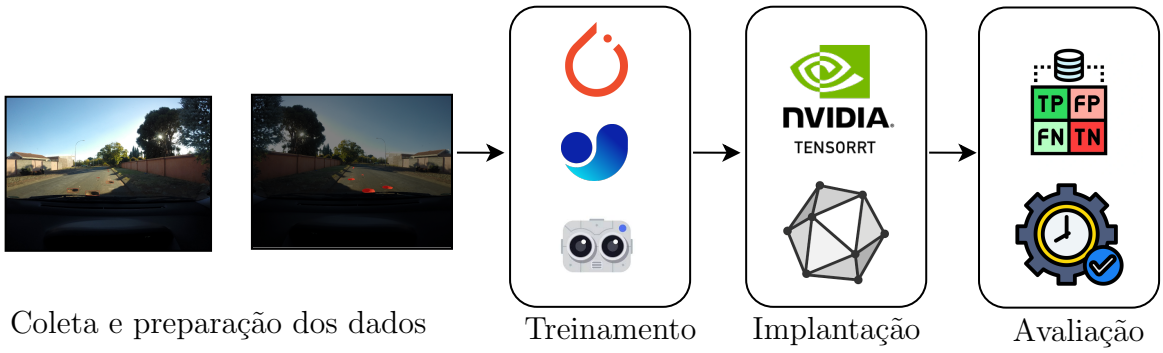
4 Metodologia

Este capítulo abordará a metodologia proposta para a detecção e segmentação de buracos em vias para sistemas embarcados de baixo custo.

Conforme apresentado na Figura 15, a metodologia deste trabalho consiste em:

1. **Coleta e preparação do conjunto de dados:** Consiste na coleta e preparação do conjunto de dados para o treinamento dos modelos de detecção e dos modelos de segmentação;
2. **Treinamento dos modelos**: Consiste no treinamento e ajuste dos modelos de segmentação e dos modelos de detecção no conjunto de dados formado;
3. **Implantação**: Aplicar as técnicas de adaptação dos modelos para sistemas embarcados de baixo custo;
4. **Avaliação e testes**: Avaliação dos resultados dos modelos e exequibilidade para contexto ADAS.

Figura 15 – Metodologia do trabalho proposto e as ferramentas utilizadas em cada etapa.



Fonte: Elaborada pelo autor.

4.1 Coleta e Preparação do Conjunto de Dados

O conjunto de dados preparado para o treinamento consiste na união de dois conjuntos de dados apresentados na literatura e, também, capturados em ruas brasileiras, em carros pessoais e, também, junto ao projeto em que estamos colaborando com o ICMC/USP, o projeto CaRINA (Carro Robótico Inteligente de Navegação Autônoma) (BRUNO et al., 2023).

O primeiro, apresentado por (NIENABER; BOOYSEN; KROON, 2015), consiste em 3777 imagens extraídas de filmagens de vídeo capturadas com uma câmera GoPro montada no veículo. O conjunto de dados é um dos poucos que são adequados para aplicações em dispositivos de borda, pois as imagens fornecem uma perspectiva em primeira pessoa do ponto de vista do veículo. Das 3777 imagens, 1119 contêm buracos e 2658 são imagens de fundo. Essa proporção é esperada; ruas regulares geralmente não têm muitos buracos, pois eles são resultado de um gerenciamento inadequado. No entanto, o desequilíbrio de classes adicionou uma dificuldade, pois a maioria dos modelos tende a apresentar viés em direção à classe majoritária (JOHNSON; KHOSHGOFTAAR, 2019; CHEN et al., 2020). Para resolver o problema do desequilíbrio de classes no conjunto de dados, foram aplicadas técnicas de aumento de dados. Parte das imagens negativas (aquelas sem anotações) foi excluída, enquanto as imagens positivas foram aprimoradas por meio de inversão horizontal para simular diferentes perspectivas das faixas.

O segundo conjunto de dados, proposto por (JAKUBEC et al., 2023), é composto por 2099 imagens, divididas entre diferentes condições climáticas — clima limpo, clima chuvoso, nascer do sol, pôr do sol e noite — permitindo que o modelo se adapte a uma variedade de cenários do mundo real. As imagens são capturadas do ponto de vista do veículo, o que as torna adequadas para sistemas ADAS. Para aumentar a diversidade do conjunto de dados, as imagens foram aprimoradas com a técnica de inversão horizontal, simulando diferentes perspectivas das faixas.

Com a união dos dois conjuntos de dados, o conjunto final utilizado para treinamento e inferência consiste em **5602** imagens (4156 com buracos + 1446 sem buracos) no conjunto de treinamento e **1375** (1057 com buracos + 318 sem buracos) no conjunto de validação.

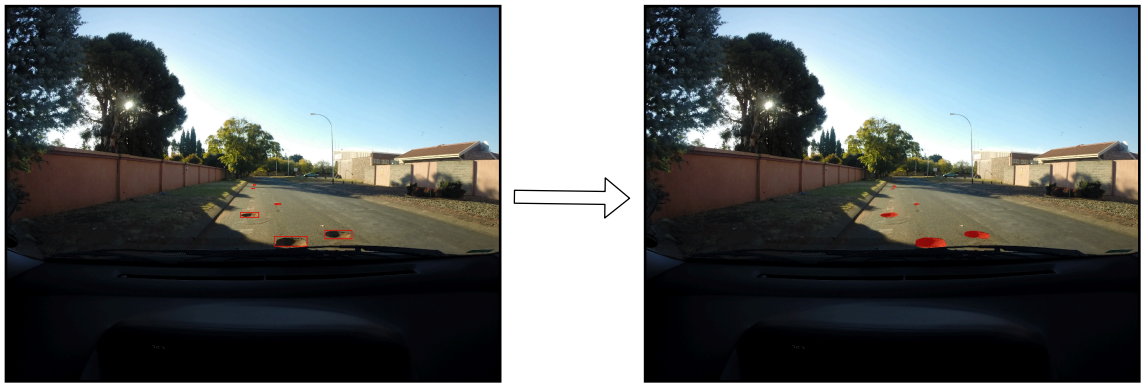
O conjunto de dados possui anotações das caixas delimitadoras, sendo possível o treinamento de detectores de objeto. A fim de adaptar o conjunto de dados para a tarefa de segmentação, será usado o modelo *Segment Anything Model* 2 (RAVI et al., 2024; ZHAO; ZHANG; QI, 2024) para a criação automática das máscaras de segmentação. Na Figura 16 é apresentado um exemplo de imagens convertidas das caixas delimitadoras para máscaras de segmentação.

4.2 Treinamento dos Modelos

Os modelos de detecção de objetos **YOLO**, **DFINE** e **RT-DETR**, descritos na Seção 2.2, foram treinados e avaliados nos *datasets* apresentados na seção anterior. Já os modelos de **Segmentação Semântica**, detalhados na Seção 2.3, foram treinados a partir das máscaras geradas pelo *Segment Anything Model 2* (SAM2), utilizando como base as caixas delimitadoras do conjunto de dados original.

