INSTITUTO FEDERAL DE SANTA CATARINA
CURSO DE ENGENHARIA ELÉTRICA

LUÍS DAVI KENIG PAGANELLA

APRENDIZADO PROFUNDO E VISÃO COMPUTACIONAL APLICADOS EM SISTEMAS DE CONTAGEM VOLUMÉTRICA CLASSIFICADA E DIRECIONAL DE VEÍCULOS

TRABALHO DE CONCLUSÃO DE CURSO

ITAJAÍ
2025

LUÍS DAVI KENIG PAGANELLA

APRENDIZADO PROFUNDO E VISÃO COMPUTACIONAL APLICADOS EM SISTEMAS DE CONTAGEM VOLUMÉTRICA CLASSIFICADA E DIRECIONAL DE VEÍCULOS

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Elétrica do Instituto Federal de Santa Catarina - IFSC Campus Itajaí, como requisito parcial para a obtenção do título de Bacharel em Engenharia Elétrica.

Orientador: Prof. Me. Ênio dos Santos Silva
Instituto Federal de Santa Catarina

ITAJAÍ
2025

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca do IFSC.

Paganella , Luís Davi Kenig
APRENDIZADO PROFUNDO E VISÃO COMPUTACIONAL APLICADOS EM SISTEMAS DE CONTAGEM VOLUMÉTRICA CLASSIFICADA E DIRECIONAL DE VEÍCULOS / Luís Davi Kenig Paganella ; orientador, Ênio dos Santos Silva, 2025.
69 p.

Trabalho de Conclusão de Curso (graduação) - Instituto Federal de Santa Catarina, Campus Itajaí, Graduação em Engenharia eletrica, Itajaí, 2025.

Inclui referências.

1. Engenharia eletrica. 2. Aprendizado Profundo. 3. Visão Computacional. 4. Contagem Volumétrica Classificada. I. Silva, Ênio dos Santos. II. Instituto Federal de Santa Catarina. Graduação em Engenharia eletrica. III. Título.

APRENDIZADO PROFUNDO E VISÃO COMPUTACIONAL APLICADOS EM SISTEMAS DE CONTAGEM VOLUMÉTRICA CLASSIFICADA E DIRECIONAL DE VEÍCULOS

Luís Davi Kenig Paganella

Este trabalho foi julgado adequado para obtenção do Título de Engenheiro Eletricista e aprovado na sua forma final pela banca examinadora do curso de engenharia elétrica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Itajaí, 09 de dezembro de 2025.

BANCA EXAMINADORA:

gov.br Documento assinado digitalmente
ENIO DOS SANTOS SILVA
Data: 09/12/2025 17:06:56-0300
Verifique em https://validar.iti.gov.br

Me. Ênio dos Santos Silva

Instituto Federal de Santa Catarina – IFSC

gov.br Documento assinado digitalmente
GUILHERME RANZOLIN PIAZZETTA
Data: 09/12/2025 17:26:26-0300
Verifique em https://validar.iti.gov.br

Me. Guilherme Ranzolin Piazzetta

Instituto Federal de Santa Catarina – IFSC

gov.br Documento assinado digitalmente
SERGIO AUGUSTO BITENCOURT PETROVCIC
Data: 09/12/2025 17:40:20-0300
Verifique em https://validar.iti.gov.br

Dr. Sérgio Augusto Bitencourt Petrovcic

Instituto Federal de Santa Catarina - IFSC

Dedico este trabalho aos pesquisadores que desejam ingressar na área de visão computacional. Espero que as referências apresentadas e as metodologias propostas sirvam como um guia introdutório de como utilizar a tecnologia para solucionar problemas reais e no auxílio à sociedade.

AGRADECIMENTOS

A conclusão deste trabalho simboliza o final de um ciclo importante na minha vida, onde muitas pessoas atuaram como pilares fundamentais para consolidar minha carreira acadêmica e profissional. Portanto, primeiramente, agradeço a todos os servidores do Instituto Federal de Santa Catarina - Campus Itajaí, em especial os professores que foram os primeiros a abrir as portas que possibilitaram o meu amadurecimento acadêmico. Além disso, foi por meio das aulas que conheci meus colegas de faculdade, que durante esse tempo se tornaram amizades para a vida, como Matheus, Sérgio e Vinícius.

Agradeço aos meus pais, Gelson e Eliane, outro pilar essencial neste trabalho, pois foram o meu maior suporte e motivação durante toda minha jornada de graduação.

Agradeço à minha namorada, Pamela, que esteve ao meu lado e incentivou os meus sonhos desde o início. Seu apoio foi fundamental para persistir diante de qualquer desafio.

Tenho imensa gratidão pelo meu orientador, Professor Ênio dos Santos Silva, que neste caso foi mais que um pilar, pois também foi a ponte que me permitiu passar pelos vales de dúvidas e dificuldades na elaboração deste trabalho. Graças à sua orientação, a minha formação acadêmica foi enriquecida e abriram-se as primeiras portas de uma carreira profissional na área.

E por fim, agradeço a empresa Consultran Engenharia, nas pessoas do Emerson, Guilherme e Rodolfo, que depositaram sua confiança na minha capacidade de conduzir este projeto, além de oferecer todo o suporte técnico necessário para o êxito dos objetivos deste trabalho.

Sem dados, você é apenas mais uma pessoa com uma opinião (W. Edwards Deming).

RESUMO

A mobilidade urbana no Brasil enfrenta desafios significativos relacionados ao congestionamento viário, especialmente em cidades com mais de 250 mil habitantes, onde 36% da população gasta mais de uma hora por dia no trânsito. No âmbito da análise de tráfego, a contagem volumétrica classificada (CVC) desempenha um papel fundamental, quantificando o volume de veículos por classe e por direção de movimento. Atualmente, esse processo tem sido realizado manualmente por operadores humanos, apresentando limitações de precisão e escalabilidade. Com o objetivo de mitigar essas limitações, o Instituto Federal de Santa Catarina, Câmpus Itajaí (IFSC-ITJ), estabeleceu parceria com a empresa Consultran Engenharia para desenvolver uma solução automatizada capaz de substituir a contagem manual em vídeos de 24 horas. Nesse contexto, este trabalho investiga técnicas de visão computacional e aprendizado profundo, estruturando a metodologia em três módulos principais: detecção, rastreamento e contagem. Na etapa de detecção, utiliza-se uma rede neural convolucional customizada (YOLOv8m), treinada com 2 744 imagens extraídas de filmagens urbanas fornecidas pela Consultran Engenharia. O rastreamento é realizado por meio do algoritmo *ByteTrack*, selecionado por seu desempenho amplamente reconhecido na associação de identidades ao longo do tempo. No módulo de contagem, este trabalho propõe dois métodos complementares: a contagem por região, voltada a fluxos unidirecionais, e a contagem por movimento, capaz de identificar a origem e o destino de veículos em interseções. Os sistemas são avaliados em dois cenários urbanos distintos ao longo de um período contínuo de 24 horas, tomando como referência contagens manuais. Resultados de simulações numéricas serão apresentados com foco em diferentes métricas de desempenho, confirmando a eficácia das estratégias empregadas. De modo geral, espera-se demonstrar a viabilidade técnica da solução proposta como alternativa para automatizar parcial ou totalmente o processo de CVC, contribuindo para a modernização dos estudos de tráfego e para a implementação de soluções mais eficientes na mobilidade urbana.