Para a segmentação, os modelos foram agrupados em dois conjuntos:

Figura 16 – Anotação automática de caixas delimitadoras para máscaras de segmentação através do modelo Segment Anything Model (RAVI et al., 2024).



Fonte: Elaborada pelo autor.

- **Genéricos** — *U-Net*, *DeepLab* e *Segformer*;
- **Otimizado para tempo real** — *BiSeNet*, *STDC* e *PPLite*.

Nos modelos genéricos, testaram-se diferentes extratores de características, conforme descrito na Seção 2.1, enquanto nos modelos voltados para tempo real manteve-se a implementação original.

4.3 Avaliação e Implantação

Para a avaliação do modelo treinado, foram utilizadas diversas métricas empregadas em tarefas de detecção de objetos (PADILLA; NETTO; SILVA, 2020) e segmentação semântica (WANG et al., 2023), como:

IoU (Intersection Over Union): Mede a sobreposição entre a caixa ou máscara predita e a caixa ou máscara de referência (real), sendo calculado pela razão entre a área de interseção e a área da união dessas duas regiões.

Precisão: Proporção de predições positivas corretas em relação a todas as predições positivas realizadas pelo modelo. Em outras palavras, dentre todas as regiões classificadas como buracos, indica a porcentagem que realmente corresponde a buracos.

Recall: Mede a capacidade do modelo de identificar corretamente instâncias positivas. Representa a porcentagem de buracos reais que foram detectados pelo modelo.

F1-score: Média harmônica entre a Precisão e o Recall, proporcionando uma métrica equilibrada que considera simultaneamente falsos positivos e falsos negativos.

AP50: Média da precisão (Average Precision) calculada considerando um limiar de IoU de 0,50, ou seja, uma predição é considerada correta se a sobreposição com a anotação real for de pelo menos 50%. Essa métrica é amplamente utilizada em tarefas de detecção de objetos para avaliar a qualidade das predições.

Acurácia: Proporção de predições corretas (tanto positivas quanto negativas) em relação ao total de amostras avaliadas. Indica a taxa global de acertos do modelo, sem diferenciar entre classes.

Coeficiente de Dice: Métrica de similaridade amplamente utilizada em segmentação de imagens, definida como o dobro da área de interseção entre a predição e a referência dividido pela soma das áreas individuais. É particularmente sensível a regiões de pequeno tamanho e correlaciona-se fortemente com o IoU.

Por fim, nos melhores modelos treinados, foram aplicadas técnicas para implantação do modelo treinado como exportação com TensorRT, quantização pós-treino e poda de modelos.

4.4 Ferramentas Computacionais

Para o desenvolvimento deste trabalho, diversas ferramentas foram empregadas. Na etapa de coleta e preparação dos dados, utilizou-se a linguagem de programação `Python`, com destaque para a biblioteca `OpenCV`. Para o treinamento dos modelos, foi utilizada a biblioteca `PyTorch`, em conjunto com os *frameworks* `Ultralytics`, `Super Gradients` e `Detectron`. Por fim, na fase de implantação, recorreu-se à ferramenta `TensorRT` para otimização e quantização, bem como ao `ONNX` para exportação dos modelos em um formato interoperável e compartilhável. Para o *pruning* dos modelos, foi utilizado a biblioteca `TorchPrune`.

5 Resultados

Esse capítulo descreverá os resultados obtidos na metodologia proposta neste trabalho. Na seção 5.1 é apresentado os resultados dos modelos de detecção de objetos e respectiva quantização; na seção 5.2 são apresentados os resultados os resultados de extratores de características para os modelos de segmentação descritos na seção 2.1; e, por fim, são apresentados os resultados dos modelos de segmentação descritos em 5.3 e experimentos com poda de modelos.

5.1 Detecção de Objetos

Os modelos de detecção de objetos empregados para a detecção de buracos em tempo real foram os modelos Yolo, RT-DETR e D-FINE, descritos na seção 2.2. As configurações do modelo são apresentados na Tabela 3.

Tabela 3 – Configurações dos modelos usados para a tarefa de Detecção de Objetos.

Modelo	**Backbone**	**Parâmetros**	**FLOPs**
RT-DETR	ResNet-50	42M	136 GFLOPs
YOLO11n	Backbone Customizado	2.6M	6.5 BFLOPs
DFINE	HGNetV2	4M	7 GFLOPs

Fonte: Elaborada pelo autor.

Os modelos foram treinados por 500 épocas, com o uso de *Early Stopping*, empregando os mesmos hiperparâmetros e estratégias de aumento de dados propostos em seus respectivos trabalhos originais. A Tabela 4 contém os resultados obtidos durante o treinamento dos modelos no conjunto de dados proposto, avaliados na placa NVIDIA RTX 3060.

Tabela 4 – Desempenho dos modelos de detecção no conjunto de dados proposto. Os valores de FPS foram obtidos a partir do tempo médio de inferência.

Modelo	**mAP@50**	**Recall@50**	**Tempo (s)**	**FPS**
RT-DETR-v2	0.765	0.495	0.034	29
YOLOv11n	0.799	0.724	**0.0018**	**555**
D-FINE	**0.918**	**0.961**	0.020	50

Fonte: Elaborada pelo autor.

O modelo D-FINE apresentou desempenho superior aos demais, alcançando um ganho de aproximadamente 11% em precisão em relação ao segundo melhor modelo. Esse resultado se deve principalmente ao refinamento iterativo das caixas delimitadoras proposto pelo módulo FDR (Fine-Grained Distribution Refinement), que realiza ajustes sucessivos e precisos nas

bordas das predições. Essa abordagem mostra-se particularmente eficaz na detecção de objetos pequenos, em que pequenos deslocamentos das caixas geram variações significativas no valor do Intersection over Union (IoU) (CHEN et al., 2022). Como o FDR refina as caixas com deslocamentos proporcionais à sua altura e largura originais, o modelo consegue corrigir e alinhar melhor as predições, resultando em maior precisão espacial.

O modelo YoloV11 apresentou desempenho superior em comparação ao RT-DETR, especialmente na métrica de Recall. O backbone do YoloV11 incorpora melhorias por meio dos módulos C2PSA e C3K2, detalhados na Seção 2.2.1, que contribuem significativamente para a detecção de objetos pequenos — uma limitação conhecida do RT-DETR (ZHAO et al., 2024). Além disso, o modelo Yolo apresentou o tempo de inferência mais rápido devido ao seu menor número de parâmetros e flops.

A fim de adaptar o melhor modelo (D-FINE) para detecção em tempo real, foi aplicado a técnica de quantização (descritas na seção 2.4.1), transformando os pesos de FP32 para FP16 e INT8. A quantização para INT8 foi realizada utilizando o método de *percentile calibration* (MCKINSTRY et al., 2019) utilizando otimizações específicas para placas `NVIDIA` através do motor `NVIDIA TensorRT`. A Tabela 5 contém os resultados das métricas mAP@50 e Recall@50 para precisões de FP16 e INT8. É possível observar uma queda maior na precisão durante a conversão para INT8, enquanto não houve uma queda proporcional no Recall.

Tabela 5 – Desempenho do modelo D-FINE em diferentes precisões. Valores entre parênteses indicam a diferença em relação ao baseline FP32.

Métrica	**FP32 (%)**	**FP16 (%)**	**INT8 (%)**
mAP@50	91.8	90.2 (-1.6)	78.8 (-13.0)
Recall@50	96.1	95.5 (-0.6)	94.2 (-1.9)

Fonte: Elaborada pelo autor.

A Tabela 6 contém uma análise detalhada do desempenho de inferência, considerando a latência total — incluindo transferência de dados entre host e dispositivo e tempo de processamento na GPU — e o *throughput*, calculado como o número total de consultas dividido pelo tempo total de execução. O tempo de inferência é medido após realizar a exportação e otimização do modelo para placas `NVIDIA` através do motor `TensorRT`. O *speedup* obtido com FP16 é superior ao alcançado pelo INT8, pois algumas camadas do modelo — especialmente aquelas que envolvem mecanismos de atenção — não possuem suporte nativo para operações em INT8 no `TensorRT`. Como consequência, essas camadas precisam ser convertidas para FP16 ou FP32 durante a inferência, o que adiciona overhead e reduz o ganho de desempenho potencial da quantização INT8.

Tabela 6 – Comparação do desempenho de inferência do modelo D-FINE em diferentes precisões. Valores entre parênteses indicam *speedup* em relação ao baseline FP32.

Métrica	**FP32**	**FP16**	**INT8**
Latência (ms)	3.48	2.25 **(1.55x)**	2.34 **(1.49x)**
GPU Compute Time (ms)	3.03	1.80 **(1.68x)**	1.95 **(1.55x)**
Throughput (QPS)	329	555 **(1.69x)**	513 **(1.56x)**

Fonte: Elaborada pelo autor.

5.2 Extratores de Características para Modelos Genéricos de Segmentação

Foram empregados modelos genéricos de segmentação de objetos e avaliados utilizando *backbones* de baixo custo, citados na seção 2.1. Os modelos empregados foram Unet++, DeepLabv3+ e Segformer, descritos nas seção 2.3. Os modelos foram treinados a 500 épocas, utilizando a estratégia de *Early Stopping* para evitar sobreajuste no treinamento. O otimizador empregado foi o Adam, com Taxa de Aprendizado inicial de 1^{-5} e decaindo até 1^{-6} utilizando o recozimento de cosseno (LOSHCHILOV; HUTTER, 2017). Todos os modelos foram treinados com imagem 640×640 e utilizando como *data augmentation* a composição de ShiftScaleRotate e AtLeastOneBBoxRandomCrop, apresentados pela biblioteca Albumentations (BUSLAEV et al., 2020). A implementação foi feita utilizando a biblioteca SMP (IAKUBOVSKII, 2019) e timm (WIGHTMAN, 2019) do Python.

A Tabela 7 contém os resultados do modelo U-net++ com diferentes *backbones*. Os melhores resultados considerando a métrica Dataset IoU foram obtidos pelo *backbone* EfficientNet-b2. Os outros modelos possuem métricas bem similares, variando cerca de 1%, exceto pelo caso do ResNet18 que possuiu a pior performance com aproximadamente 7% a menos do que o EfficientNet-b2. Isso se deve ao fato do ResNet18 ser o modelo mais antigo, não incorporando as evoluções arquitetônicas que os outros *backbones* possuem. MobileNet_v2 possuiu o tempo de inferência mais rápido, isso se deve especialmente pela sua arquitetura otimizada para dispositivos embarcados (HOWARD et al., 2017).

A Tabela 8, por sua vez, contém os resultados do modelo DeepLab com diferentes *backbones*. O melhor resultado obtido foi pelo modelo MobileVit, que sintetiza melhor com as convoluções múltiplas em paralelo do DeepLab (AZAD et al., 2022). ResNet18 obteve o pior resultado devido aos mesmos motivos citados na análise do Unet++. Da mesma forma, MobileNet_v2 obteve o tempo de inferência mais rápido. O HRNet não possui suporte para convoluções dilatadas devido à sua fusão de características em múltiplas resoluções diferentes.

Por fim, a Tabela 9 contém os resultados do modelo Segformer. Sua arquitetura se

Tabela 7 – Desempenho do modelo Unet++ de acordo com o *backbone* utilizado.

Backbone	IoU	Acurácia	Precisão	Recall	F1-score
EfficientNet-b2	**0.6505**	0.993	0.8300	0.8031	0.7882
MobileNet_v2	0.6208	0.990	0.8022	0.7814	0.7660
HRNet	0.6370	0.990	0.8100	0.7800	0.7700
ResNet_18	0.5832	0.990	0.7895	0.7618	0.7300
MobileVit_v2	0.6300	0.990	0.8000	0.7892	0.7700

Fonte: Elaborada pelo autor.

Tabela 8 – Desempenho do modelo DeepLab de acordo com diferentes backbones, considerando apenas as métricas principais de segmentação.

Backbone	IoU	Acurácia	Precisão	Recall	F1-score
EfficientNet-b2	0.6419	0.993	0.8288	0.8153	0.7817
MobileNet_v2	0.5920	0.990	0.8123	0.7585	0.7438
HRNet			*não suportado*		
ResNet_18	0.5870	0.990	0.7772	0.7656	0.7395
MobileVit_v2	**0.6752**	0.990	0.8398	0.8272	0.8061

Fonte: Elaborada pelo autor.

destaca por utilizar um decodificador leve, baseado em MLPs, que o torna quase inteiramente dependente da qualidade das características extraídas pelo *backbone*, no qual se concentra boa parte das inovações do modelo. Em nítido contraste com os outros modelos analisados, onde o decodificador desempenha um papel mais ativo no refinamento do resultado, o Segformer testa a capacidade pura do extrator de características. Isso explica por que arquiteturas mais robustas como a EfficientNet-b2 e a HRNet alcançaram os melhores índices de desempenho, enquanto backbones mais leves, como MobileNet_v2 e ResNet_18, apresentaram uma performance inferior. Novamente o MobileNet_v2 obteve o tempo de inferência mais rápido.

Tabela 9 – Desempenho do modelo Segformer de acordo com diferentes backbones, considerando apenas as métricas principais de segmentação.

Backbone	IoU	Acurácia	Precisão	Recall	F1-score
EfficientNet-b2	**0.6620**	0.9940	0.8255	0.8063	0.7966
MobileNet_v2	0.5670	0.9991	0.7533	0.7828	0.7239
HRNet	0.6004	0.9900	0.8000	0.7680	0.7503
ResNet_18	0.5810	0.9990	0.7953	0.7500	0.7349
MobileVit_v2	0.5600	0.9991	0.7959	0.7344	0.7173

Fonte: Elaborada pelo autor.