Palavras-chave: Aprendizado profundo, contagem classificada de veículos, YOLOv8, rastreamento de veículos.

ABSTRACT

Urban mobility in Brazil faces significant challenges related to traffic congestion, especially in cities with more than 250 thousand inhabitants, where 36% of the population spends more than one hour per day in traffic. In traffic analysis, classified volumetric counting (CVC) plays a fundamental role, quantifying the volume of vehicles by class and direction of movement. Currently, this process has been performed manually by human operators, which presents limitations in precision and scalability. Aiming to mitigate these limitations, the Federal Institute of Santa Catarina, Itajaí Campus (IFSC-ITJ), established a partnership with the company Consultran Engenharia to develop an automated solution capable of replacing manual counting in 24-hour videos. This work investigates computer vision and deep learning techniques, structuring the methodology in three main modules: detection, tracking and counting. For the detection stage, a custom convolutional neural network (YOLOv8m) was trained with 2 744 images extracted from urban footage provided by Consultran Engenharia. The tracking is performed by applying the *ByteTrack* algorithm, selected for its widely recognized performance in associating identities over time. In the counting module, this work proposes two complementary methods: region-based counting, aimed at unidirectional flows, and movement-based counting, capable of identifying the origin and destination of vehicles at intersections. The systems are evaluated in two distinct urban scenarios over a continuous 24-hour period, taking manual counts as reference. Results will be presented with focus on different performance metrics, confirming the effectiveness of the strategies employed. Overall, it is expected to demonstrate the technical feasibility of the proposed solution as an alternative to partially or fully automate the CVC process, contributing to the modernization of traffic studies and to the implementation of more efficient solutions in urban mobility.

Keywords: Deep learning, classified vehicle counting, YOLOv8, vehicle tracking, computer vision.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP	*Average Precision*
CVC	Contagem Volumétrica Classificada
CNN	*Convolutional Neural Network*
CPU	*Central Processing Unit*
CUDA	*Compute Unified Device Architecture*
DNIT	Departamento Nacional de Infraestrutura e Transportes
FHWA	*Federal Highway Administration*
FP	*False Positive*
FPN	*Feature Pyramid Network*
FPS	*Frames Per Second*
FN	*False Negative*
GPU	*Graphics Processing Unit*
IA	Inteligência Artificial
ID	Identificação
IDSW	*Identities Switch*
IFSC-ITJ	Instituto Federal de Santa Catarina - Câmpus Itajaí
MLP	*Multilayer Perceptron*
MOT	*Multiple Object Tracking*
MOTA	*Multiple Object Tracking Accuracy*
MS COCO	*Microsoft Common Objects in Context*
PASCAL VOC	*Pattern Analysis, Statistical Modelling, and Computational Learning Visual Object Classes*
RCNN	*Region Based Convolutional Neural Network*
TP	*True Positive*
YOLO	*You Only Look Once*

LISTA DE SÍMBOLOS

CIoU	*Complete Intersection over Union*
DFL	*Distribution Focal Loss*
IoU	*Intersection over Union*
mAP	*mean Average Precision*
mAP50-95	*mean Average Precision* no limiar 50
mAP50-95	*mean Average Precision* nos limiares de 50 a 95

SUMÁRIO

1 INTRODUÇÃO

A mobilidade urbana no Brasil enfrenta desafios significativos, especialmente nas cidades com mais de 250 mil habitantes, onde 36% da população passa mais de uma hora por dia no trânsito (Abreu, 2023). Em grandes metrópoles, como a cidade de São Paulo, o congestionamento viário tem se configurado como um problema significativo há décadas. Segundo Scaringella (2001), em um intervalo de apenas cinco anos, a média de quilômetros congestionados passou de aproximadamente 40 km para cerca de 120 km, evidenciando uma intensificação relevante da sobrecarga no sistema de transporte.

Após a pandemia de COVID-19, o cenário brasileiro tornou-se ainda mais crítico: cerca de 29% da população relatou um aumento no tempo diário gasto no trânsito (Abreu, 2023), o que reforça a urgência na busca por soluções inovadoras e eficazes para a melhoria da mobilidade urbana. Nesse contexto, o estudo de tráfego desempenha um papel fundamental na busca por soluções para a mobilidade urbana, pois permite a coleta e análise detalhada dos cinco elementos essenciais do tráfego: motorista, pedestre, veículo, via e meio ambiente, além de suas interações (Departamento Nacional de Infraestrutura e Transportes, DNIT, 2006). Entre os métodos mais utilizados para analisar congestionamentos, destaca-se a contagem volumétrica classificada (CVC), que quantifica por diferentes classes o volume de veículos e suas respectivas direções de conversão em vias ou cruzamentos.

Para a execução da análise de congestionamento mencionada, uma equipe de operadores humanos pode ser encarregada de realizar a contagem por meio de observação direta da via, ou seja, presencialmente no local, ou por meio de filmagens, o que se mostra uma alternativa mais prática, reduzindo as chances de erro (DNIT, 2006). No entanto, por se tratar de uma tarefa manual e repetitiva, o operador humano está suscetível a falhas, especialmente em situações de grande fluxo de veículos.

Essa limitação é amplamente reconhecida tanto no âmbito institucional quanto na prática profissional. Nesse contexto, o DNIT, em seu manual técnico, recomenda que engenheiros de tráfego considerem a precisão alcançável e utilizem a experiência consolidada pelos órgãos rodoviários para ajustar suas pesquisas. Consequentemente, o elevado custo de mobilizar equipes, associado ao potencial de falhas decorrentes da intervenção humana, compromete a ampla aplicação da CVC em estudos de tráfego, dificultando a implementação de soluções eficazes para os desafios da mobilidade urbana. Contudo, o manual supracitado foi publicado em 2006 e, portanto, não contempla os avanços tecnológicos ocorridos nas últimas décadas. Em contraste, o Departamento de Transportes dos Estados Unidos (*Federal Highway Administration* - FHWA) já recomenda, em seu manual de monitoramento de tráfego, o uso de ferramentas

computacionais para a contagem automática de veículos em vídeo (FHWA, 2022). Esse tipo de solução insere-se no campo da visão computacional, área dedicada ao desenvolvimento de métodos capazes de reproduzir a habilidade humana de interpretar informações visuais presentes em imagens e vídeos (Szeliski, 2022).