Em relação ao tempo de inferência, tanto o Segformer quanto o DeepLab apresentaram tempos relativamente consistentes entre os diferentes *backbones*, todos em torno de 0,01 segundos por imagem. Já o Unet++ registrou os piores tempos de inferência, o que pode ser atribuído principalmente à presença de conexões densas e profundas entre os blocos do *encoder* e *decoder*. Essa estrutura, embora beneficie a propagação de características de forma

independente do *encoder*, como é possível perceber devido à maior consistência de resultados altos na Tabela 7, aumenta significativamente a complexidade computacional do modelo. A Tabela 10 apresenta uma comparação geral dos melhores resultados de cada modelo de segmentação considerando a métrica Dataset IoU, e a Tabela 11 em relação ao tempo médio de inferência.

Tabela 10 – Comparativo dos melhores resultados para cada arquitetura com seu respectivo *backbone*, incluindo tempo de inferência médio por imagem.

Modelo + Backbone	**IoU**	**Acurácia**	**Precisão**	**Recall**	**F1-score**	**Inferência (s)**
Unet++ + EfficientNet-b2	0.6505	0.993	0.8300	0.8031	0.7882	0.0313
DeepLab + MobileViT_v2	**0.6752**	**0.994**	**0.8398**	**0.8272**	**0.8061**	**0.0114**
Segformer + EfficientNet-b2	0.6620	0.994	0.8255	0.8063	0.7966	0.0178

Fonte: Elaborada pelo autor.

Tabela 11 – Comparativo dos modelos com menor tempo de inferência em cada arquitetura, mantendo apenas as métricas principais e tempo de inferência.

Modelo + Backbone	**IoU**	**Acurácia**	**Precisão**	**Recall**	**F1-score**	**Inferência (s)**
Unet++ + MobileNet_v2	**0.6208**	0.991	0.8022	0.7814	**0.7660**	0.0192
DeepLab + MobileNet_v2	0.5920	0.990	**0.8123**	0.7585	0.7438	**0.0092**
Segformer + MobileNet_v2	0.5670	0.991	0.7533	**0.7828**	0.7239	0.0106

Fonte: Elaborada pelo autor.

Dessa maneira, conclui-se que MobileNet segue como sendo o principal extrator de características de baixo custo em relação ao balanceamento entre o resultado do modelo e métricas de desempenho. Contudo, modelos ViT adaptados como o MobileViT apresentam também resultados competitivos para extração de características em tempo real.

5.3 Modelos Específicos para Segmentação em Tempo Real

Após a avaliação de modelos genéricos de segmentação de objetos, foram empregados modelos estado da arte para segmentação semântica em tempo real: BiseNet, STDC e PPLITE, conforme descritos na seção 2.3. Os modelos foram treinados utilizando o framework de deep learning SuperGradients e MMSegmentation, inicializados com pesos pré-treinados no dataset Cityscapes. A otimização foi realizada com o otimizador Adam, partindo de uma taxa de aprendizado inicial de 1e-5, decaída dinamicamente por meio de um agendador de cosseno até 1e-6. O treinamento foi conduzido minimizando a função de perda Binary Cross Entropy Dice Loss, com tamanho de lote igual a 8, durante até 500 épocas, com *early stopping* baseado na métrica de IoU para a classe de buracos. Para aumentar a robustez e a capacidade de generalização dos modelos, aplicaram-se técnicas de aumento de dados, incluindo Random Erasing (ZHONG et al., 2017) e transformações Affine (SANDRU; GEORGESCU; IONESCU, 2022). O treinamento e inferência foram realizadas em uma placa NVIDIA RTX 3060.

A Tabela 12 contém os resultados dos modelos na resolução 640×640. O modelo PPLite obteve o melhor desempenho geral, alcançando as maiores métricas de IoU, Recall e F1-score. Esse resultado se deve ao fato de o PPLite herdar elementos estruturais do STDC — como o uso de blocos STDC — ao mesmo tempo em que incorpora um decodificador mais robusto, capaz de refinar melhor as predições de segmentação. Por outro lado, o BiseNetv2 apresentou o pior desempenho, resultado esperado por se tratar de uma arquitetura mais antiga e simplificada, que não aproveita técnicas mais recentes de balanceamento entre eficiência e precisão. Já o STDC-2 demonstrou métricas intermediárias, destacando-se pela maior precisão entre os modelos, mas com recall inferior ao do PPLite, o que compromete sua capacidade de detecção em cenários práticos.

A Tabela 13 contém os resultados dos mesmos modelos, mas considerando a resolução 1024×1024. Observa-se que o aumento da resolução proporcionou melhorias consistentes em todas as métricas de segmentação, indicando que os modelos conseguem explorar mais informações espaciais quando recebem imagens de maior detalhamento. Contudo, esse ganho vem acompanhado de um custo computacional elevado, refletindo em maior tempo de inferência e maior consumo de memória, aspectos que serão discutidos posteriormente.

Tabela 12 – Desempenho dos modelos na resolução 640×640, considerando as métricas principais de segmentação.

Modelo	**IoU**	**Acurácia**	**Precisão**	**Recall**	**F1-score**
BiseNetv2	53.93%	**99.98%**	75.1%	65.67%	70.07%
STDC-2	61.60%	99.84%	**83.23%**	69.23%	75.59%
PPLite	**62.54%**	99.84%	81.39%	**72.78%**	**76.85%**

Fonte: Elaborada pelo autor.

Tabela 13 – Desempenho dos modelos na resolução 1024×1024, considerando as métricas principais de segmentação.

Modelo	**IoU**	**Acurácia**	**Precisão**	**Recall**	**F1-score**
BiseNet	58.97%	**99.98%**	78.90%	70.90%	74.19%
STDC	64.73%	99.84%	83.78%	73.24%	78.15%
PPLite	**66.71%**	99.84%	**84.37%**	**75.99%**	**79.96%**

Fonte: Elaborada pelo autor.

Após o treinamento, o tempo de inferência dos modelos foi avaliado considerando seus melhores desempenhos. A Tabela 14 contém esses resultados. Observa-se que o modelo PPLite apresenta os menores tempos de inferência, resultado de suas melhorias arquiteturais em relação aos demais modelos. Já os modelos STDC e BiseNet exibem tempos de inferência semelhantes, devido à similaridade em suas estruturas. Além disso, o aumento da resolução resultou em um incremento no tempo de inferência, evidenciando um trade-off entre a melhora de desempenho nas métricas de segmentação e a eficiência computacional.

Tabela 14 – Tempo médio de inferência dos modelos nas resoluções 640×640 e 1024×1024, em ms, com FPS entre parênteses.

Modelo	**640×640**	**1024×1024**
BiseNetv2	8.93 ms (112.02 FPS)	20.12 ms (49.7 FPS)
STDC-2	8.63 ms (115.85 FPS)	19.28 ms (51.9 FPS)
PPLite	5.55 ms (180.18 FPS)	10.41 ms (96.1 FPS)

Fonte: Elaborada pelo autor.