Nesse contexto, o Instituto Federal de Santa Catarina, Câmpus Itajaí, (IFSC-ITJ), em parceria com a empresa Consultran Engenharia, sediada no Vale do Itajaí, estabeleceu estratégias de colaboração técnico-científica, por meio da realização de Projetos Integradores e de Programa de Estágio, com o objetivo de desenvolver uma solução tecnológica voltada à automação da contagem volumétrica classificada e direcional de veículos. Até o ano de 2022, período que marca o início da cooperação entre o IFSC-ITJ e a Consultran Engenharia, esse processo era executado de forma manual, por operadores contratados, que analisavam vídeos com duração de até 24 horas, obtidos por câmeras instaladas em posição isométrica sobre a via.

Diante dessas limitações e da crescente demanda por métodos mais precisos, eficientes e escaláveis, este trabalho propõe o desenvolvimento de um sistema automatizado de contagem de veículos com base em técnicas modernas de visão computacional. Espera-se, assim, contribuir para a modernização dos estudos de tráfego e para a implementação de soluções mais eficazes na área de mobilidade urbana.

1.1 Objetivos

1.1.1 Objetivo Geral

Aplicar técnicas de visão computacional e aprendizado profundo para desenvolver um sistema de contagem volumétrica classificada e direcional de veículos em filmagens de tráfego e avaliar a sua viabilidade a partir dos resultados obtidos em relação à contagem manual.

1.1.2 Objetivos Específicos

- Investigar modelos de redes neurais para detecção e classificação de veículos em motocicleta, carro, ônibus e caminhão;
- Investigar algoritmos de rastreamento de múltiplos objetos para aplicação em análise de tráfego;
- Avaliar o desempenho de modelos de rede neural pré-treinados para identificar possíveis limitações de desempenho nas filmagens de tráfego disponibilizadas pela Consultran Engenharia;
- Coletar e preparar conjunto de dados específico para treinamento, baseado nas limitações identificadas no modelo analisado;
- Treinar e validar modelo customizado de rede neural para detecção e classificação das categorias veiculares de interesse;

- Aplicar e avaliar o desempenho do algoritmo rastreador a partir das inferências do modelo de rede neural nas filmagens de tráfego disponibilizadas pela Consultran Engenharia.
- Desenvolver algoritmo para transformação dos dados de detecção e rastreamento em contagem classificatória, segmentada por direção de movimento e fracionada em intervalos de 15 minutos;
- Aplicar o sistema completo em filmagens de tráfego de 24 horas disponibilizadas pela Consultran Engenharia e avaliar os resultados em relação à contagem manual de referência.

1.2 Organização do Trabalho

Este documento está estruturado em cinco capítulos, organizados conforme descrito a seguir. O Capítulo 2 apresenta o referencial teórico que fundamenta o desenvolvimento deste trabalho, abordando conceitos como: métodos de contagem volumétrica classificada para estudos de tráfego, fundamentos de inteligência artificial e visão computacional, redes neurais convolucionais e algoritmos de rastreamento de múltiplos objetos. No Capítulo 3, é detalhada a metodologia empregada, descrevendo as estratégias de desenvolvimento dos três módulos principais do sistema proposto: detecção e classificação de veículos, rastreamento de objetos e contagem direcional automatizada. São apresentados desde a configuração do ambiente computacional, preparação e rotulação do conjunto de dados, treinamento customizado da rede neural, até a implementação dos algoritmos de rastreamento e da metodologia de contagem proposta. O Capítulo 4 apresenta e discute os resultados obtidos em cada etapa do sistema, incluindo métricas de desempenho do modelo treinado, análise do comportamento do rastreador e validação da contagem automatizada em cenários reais de 24 horas. Por fim, o Capítulo 5 traz as conclusões e considerações finais sobre a viabilidade e eficácia da solução desenvolvida.

2 REFERENCIAL TEÓRICO

Esta seção apresenta, inicialmente, os conceitos fundamentais da pesquisa de tráfego veicular, área da Engenharia na qual este trabalho pretende aplicar tecnologias do estado da arte para realizar contagens automáticas. Na sequência, são introduzidos os fundamentos teóricos da Inteligência Artificial e da Visão Computacional, cuja compreensão conceitual auxilia o entendimento posterior dos mecanismos de aprendizado de máquina e das tecnologias empregadas na detecção e classificação automática de veículos.

Torna-se igualmente relevante abordar as técnicas de processamento temporal utilizadas pelo estado da arte, particularmente no que se refere à correlação entre objetos detectados em diferentes quadros de uma sequência de imagens. Esta fundamentação teórica possibilita a compreensão adequada dos algoritmos de rastreamento múltiplo de objetos, componente essencial da metodologia desenvolvida.

Por fim, são apresentados trabalhos relacionados ao tema, acompanhados de análise crítica quanto à adequação de seus resultados e metodologias em relação aos objetivos específicos demandados pela empresa Consultran Engenharia, estabelecendo assim o contexto científico no qual este estudo se insere.

É importante destacar que o desenvolvimento prático da solução para a demanda da empresa Consultran Engenharia foi realizado (inicialmente) no período compreendido entre setembro de 2022 e agosto de 2023. Este recorte temporal é fundamental para contextualizar que as tecnologias pesquisadas e implementadas correspondem ao estado da arte disponível até a referida data. Cabe ressaltar que o referencial teórico contempla trabalhos relacionados publicados posteriormente a este período, decisão justificada tanto pela relevância científica dessas contribuições quanto pelo fato de que as tecnologias empregadas neste estudo mantêm sua aplicabilidade e pertinência no cenário atual da área.

2.1 Contagem Volumétrica Classificada

A Contagem Volumétrica Classificada (CVC) é um método de pesquisa de tráfego que quantifica o volume de veículos em vias e os classifica por tipo de veículo (como por exemplo: automóveis, motocicletas, ônibus e caminhões). Em análises de interseções de vias (cruzamento), a pesquisa de CVC registra também as direções de conversão (movimentos) que os veículos realizaram, como seguir em frente, virar à direita ou à esquerda (DNIT, 2006). Os resultados das contagens são organizadas em intervalos regulares de 15 minutos, denominados quartis. Dessa forma, uma contagem realizada ao longo de 24 horas compreende 96 quartis. Um exemplo desse tipo de

planilha pode ser observado no material fornecido pela empresa Consultran Engenharia, ilustrado na Figura 1.

Figura 1 – Modelo de planilha utilizado pela Consultran Engenharia.

Formulário padrão de contagem de VEÍCULOS											
Cruzamento		153-010B	Marginal Rod. BR-101 (Leste - R. Salima Salum) X Ac. Entrada Rod. BR-101 (Sent. Norte - Próx. INPLAC) - Quarta-feira								
Data	Número do Posto	Número do Movimento	Horário de Início	Horário de Fim	Volumes de Veículos						Eventuais obstruções aos fluxos de tráfego pesquisados
					Moto	Automóvel	Ônibus	Caminhão	Bicicleta	Bicicleta contramão	
08/06/2022	153-010B	02	06:00	06:15	18	61	5	1	5	0	
08/06/2022	153-010B	02	06:15	06:30	16	59	12	8	6	1	
08/06/2022	153-010B	02	06:30	06:45	34	96	8	7	11	0	
08/06/2022	153-010B	02	06:45	07:00	47	151	2	6	10	1	
08/06/2022	153-010B	02	07:00	07:15	27	186	8	7	8	3	
08/06/2022	153-010B	02	07:15	07:30	32	287	4	9	4	2	
08/06/2022	153-010B	02	07:30	07:45	55	258	6	5	17	0	
08/06/2022	153-010B	02	07:45	08:00	70	274	6	5	12	1	
08/06/2022	153-010B	02	08:00	08:15	56	212	4	8	10	1	
08/06/2022	153-010B	02	08:15	08:30	51	204	3	4	7	2	
08/06/2022	153-010B	02	08:30	08:45	38	227	3	12	7	0	
08/06/2022	153-010B	02	08:45	09:00	45	255	3	10	3	1	
08/06/2022	153-010B	02	09:00	09:15	37	225	5	8	5	2	
08/06/2022	153-010B	02	09:15	09:30	47	194	1	16	3	2	
08/06/2022	153-010B	02	09:30	09:45	50	233	4	14	1	3	
08/06/2022	153-010B	02	09:45	10:00	48	233	4	10	6	3	
08/06/2022	153-010B	02	10:00	10:15	49	229	2	15	2	5	
08/06/2022	153-010B	02	10:15	10:30	56	215	5	10	6	3	
08/06/2022	153-010B	02	10:30	10:45	48	242	4	18	3	3	
08/06/2022	153-010B	02	10:45	11:00	53	234	3	11	4	2	
08/06/2022	153-010B	02	11:00	11:15	51	242	6	13	5	2	
08/06/2022	153-010B	02	11:15	11:30	44	247	5	12	4	2	
08/06/2022	153-010B	02	11:30	11:45	46	272	7	8	1	5	
08/06/2022	153-010B	02	11:45	12:00	53	255	9	17	6	4	
08/06/2022	153-010B	02	12:00	12:15	64	258	1	11	7	2	
08/06/2022	153-010B	02	12:15	12:30	65	224	5	9	2	2	
08/06/2022	153-010B	02	12:30	12:45	44	255	11	7	5	1	
08/06/2022	153-010B	02	12:45	13:00	59	294	6	15	10	3	
08/06/2022	153-010B	02	13:00	13:15	60	361	4	12	4	0	
08/06/2022	153-010B	02	13:15	13:30	72	357	3	5	9	1	
08/06/2022	153-010B	02	13:30	13:45	67	325	3	12	6	3	
08/06/2022	153-010B	02	13:45	14:00	54	314	3	10	7	4	

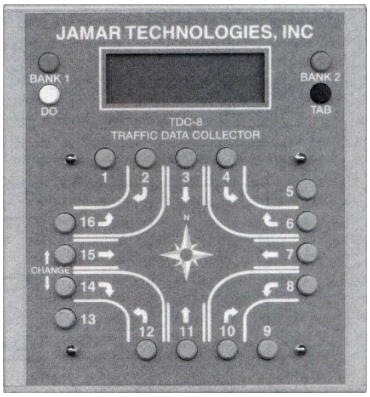
Fonte: Consultran Engenharia LTDA (2024).

Para a realização dessas pesquisas, agentes de trânsito formam equipes com operadores responsáveis por observar presencialmente o tráfego de veículos e registrar as contagens. É comum a utilização de pranchetas acopladas a equipamentos de contagem mecânica, sendo cada dispositivo destinado a uma classe específica de veículo, conforme ilustrado na Figura 2. Além dessa estratégia de contagem manual, também existem equipamentos eletrônicos, como o contador da empresa Jamar Technologies mostrado na Figura 3. Esse equipamento possibilita registrar diretamente os dados correspondentes aos 16 possíveis movimentos realizados em uma interseção.

Figura 2 – Foto de uma prancheta com contadores mecânicos.

Fonte: Neto (2015).

Figura 3 – Foto de um contador eletrônico.



Fonte: DNIT (2006).

Dentre as orientações relacionadas à contagem CVC, o manual técnico do DNIT (2006) também sugere que a contagem manual seja realizada a partir da análise de gravações da via, uma vez que esse procedimento tende a reduzir a ocorrência de erros quando comparado ao trabalho dos operadores atuando *in loco*. Dessa forma, seguindo esta sugestão, a empresa Consultran Engenharia adota a metodologia supramencionada, realizando filmagens das interseções de interesse para posterior execução da contagem manual. A Figura 4 apresenta um croqui da visada da câmera, no qual são representadas linhas que delimitam as regiões de entrada e saída dos veículos, caracterizando os diferentes movimentos observados.

Figura 4 – Croqui de uma interseção.

Fonte: Consultran Engenharia LTDA (2024).

2.1.1 Filmagens da Consultran Engenharia

O desenvolvimento deste trabalho contou com o apoio da empresa Consultran Engenharia, que forneceu um acervo de vídeos obtidos por meio de câmeras posicionadas em vias e interseções urbanas, utilizados em suas pesquisas de tráfego. Os vídeos contemplam 24 horas ininterruptas de filmagem, com as câmeras instaladas a aproximadamente 3 metros de altura, garantindo uma visão ampla da área monitorada.

As gravações foram realizadas com o modelo de câmera *Intelbras iM5 S*, cujas especificações técnicas estão apresentadas no Quadro 1. Já as características dos arquivos de vídeo resultantes estão descritas no Quadro 2.

Quadro 1 – Especificação da câmera iM5 S.

Característica	**Especificação**
Sensor	1/2.8"2 MP CMOS
Pixels relativos	1920(H) x 1080 (V)
Ângulo de visão	125° (diagonal), 106° (horizontal) e 56° (vertical)
Frame rate	até 30 fps
Compressão de vídeo	H.265 ou H.264
Taxa de bits	adaptável

Fonte: Dados extraídos de Intelbras S.A. (2024).

Quadro 2 – Especificação dos vídeos.

Parâmetro	**Valor**
Formato	.mp4
Dimensão	640x480 (4:3)
Frame rate	10 fps
Compressão de vídeo	H.265
Taxa de bits	1100 kbps

Fonte: Dados extraídos de Consultran Engenharia LTDA (2024).