Após o treinamento, foi aplicada quantização ao modelo PP-LiteSeg na resolução 1024×1024 visando melhorar a eficiência da inferência do modelo que obteve o melhor resultado. A quantização INT8 foi realizada utilizando o método de *percentile calibration* (MCKINSTRY et al., 2019). O modelo treinado foi inicialmente exportado para o formato `ONNX` e, em seguida, otimizado com `NVIDIA TensorRT`, permitindo inferência de baixa latência em sistemas com GPU da NVIDIA.

A Tabela 15 contém uma análise detalhada do desempenho de inferência, considerando a latência total — incluindo transferência de dados entre host e dispositivo e tempo de processamento na GPU — e o *throughput*, calculado como o número total de consultas dividido pelo tempo total de execução. O desempenho de segmentação do modelo em diferentes precisões também é resumido na Tabela 16.

Tabela 15 – Comparação do desempenho de inferência em diferentes precisões com entrada de 1024×1024. Valores entre parênteses indicam aceleração relativa ao baseline FP32.

Métrica	**FP32**	**FP16**	**INT8**
Latência (ms)	6.72	3.42 **(1,96x)**	2.54 **(2,65x)**
GPU Compute Time (ms)	5.84	2.53 **(2,31x)**	1.65 **(3,54x)**
Throughput (QPS)	170	393 **(2,31x)**	603 **(3,55x)**

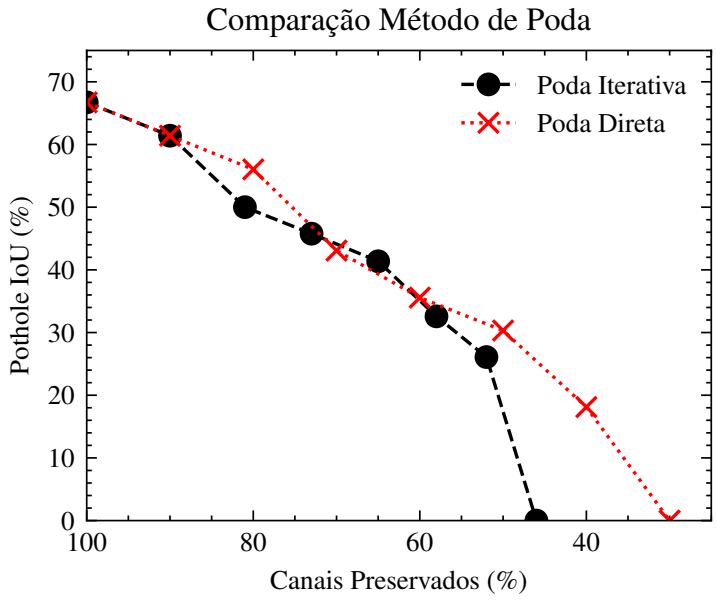
Fonte: Elaborada pelo autor.

Tabela 16 – Desempenho do modelo PP-LiteSeg a 1024×1024 em diferentes precisões. Valores entre parênteses indicam a diferença absoluta em pontos percentuais em relação ao baseline FP32.

Métrica	**FP32 (%)**	**FP16 (%)**	**INT8 (%)**
IoU	66.71	66.71 (0.00)	64.64 (-2,07)
Acurácia	99.84	99.84 (0.00)	99.83 (-0.01)
Precisão	84.37	84.37 (0.00)	80.41 (-3.96)
Recall	75.99	75.97 (-0.02)	76.75 (+0.76)
F1-score	79.96	79.94 (-0.02)	78.53 (-1.43)

Fonte: Elaborada pelo autor.

Figura 17 – Comparação entre a Poda Direta e a Poda Iterativa para razões similares. É possível observar que a Poda Direta mantém resultados mais altos de IoU para razões similares.



Fonte: Elaborada pelo autor.

Os resultados resumidos mostram uma diferença mínima de desempenho entre os modelos FP32 e FP16, confirmando que o FP16 oferece um aumento significativo de 2,31x no *throughput* e no tempo de computação na GPU, com impacto negligenciável na acurácia. Embora a quantização INT8 introduza uma pequena perda de desempenho, ela é bem controlada, sendo mitigada pelo método de calibração baseado em percentis, que lida efetivamente com valores atípicos de ativação ao garantir um intervalo adequado de clipping com base na distribuição do conjunto de dados. Em troca dessa pequena perda de acurácia, o modelo INT8 oferece o maior desempenho, atingindo um aumento máximo de 3,55x no tempo de computação na GPU. Isso torna o INT8 uma opção atraente para aplicações sensíveis à latência, nas quais a máxima eficiência é essencial.

Além da quantização, foi avaliado a poda para o modelo PPLite através do algoritmo `DepGraph` conforme descrito na seção 2.4.2. Foi empregada duas abordagens, com os resultados apresentados na Figura 17:

Poda Iterativa: Seguindo a metodologia de poda gradual (HAN et al., 2015), essa estratégia consiste em remover uma fração fixa dos pesos da rede ao longo de várias etapas. Após cada etapa de poda, o modelo foi ajustado (fine-tuning) por 20 épocas para permitir a recuperação após a remoção dos parâmetros.

Poda Direta: Nesta abordagem mais direta, o percentual alvo de pesos foi removido de uma só vez. A rede resultante, menor, foi então ajustada (fine-tuning) por 20 épocas para recuperar seu desempenho.

O gráfico mostra que podar diretamente a arquitetura resulta em melhores desempenhos do que a abordagem iterativa de podar sequencialmente. Como observado por Liu *et al.* (LIU et al., 2019), aprender os pesos importantes de um modelo maior por meio da poda iterativa não beneficia o modelo podado no caso estruturado. De fato, pode apresentar desempenho equivalente ou até pior do que usar pesos aleatórios, já que a configuração de poda anterior pode servir como um ponto de partida inadequado para podas posteriores. Os resultados adicionais da poda direta são apresentados na Tabela 17.

Tabela 17 – Desempenho e aceleração de inferência do método de Poda Direta em diferentes tamanhos de modelo.

Canais Preservados	**IoU (%)**	**Inferência (ms)**
100%	66.72	5.84
90%	61.40	5.31 **(1.10x)**
80%	56.03	4.59 **(1.27x)**
70%	43.08	3.93 **(1.49x)**
60%	35.57	3.49 **(1.67x)**
50%	30.32	2.75 **(2.12x)**
40%	18.11	2.46 **(2.37x)**
30%	0.00	1.91 **(3.06x)**

Fonte: Elaborada pelo autor.

6 Conclusões

Este trabalho implementou e comparou modelos de detecção e segmentação de objetos voltados à identificação de buracos em vias, com ênfase na utilização desses modelos em sistemas embarcados de baixo custo. Para isso, foram aplicadas técnicas de otimização, como quantização e poda de rede, além do uso de um conjunto de dados com perspectiva veicular, geograficamente e climaticamente diversificado, contendo anotações para detecção de objetos e anotações automáticas para segmentação obtidas por meio do modelo SAM.