2.2 Visão Computacional e Inteligência Artificial

O ser humano possui, de forma inerente, a capacidade de interpretar projeções bidimensionais do ambiente e, a partir delas, extrair informações complexas, como características, estrutura e individualidade dos objetos presentes no espaço. Replicar essa habilidade é o objetivo da Visão Computacional, área que investiga e desenvolve tecnologias destinadas à reconstrução e representação das propriedades do mundo real em ambientes virtuais (Szeliski, 2022).

Nesse contexto, a Inteligência Artificial (IA) abrange um campo dedicado ao desenvolvimento de máquinas capazes de atuar de maneira eficiente em diferentes cenários (Russell; Norvig, 2020). Para que esses sistemas inteligentes possam interagir

de forma adequada com o ambiente que os rodeia, a Visão Computacional desempenha um papel crucial, atuando como o mecanismo responsável pela entrada de dados visuais, isto é, o sistema que confere “visão” à máquina.

Embora (inicialmente) os pioneiros da IA, na década de 1970, acreditassem que a interpretação visual seria uma tarefa simples, essa se revelou mais complexa que o esperado. Os primeiros esforços concentraram-se em técnicas básicas de detecção de bordas, evoluindo até as modernas técnicas de aprendizado de máquina, que revolucionaram a identificação de objetos em imagens e vídeos. Atualmente, a visão computacional está presente em diversas aplicações industriais, desde equipamentos médicos até sistemas de monitoramento (Szeliski, 2022).

2.2.1 Aprendizado de Máquina

O aprendizado de máquina é um subcampo da IA que se dedica ao desenvolvimento de sistemas capazes de aprimorar seu desempenho a partir da experiência. Em essência, busca-se permitir que computadores observem dados, identifiquem padrões e ajustem seu comportamento de forma autônoma, dispensando a necessidade de programação explícita para cada tarefa específica (Russell; Norvig, 2020). Pode-se considerar três principais tipos de aprendizado de máquina:

- Aprendizado supervisionado. Um método de ensino onde a máquina aprende a partir de rótulos (*labels*) que indicam o significado de cada amostra do conjunto de dados. Nesse processo, a máquina recebe entradas (como imagens ou dados numéricos) juntamente com as saídas preciamente rotuladas (por exemplo, “gato” ou “cachorro” para imagens de animais). A máquina, então, compara as entradas com essas saídas rotuladas para aprender a associar corretamente as entradas futuras com os rótulos apropriados.
- Aprendizado não supervisionado. A máquina aprende a partir de dados que não possuem rótulos. Em vez de dizer à máquina o que cada dado significa, ela tenta encontrar padrões e agrupamentos (*clusters*) por conta própria.
- Aprendizado por reforço. A máquina aprende por meio de tentativa e erro, recebendo recompensas ou punições com base nas suas ações. A máquina experimenta várias ações e aprende quais são mais benéficas ao longo do tempo, buscando maximizar suas recompensas (Russell; Norvig, 2020).

2.2.2 Aprendizado Profundo

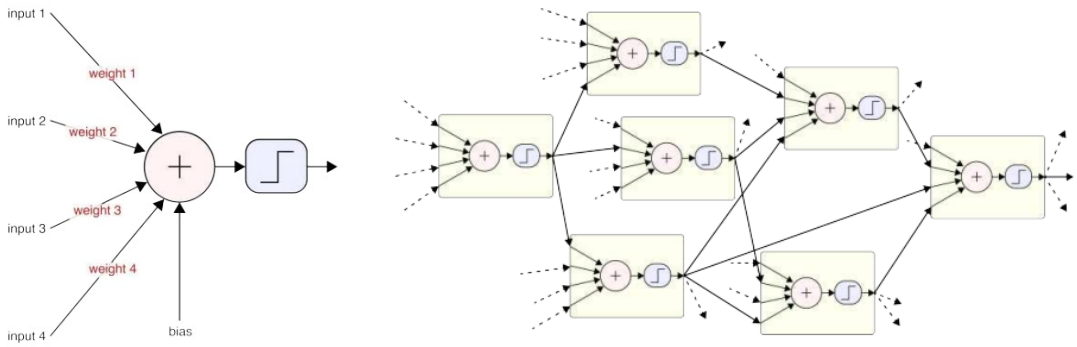
Ainda no campo da IA, o aprendizado profundo (*deep learning*) constitui uma abordagem avançada de aprendizado de máquina. Essa técnica vem ganhando ampla popularidade em tarefas de detecção em imagens devido à sua capacidade de otimizar o processo de treinamento de redes neurais, minimizando os erros associados à função de custo e elevando o desempenho dos modelos (Szeliski, 2022). Nesse contexto, o

termo “profundo” refere-se ao uso de múltiplas camadas em uma rede neural, por meio das quais os dados de entrada são sucessivamente transformados e refinados até a geração da estimativa final na saída (Russell; Norvig, 2020).

2.2.3 Neurônio Artificial e Rede Neural

Um neurônio artificial é uma unidade computacional que realiza uma soma ponderada das entradas, multiplicando cada sinal por um peso (*weight*) correspondente, somando um viés (*bias*) para ajuste final e aplicando uma função de ativação para determinar a saída (Figura 5.a). A partir da década de 1980, pesquisadores começaram a conectar múltiplos neurônios em paralelo e em série, formando redes neurais, permitindo criar modelos mais complexos e capazes de capturar padrões não lineares (Figura 5.b) (Razavi, 2021).

Figura 5 – Em (a) modelo de um perceptron (neurônio artificial). Em (b) ilustração de uma rede neural.



(a) (b)

Fonte: Szeliski (2022).

2.2.4 Redes Neurais Convolucionais

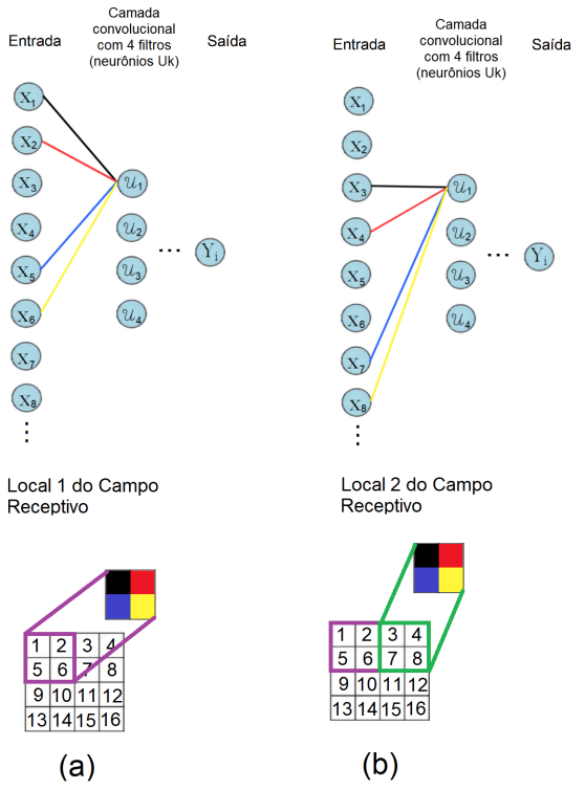
No início da década de 1990, era comum redimensionar uma imagem bidimensional para um único vetor unidimensional, concatenando todas as suas linhas ou colunas. No entanto, segundo Russell e Norvig (2020), uma imagem não pode ser tratada como um simples vetor de *pixels*, pois a relação espacial entre *pixels* adjacentes é fundamental para capturar sua estrutura e os padrões visuais presentes.

Além da importância dessa relação espacial, utilizar imagens bidimensionais como entrada em redes neurais do tipo *multilayer perceptron* (MLP) gera um número extremamente elevado de conexões. Assim, redes neurais totalmente conectadas aplicadas diretamente ao processamento de imagens demandam um poder computacional excessivo, tornando o treinamento inviável para dados de alta dimensionalidade.

Para mitigar esse problema, são introduzidas as redes neurais convolucionais (*convolutional neural networks* - CNNs). Nas CNNs, as camadas (camadas convolucionais) operam sobre pequenas regiões locais da imagem, reduzindo drasticamente o número de parâmetros e preservando a estrutura espacial do sinal visual. Dessa forma, as CNNs conseguem identificar padrões em diferentes áreas da imagem, de modo semelhante ao processo de análise visual realizado pelos seres humanos. Essas regiões são processadas por matrizes de pesos de dimensões reduzidas (por exemplo, 3×3, 5×5, 7×7, entre outras), denominadas *kernels* ou filtros convolucionais.

Durante o processamento da camada convolucional, o *kernel* "desliza" sobre toda a extensão da matriz de entrada (por exemplo imagem), multiplicando seus pesos pelos valores dos *pixels* correspondentes e somando os produtos resultantes a cada posição. Este processo gera um mapa de características (*feature map*), que preserva a informação espacial da imagem original e evidencia onde determinados padrões foram detectados (Goodfellow *et al.*, 2016). Dessa forma, os mapas de características constituem a base para a identificação e classificação de objetos na imagem. A Figura 6 de Albuquerque (2022) ilustra bem como ocorre a convolução, em que uma matriz 4×4 é processada por uma camada convolucional com 4 filtros.

Figura 6 – Ilustração de uma convolução.



Fonte: Albuquerque (2022).

A partir desses conceitos, arquiteturas com diversas camadas convolucionais

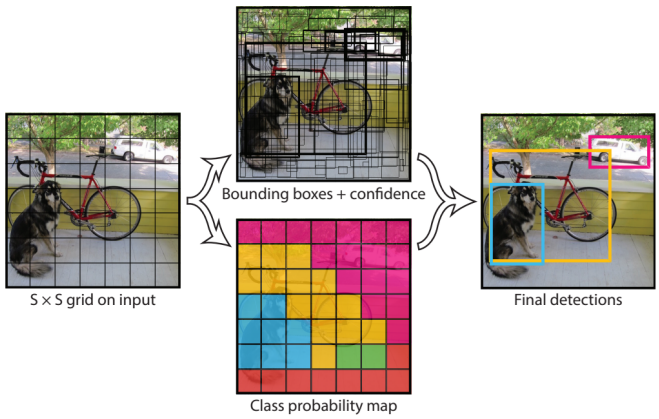
podem ser criadas para detecção e classificação de objetos. Hussain (2023) destaca arquiteturas como redes neurais convolucionais baseadas em regiões (*region based convolutional neural networks* - RCNN), *Fast* RCNN e *Faster* RCNN, que seguem uma abordagem de dois estágios: primeiro propõem regiões candidatas na imagem e depois realizam classificação e localização dessas regiões.

2.3 YOLO (You Only Look Once)

A YOLO (You Only Look Once) foi introduzida por Redmon et al. no artigo *You Only Look Once: Unified, Real-Time Object Detection* (Redmon *et al.*, 2016), publicado em 2016. Essa arquitetura surgiu como uma alternativa mais rápida e eficiente aos detectores de dois estágios, como as RCNNs. Em vez de primeiro propor regiões de interesse e, posteriormente, classificá-las, a YOLO reformula a detecção de objetos como uma única tarefa de regressão. Por isso o nome *You Only Look Once* (Você só vê uma vez), em que prevê simultaneamente as caixas delimitadoras e as probabilidades de classe diretamente da imagem completa.

Redmon *et al.* (2016) explica que, para isso, a YOLO divide a imagem em uma grade de células de tamanho $S \times S$, onde cada célula é responsável por detectar quais objetos estão presentes nos pixels dentro de sua área. Cada célula, então, prevê as caixas delimitadoras que podem englobar um objeto (*bounding boxes*), juntamente com um *score* de confiança que reflete o quanto o modelo acredita que aquela previsão está correta. Essa abordagem diminui consideravelmente a quantidade de *bounding boxes*, chegando a apenas 98 por imagem, contra cerca de 2 000 da RCNN. A Figura 7 ilustra esse processo, mostrando a divisão da imagem e o mapeamento das probabilidades para a detecção dos objetos.

Figura 7 – Ilustração da etapa de divisão em grade para mapeamento das probabilidades e retorno da detecção classificada pela YOLOv1.



Fonte: Redmon *et al.* (2016).

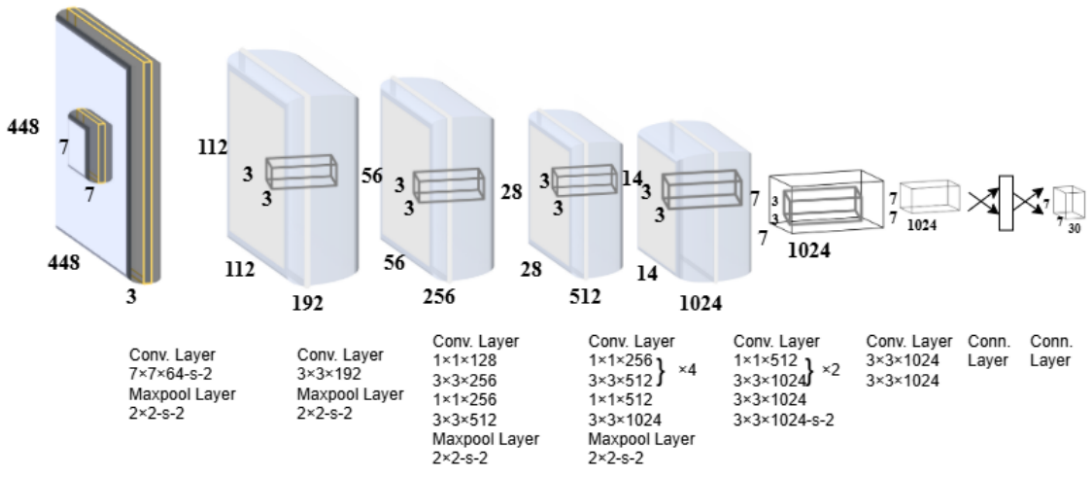
Cada célula de grade na YOLO prevê um tensor com as dimensões $S \times S \times (B \times 5 + C)$, onde:

- $S \times S$ é a divisão da imagem em uma grade, sendo S o número de células por eixo.
- B é o número de caixas preditas por célula de grade.
- 5 é o número de valores para cada caixa predita (coordenadas x, y, largura w, altura h, e a confiança).
- C é o número de classes possíveis.