Entre os detectores de objetos avaliados — YOLO, D-FINE e RT-DETR —, o modelo D-FINE apresentou os melhores resultados, principalmente devido ao seu módulo de refinamento, que se mostrou altamente eficaz na detecção de objetos pequenos. O modelo YOLO obteve os menores tempos de inferência; contudo, após a quantização, o modelo D-FINE conseguiu equiparar sua velocidade, apresentando perda mínima de acurácia em FP16 e uma perda mais acentuada em INT8.

A avaliação dos modelos de segmentação foi dividida em duas análises: modelos genéricos e modelos voltados para aplicações em tempo real. Na análise dos modelos genéricos, buscou-se avaliar extratores de características de baixo custo, e os resultados indicaram que as CNNs baseadas na MobileNet ainda apresentam desempenho dominante nessa categoria. Entretanto, arquiteturas mais recentes baseadas em *Vision Transformers* (ViT), como o MobileViT, também demonstraram resultados competitivos. Já entre os modelos voltados para tempo real, o PPLite apresentou o melhor desempenho e o menor tempo de inferência, sendo ainda mais otimizado após a quantização.

Em relação à otimização, a quantização mostrou-se altamente eficaz, com o modelo de segmentação PPLITE em *INT8* alcançando um expressivo *speedup* de 3,54$\times$ no tempo de computação da GPU, apresentando apenas uma perda mínima na acurácia de segmentação. Por outro lado, a poda de rede (*network pruning*) reduz as restrições de memória e melhora o tempo de inferência, porém com um impacto mais acentuado no desempenho em comparação à quantização. Embora a quantização tenha superado a poda de rede em termos de equilíbrio entre velocidade e precisão, a poda ainda se mantém como uma estratégia valiosa para aplicações em que a compactação de memória é o principal objetivo, como em dispositivos com recursos altamente limitados.

6.1 Trabalhos Futuros

Como consequência desse trabalho, é possível apresentar direções para trabalhos futuros e explorar aspectos que não foram abordados:

- **Fusão com dados de sensores:** Desenvolver uma abordagem de detecção que integre informações provenientes de sensores 3D, como LiDAR, com as imagens capturadas por câmeras RGB. Essa fusão multimodal pode aumentar a confiabilidade das predições, combinando a precisão espacial dos sensores 3D com a riqueza visual das câmeras.
- **Adaptação de Domínio em estradas brasileiras:** Embora o conjunto de dados utilizado apresente diversidade geográfica e climática, é fundamental avaliar o desempenho dos modelos em contextos reais de estradas brasileiras. Para isso, será necessária a coleta de dados locais e a aplicação de técnicas de adaptação de domínio, a fim de transferir de forma eficaz o conhecimento adquirido para o novo ambiente.
- **Outras técnicas de otimização de modelos**: O trabalho avaliou Quantização e Poda para melhorar a performance do modelo. Outras técnicas como Destilação de Conhecimento ou *Quantization Aware Training* podem ser aplicadas e avaliadas com as técnicas já propostas.
- **Exploração de Informação Temporal**: Utilizar da informação temporal fornecida pelo conjunto de dados para aumentar a robustez da predição, principalmente reduzindo falsos positivos.

6.2 Lista de Publicações

Nesta seção, lista-se as publicações resultantes a partir do desenvolvimento deste trabalho:

Publicados

1. **Bruno, D.; Martins, W.; Berri, R.; Osório, F.** *Robotic Visual Attention Architecture for ADAS in Critical Embedded Systems for Smart Vehicles.* In: Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2025). SCITEPRESS – Science and Technology Publications, 2025.
2. **Martins, W. D.; Osorio, F. S.; Bruno, D. R.** *Real-Time Deep Learning-Based Pothole Detection on Low-Cost Embedded Devices for ADAS.* In: *Brazilian Conference on Robotics (CROS 2025)*, Belo Horizonte, Brasil, 2025. IEEE.

Aceita para publicação

1. **Martins, W.; Osorio, F.; Bruno, D.** *Efficient Pothole Segmentation Using Quantization and Pruning for ADAS*. In: *Revista De Informática Teórica E Aplicada*, WVC 2025.

Referências

ABU-RADDAHA, A.; EL-SHAIR, Z. A.; RAWASHDEH, S. Leveraging perspective transformation for enhanced pothole detection in autonomous vehicles. *Journal of Imaging*, MDPI AG, v. 10, n. 9, p. 227, set. 2024. ISSN 2313-433X. Disponível em: <http://dx.doi.org/10.3390/jimaging10090227>.

ADALOGLOU, N. Understanding the receptive field of deep convolutional networks. *https://theaisummer.com/*, 2020. Disponível em: <https://theaisummer.com/receptive-field/>.

AKAGIC, A.; BUZA, E.; OMANOVIC, S. Pothole detection: An efficient vision based method using rgb color space image segmentation. In: *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. [S.l.: s.n.], 2017. p. 1104–1109.

ANWAR, S.; HWANG, K.; SUNG, W. Structured pruning of deep convolutional neural networks. *J. Emerg. Technol. Comput. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 13, n. 3, fev. 2017. ISSN 1550-4832. Disponível em: <https://doi.org/10.1145/3005348>.

ASAD, M. H. et al. Pothole detection using deep learning: A real-time and ai-on-the-edge perspective. *Advances in Civil Engineering*, v. 2022, n. 1, p. 9221211, 2022. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/9221211>.

ASAD, M. H. et al. Pothole detection using deep learning: A real-time and ai-on-the-edge perspective. *Advances in Civil Engineering*, Wiley, v. 2022, n. 1, jan. 2022. ISSN 1687-8094. Disponível em: <http://dx.doi.org/10.1155/2022/9221211>.

AZAD, R. et al. Transdeeplab: Convolution-free transformer-based deeplab v3+ for medical image segmentation. In: _____. *Predictive Intelligence in Medicine*. Springer Nature Switzerland, 2022. p. 91–102. ISBN 9783031169199. Disponível em: <http://dx.doi.org/10.1007/978-3-031-16919-9$_9$¿.

BRUNO, D. R. *Análise e fusão de imagens 2D e 3D com vistas para detecção e classificação de sinais de trânsito verticais em prol da segurança viária com veículos robóticos inteligentes*. Tese (Tese de Doutorado) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2020. Acesso em: 2024-10-01. Disponível em: <https://doi.org/10.11606/T.55.2020.tde-23072020-180612>.

BRUNO, D. R. et al. Carina project: Visual perception systems applied for autonomous vehicles and advanced driver assistance systems (adas). *IEEE Access*, v. 11, p. 69720–69749, 2023.

BUSLAEV, A. et al. Albumentations: Fast and flexible image augmentations. *Information*, v. 11, n. 2, 2020. ISSN 2078-2489. Disponível em: <https://www.mdpi.com/2078-2489/11/2/125>.