Na primeira versão apresentada por Redmon *et al.* (2016), foram utilizados os seguintes parâmetros: $B = 2$ caixas delimitadoras por célula, $C = 20$ classes e uma divisão da imagem em $S = 7$ células por eixo, resultando em uma grade de 7×7. Dessa forma, o tensor final gerado pela rede possui dimensões $7 \times 7 \times 30$, onde cada célula da grade contém as previsões das caixas delimitadoras e as probabilidades associadas às classes. Esse tensor tridimensional constitui a saída final da rede e é utilizado no processo de detecção e classificação dos objetos presentes na imagem.

A arquitetura da YOLO foi implementada por (Redmon *et al.*, 2016) como uma rede neural convolucional composta por 24 camadas convolucionais seguidas de 2 camadas totalmente conectadas, totalizando 26 (Figura 8). Essa arquitetura foi implementada utilizando a estrutura de rede denomidada *Darknet*, uma estrutura leve desenvolvida especificamente para redes neurais convolucionais. As primeiras camadas convolucionais têm a função de extrair características relevantes da imagem de entrada, enquanto as camadas finais totalmente conectadas são responsáveis por realizar as previsões das coordenadas das caixas delimitadoras, dos *scores* de confiança e das probabilidades de classe.

Figura 8 – Arquitetura da primeira versão da YOLO.

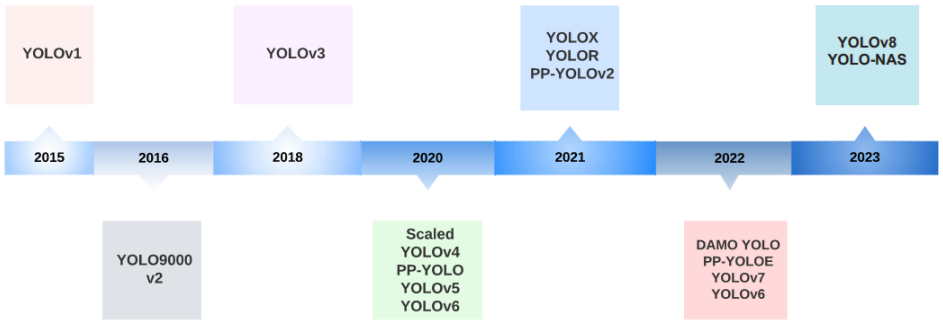


Fonte: Zareen *et al.* (2025).

2.3.1 Evolução da YOLO

A YOLO evoluiu significativamente desde sua primeira publicação, conhecida como YOLOv1 (*You Only Look Once* versão 1). A Figura 9 ilustra os respectivos anos de lançamento de cada versão, culminando na YOLOv8 em 2023. Terven *et al.* (2023) e Hussain (2023) fazem uma análise detalhada de cada versão, destacando a introdução de melhorias como *anchor boxes*, *passthrough layers* e detecção em múltiplas escalas, técnicas que contribuíram para a evolução da acurácia do modelo ao longo do tempo.

Figura 9 – Histórico de lançamento de arquiteturas da família YOLO até 2023.



Fonte: Terven *et al.* (2023).

Conforme aumentou a complexidade, Terven *et al.* (2023) explicam que a anatomia das arquiteturas de detectores de objeto, incluindo versões mais recentes da YOLO, passaram a ser organizadas em três partes principais: *Backbone* (Espinha Dorsal), *Neck* (Pescoço) e *Head* (Cabeça). O *Backbone* é a primeira etapa, onde a imagem de entrada passa por uma CNN, que captura as características em diferentes níveis. O *Neck* conecta os dados até o *Head*, agregando e refinando as características. O *Head* é a responsável por calcular as predições baseadas nas características que chegaram até ela.

A YOLOv8, lançada pela empresa Ultralytics, possui versões escalonadas, variando de modelos mais leves (nano) até versões mais robustas (*extra-large*), sendo elas: YOLOv8n (nano), YOLOv8s (*small*), YOLOv8m (*medium*), YOLOv8l (*large*) e YOLOv8x (*extra-large*). Desenvolvida com o *framework* PyTorch, a arquitetura utiliza a CSPDarknet53 como rede neural em seu *backbone*, uma versão aprimorada da Darknet. No *neck*, são empregadas técnicas que favorecem a combinação de características em diferentes escalas, como o Módulo C2f, além da Rede em Pirâmide de Características (*Feature Pyramid Network* - FPN), que organiza essas características em uma hierarquia mais informativa (Terven *et al.*, 2023).

2.4 Treinamento

A YOLO é um modelo de inteligência artificial treinado por meio de aprendizado supervisionado, no qual cada imagem de entrada possui objetos previamente anotados. O treinamento do modelo, portanto, inicia-se com a etapa de preparação do conjunto de dados (*dataset*). Uma vez definido o *dataset*, torna-se possível realizar o processo de ajuste da rede. Contudo, antes de compreender em detalhes como esse treinamento ocorre, é necessário apresentar as métricas de avaliação utilizadas, pois são elas que permitem mensurar a qualidade das predições e orientar a busca por um melhor desempenho do modelo.

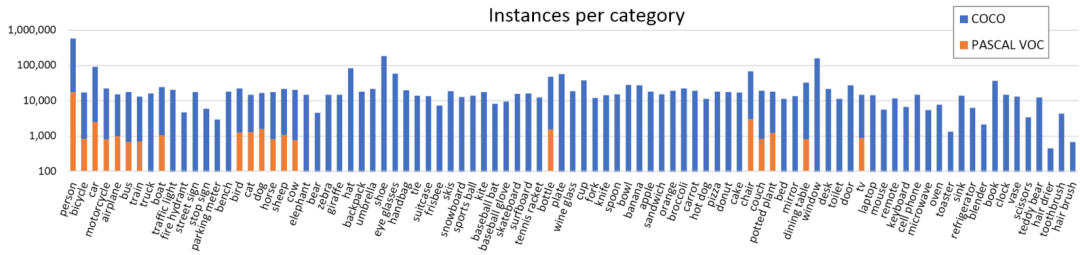
2.4.1 Conjunto de Dados

O repositório oficial da Ultralytics (2023) fornece um tutorial que explica como montar um *dataset* para treinamento supervisionado na YOLO. Basicamente, o *dataset* deve ser dividido em dois subconjuntos principais: *dataset* de treino e *dataset* de validação. Esses subconjuntos possuem imagens e *labels* que indicam as coordenadas e classe de cada objeto na imagem.