CARION, N. et al. End-to-end object detection with transformers. In: VEDALDI, A. et al. (Ed.). *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, 2020. p. 213–229. ISBN 978-3-030-58452-8.

CHEN, G. et al. A survey of the four pillars for small object detection: Multiscale representation, contextual information, super-resolution, and region proposal. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, v. 52, n. 2, p. 936–953, 2022.

CHEN, J. et al. *Foreground-Background Imbalance Problem in Deep Object Detectors: A Review*. 2020. Disponível em: <https://arxiv.org/abs/2006.09238>.

CHEN, L.-C. et al. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs*. 2017. Disponível em: <https://arxiv.org/abs/1606.00915>.

CUN, Y. L.; DENKER, J. S.; SOLLA, S. A. Optimal brain damage. In: *Proceedings of the 3rd International Conference on Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 1989. (NIPS'89), p. 598–605.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. [S.l.: s.n.], 2005. v. 1, p. 886–893 vol. 1.

DHIMAN, A.; KLETTE, R. Pothole detection using computer vision and learning. *IEEE Transactions on Intelligent Transportation Systems*, v. 21, n. 8, p. 3536–3550, 2020.

DOSOVITSKIY, A. et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. Disponível em: <https://arxiv.org/abs/2010.11929>.

EGAJI, O. A. et al. Real-time machine learning-based approach for pothole detection. *Expert Systems with Applications*, v. 184, p. 115562, 2021. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417421009684>.

FAN, M. et al. Rethinking bisenet for real-time semantic segmentation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2021. p. 9716–9725.

FAN, R. et al. We learn better road pothole detection: From attention aggregation to adversarial domain adaptation. In: _____. *Computer Vision – ECCV 2020 Workshops*. Springer International Publishing, 2020. p. 285–300. ISBN 9783030668235. Disponível em: <http://dx.doi.org/10.1007/978-3-030-66823-5$_1$7¿.

FAN, R. et al. Graph attention layer evolves semantic segmentation for road pothole detection: A benchmark and algorithms. *IEEE Transactions on Image Processing*, v. 30, p. 8144–8154, 2021.

FANG, G. et al. Depgraph: Towards any structural pruning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2023. p. 16091–16101.

FELZENSZWALB, P.; MCALLESTER, D.; RAMANAN, D. A discriminatively trained, multiscale, deformable part model. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2008. p. 1–8.

FENG, Z. et al. Mafnet: Segmentation of road potholes with multimodal attention fusion network for autonomous vehicles. *IEEE Transactions on Instrumentation and Measurement*, v. 71, p. 1–12, 2022.

FERNáNDEZ, C. H.; RODRíGUEZ, J. C. Detection of aircraft traces in night-time all-sky camera images using deep learning. a new way to improve aerial safety in satellite laser ranging operations. Unpublished, 2023. Disponível em: <https://rgdoi.net/10.13140/RG.2.2.16114.93128>.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. ⟨http://www.deeplearningbook.org⟩.

GUPTA, S. et al. Detection and localization of potholes in thermal images using deep neural networks. *Multimedia Tools and Applications*, Springer Science and Business Media LLC, v. 79, n. 35–36, p. 26265–26284, jul. 2020. ISSN 1573-7721. Disponível em: <http://dx.doi.org/10.1007/s11042-020-09293-8>.

HAN, S. et al. Learning both weights and connections for efficient neural networks. In: *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1*. Cambridge, MA, USA: MIT Press, 2015. (NIPS'15), p. 1135–1143.

HOWARD, A. G. et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. 2017. Disponível em: <https://arxiv.org/abs/1704.04861>.

IAKUBOVSKII, P. *Segmentation Models Pytorch*. [S.l.]: GitHub, 2019. ⟨https://github.com/qubvel/segmentation_models.pytorch⟩.

JAKUBEC, M. et al. Comparison of cnn-based models for pothole detection in real-world adverse conditions: Overview and evaluation. *Applied Sciences*, v. 13, n. 9, 2023. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/13/9/5810>.

JOHNSON, J. M.; KHOSHGOFTAAR, T. M. Survey on deep learning with class imbalance. *Journal of Big Data*, v. 6, n. 1, p. 27, Mar 2019. ISSN 2196-1115. Disponível em: <https://doi.org/10.1186/s40537-019-0192-5>.

KHANAM, R.; HUSSAIN, M. *YOLOv11: An Overview of the Key Architectural Enhancements*. 2024. Disponível em: <https://arxiv.org/abs/2410.17725>.

KRäHENBüHL, P.; KOLTUN, V. Efficient inference in fully connected crfs with gaussian edge potentials. In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2011. (NIPS'11), p. 109–117. ISBN 9781618395993.

KUHN, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, v. 2, n. 1-2, p. 83–97, 1955. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109>.

KUZMIN, A. et al. Pruning vs quantization: Which is better? *Neural Information Processing Systems*, 2023.

LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, v. 1, n. 4, p. 541–551, 1989.

LIU, Z. et al. Rethinking the value of network pruning. In: *International Conference on Learning Representations*. [s.n.], 2019. Disponível em: <https://openreview.net/forum?id=rJlnB3C5Ym>.

LONG, J.; SHELHAMER, E.; DARRELL, T. *Fully Convolutional Networks for Semantic Segmentation*. 2015. Disponível em: <https://arxiv.org/abs/1411.4038>.

LOSHCHILOV, I.; HUTTER, F. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2017. Disponível em: <https://arxiv.org/abs/1608.03983>.

LYDON, D. et al. A review of vision-based methods for pothole detection and road profile analysis. In: *Civil Engineering Research in Ireland 2020*. [S.l.: s.n.], 2020.

MA, N. et al. Computer vision for road imaging and pothole detection: a state-of-the-art review of systems and algorithms. *Transportation Safety and Environment*, v. 4, n. 4, p. tdac026, 11 2022. ISSN 2631-4428. Disponível em: <https://doi.org/10.1093/tse/tdac026>.

MAEDA, H. et al. Road damage detection and classification using deep neural networks with smartphone images. *Computer-Aided Civil and Infrastructure Engineering*, Wiley, v. 33, n. 12, p. 1127–1141, jun. 2018. ISSN 1467-8667. Disponível em: <http://dx.doi.org/10.1111/mice.12387>.

MASIHULLAH, S. et al. Attention based coupled framework for road and pothole segmentation. In: *2020 25th International Conference on Pattern Recognition (ICPR)*. [S.l.: s.n.], 2021. p. 5812–5819.

MCKINSTRY, J. L. et al. Discovering low-precision networks close to full-precision networks for efficient inference. In: *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*. [S.l.: s.n.], 2019. p. 6–9.

Ministério do Desenvolvimento, Indústria, Comércio e Serviços. *Rota 2030 - Mobilidade e Logística*. 2024. Disponível em: <https://www.gov.br/mdic/pt-br/assuntos/competitividade-industrial/setor-automotivo/rota-2030-mobilidade-e-logistica>.

MUSA, A. et al. A lightweight cnn-based pothole detection model for embedded systems using knowledge distillation. In: _____. *New Trends in Intelligent Software Methodologies, Tools and Techniques*. IOS Press. Disponível em: <http://dx.doi.org/10.3233/FAIA220281>.

NIENABER, S.; BOOYSEN, M.; KROON, R. Detecting potholes using simple image processing techniques and real-world footage. In: *Proc. 34th Annu. Southern African Transport Conf.* Pretoria, South Africa: [s.n.], 2015. p. 1–10.

PADILLA, R.; NETTO, S. L.; SILVA, E. A. B. da. A survey on performance metrics for object-detection algorithms. In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. [S.l.: s.n.], 2020. p. 237–242.

PAUL, A. et al. Advanced driver assistance systems. In: *SAE Technical Paper*. [S.l.: s.n.], 2016.

PENG, J. et al. *PP-LiteSeg: A Superior Real-Time Semantic Segmentation Model*. 2022. Disponível em: <https://arxiv.org/abs/2204.02681>.

PENG, Y. et al. *D-FINE: Redefine Regression Task in DETRs as Fine-grained Distribution Refinement*. 2024.

PRINCE, S. J. *Understanding Deep Learning*. The MIT Press, 2023. Disponível em: <http://udlbook.com>.

RAVI, N. et al. Sam 2: Segment anything in images and videos. *arXiv preprint*, 2024.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016.

RONNEBERGER, O.; FISCHER, P.; BROX, T. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. Disponível em: <https://arxiv.org/abs/1505.04597>.

ROTH, W. et al. *Resource-Efficient Neural Networks for Embedded Systems*. [S.l.], 2019.

SAFYARI, Y.; MAHDIANPARI, M.; SHIRI, H. A review of vision-based pothole detection methods using computer vision and machine learning. *Sensors*, v. 24, n. 17, 2024. ISSN 1424-8220. Disponível em: <https://www.mdpi.com/1424-8220/24/17/5652>.

SANDRU, A.; GEORGESCU, M.-I.; IONESCU, R. T. *Feature-level augmentation to improve robustness of deep neural networks to affine transformations*. 2022. Disponível em: <https://arxiv.org/abs/2202.05152>.

SHAFIQUE, M. A.; MUNIR, A.; KONG, J. Deep learning performance characterization on gpus for various quantization frameworks. *AI*, MDPI AG, v. 4, n. 4, p. 926–948, out. 2023. ISSN 2673-2688. Disponível em: <http://dx.doi.org/10.3390/ai4040047>.

SHEHZADI, T. et al. *Object Detection with Transformers: A Review*. 2023. Disponível em: <https://arxiv.org/abs/2306.04670>.

SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 56, p. 1929–1958, 2014. Disponível em: <http://jmlr.org/papers/v15/srivastava14a.html>.

TIAN, Z. et al. Fcos: A simple and strong anchor-free object detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 44, n. 4, p. 1922–1933, 2022.

TRIPATHI, R.; INDU, S.; KUMAR, R. Ercu-net: segmentation of road potholes using enhanced residual convolutional block based on u-net for adas. *Signal, Image and Video Processing*, Springer Science and Business Media LLC, v. 18, n. S1, p. 385–394, abr. 2024. ISSN 1863-1711. Disponível em: <http://dx.doi.org/10.1007/s11760-024-03161-w>.

VASWANI, A. et al. Attention is all you need. In: GUYON, I. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017. v. 30. Disponível em: <https://proceedings.neurips.cc/paper$_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf$¿.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. [S.l.: s.n.], 2001. v. 1, p. I–I.

WANG, A. et al. *YOLOv10: Real-Time End-to-End Object Detection*. 2024. Disponível em: <https://arxiv.org/abs/2405.14458>.

WANG, Y. et al. A comprehensive review of modern object segmentation approaches. *Foundations and Trends® in Computer Graphics and Vision*, Now Publishers, v. 13, n. 2–3, p. 111–283, 2022. ISSN 1572-2759. Disponível em: <http://dx.doi.org/10.1561/0600000097>.

WANG, Z. et al. Revisiting evaluation metrics for semantic segmentation: optimization and evaluation of fine-grained intersection over union. In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2023. (NIPS '23).

WIGHTMAN, R. *PyTorch Image Models*. [S.l.]: GitHub, 2019. ⟨https://github.com/rwightman/pytorch-image-models⟩.

WU, H. et al. *Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation*. 2020. Disponível em: <https://arxiv.org/abs/2004.09602>.

XIE, E. et al. Segformer: Simple and efficient design for semantic segmentation with transformers. In: *Neural Information Processing Systems (NeurIPS)*. [S.l.: s.n.], 2021.

YANG, J. et al. Semantic segmentation for autonomous driving. In: _____. *Autonomous Driving Perception: Fundamentals and Applications*. Singapore: Springer Nature Singapore, 2023. p. 101–137. ISBN 978-981-99-4287-9. Disponível em: <https://doi.org/10.1007/978-981-99-4287-9$_4$¿.

YOUNESI, A. et al. A comprehensive survey of convolutions in deep learning: Applications, challenges, and future trends. *IEEE Access*, v. 12, p. 41180–41218, 2024.

YU, C. et al. Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation. *International Journal of Computer Vision*, Springer, p. 1–18, 2021.

YU, C. et al. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. [S.l.: s.n.], 2018.

ZAIDI, S. S. A. et al. *A Survey of Modern Deep Learning based Object Detection Models*. 2021. Disponível em: <htt@miscminaee2020imagesegmentationusingdeep, title=Image Segmentation Using Deep Learning: A Survey, author=Shervin Minaee and Yuri Boykov and Fatih Porikli and Antonio Plaza and Nasser Kehtarnavaz and Demetri Terzopoulos, year=2020, eprint=2001.05566, archivePrefix=arXiv, primaryClass=cs.CV, url=https://arxiv.org/abs/2001.05566, ps://arxiv.org/abs/2104.11892>.

ZHAO, H. et al. *Pyramid Scene Parsing Network*. 2017. Disponível em: <https://arxiv.org/abs/1612.01105>.

ZHAO, J.; ZHANG, Z.; QI, F. An intelligent image segmentation annotation method based on segment anything model. In: CRUZ, C.; ZHANG, Y.; GAO, W. (Ed.). *Intelligent Computers, Algorithms, and Applications*. Singapore: Springer Nature Singapore, 2024. p. 237–243. ISBN 978-981-97-0065-3.

ZHAO, Y. et al. Detrs beat yolos on real-time object detection. In: *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2024. p. 16965–16974.

ZHONG, Z. et al. *Random Erasing Data Augmentation*. 2017. Disponível em: <https://arxiv.org/abs/1708.04896>.

ZHOU, Z. et al. Unet++: A nested u-net architecture for medical image segmentation. In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. [S.l.]: Springer, 2018. p. 3–11.

ZOU, Z. et al. Object detection in 20 years: A survey. *Proceedings of the IEEE*, v. 111, n. 3, p. 257–276, 2023.