As imagens escolhidas para o *dataset* devem refletir as condições reais em que o modelo vai ser aplicado, contemplando diferentes horários do dia, níveis de iluminação e ângulo. Já a rotulação do *dataset* deve ser feita com bastante atenção, em que as caixas delimitadoras devem se ajustar firmemente aos contornos de cada objeto. Além disso, a quantidade de instâncias de cada objeto é importante, sendo recomendado ao menos 1 500 imagens e cerca de 10 000 instâncias anotadas (Ultralytics, 2023).

Um *dataset* amplamente utilizado na visão computacional para tarefas de detecção em cenas complexas do cotidiano é o *Microsoft Common Objects in Context* (MS COCO). O COCO apresenta um número elevado de instâncias por categoria, contendo aproximadamente 2,5 milhões de objetos rotulados distribuídos em 328 mil imagens, abrangendo 91 categorias de objetos comuns (Lin, T.-Y. *et al.*, 2015). A Figura 10 a seguir mostra a quantidade de instâncias por categoria do COCO comparado com outro *dataset* popular, *Pattern Analysis, Statistical Modelling, and Computational Learning Visual Object Classes* (PASCAL VOC):

Figura 10 – Quantidade de instâncias por categoria do COCO comparado ao PASCAL VOC.



Fonte: Tsung-Yi Lin *et al.* (2015).

2.4.2 Métricas de Avaliação do Modelo

Com um *dataset* devidamente rotulado, é obtida a quantidade de *bounding boxes* reais de cada objetos (também chamadas de *ground truth*) para cada classe. Modelos de IA como a YOLO podem acabar prevendo que existem objetos onde não existe, chamado de *false-positive* (falso positivo, abreviado por FP). Esses modelos, também podem acabar deixando de prever objetos, e quando isso ocorre é chamado de *false-negative* (falso negativo, abreviado por FN). Ou seja, o modelo pode tanto errar por detectar a mais do que de fato existe de objetos, assim como errar por deixar de detectar os objetos.

A métrica que avalia a qualidade do modelo em detectar corretamente apenas os objetos *true-positive* (TP), ou seja, quantas de suas previsões estavam corretas, é chamada de *precision* (precisão). Já a métrica que avalia a capacidade do modelo em encontrar todos os objetos que existem na imagem é chamada de *recall* (revocação). O cálculo da *precision* e *recall* é dado pela equação 1 e equação 2, respectivamente (Szeliski, 2022).

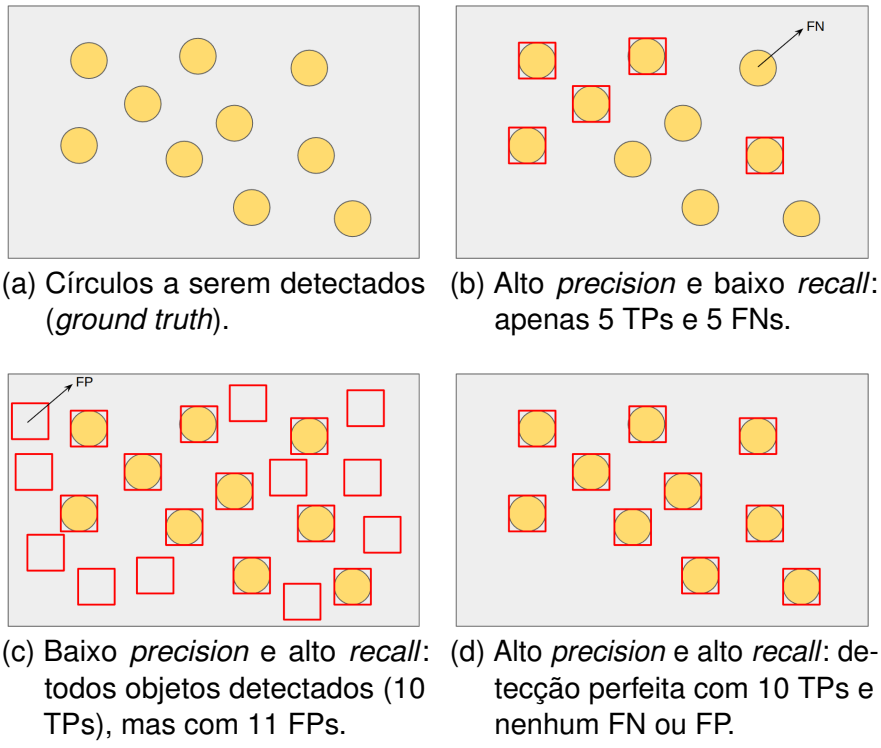
$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{1}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{2}$$

Para ilustrar na prática as métricas de *precision* e *recall*, a Figura 11 apresenta uma sequência de imagens em que um modelo de IA deve detectar 10 círculos amarelos. No primeiro cenário (Figura 11.b), o modelo detecta apenas 5 dos 10 círculos, gerando 5 falsos negativos (FN) e, portanto, apresentando um problema de *recall*. Entretanto, não há falsos positivos (FP), o que resulta em precisão perfeita. No segundo cenário (Figura 11.c), o modelo encontra todos os 10 círculos, alcançando um *recall* alto, mas também faz detecções adicionais incorretas, aumentando os FPs e reduzindo a precisão. Dessa forma, um modelo confiável deve identificar corretamente todas as

instâncias reais sem gerar detecções excedentes, mantendo altos níveis de *precision* e *recall* simultaneamente (Figura 11.d).

Figura 11 – Ilustração da *precision* e *recall*.



(a) Círculos a serem detectados (*ground truth*).

(b) Alto *precision* e baixo *recall*: apenas 5 TPs e 5 FNs.

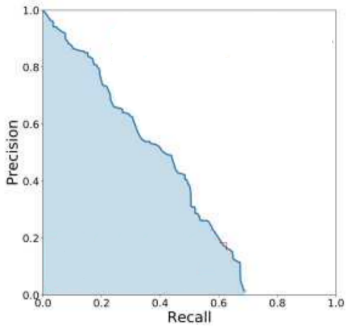
(c) Baixo *precision* e alto *recall*: todos objetos detectados (10 TPs), mas com 11 FPs.

(d) Alto *precision* e alto *recall*: detecção perfeita com 10 TPs e nenhum FN ou FP.

Fonte: Elaborado pelo autor (2025).

Portanto, analisar *precision* e *recall* isoladamente não é suficiente para avaliar o desempenho do modelo. A métrica *Average Precision* (AP) resolve esse problema ao calcular a área da curva dessas duas métricas, resultando na precisão média do modelo.

Figura 12 – Curva da *precision* pelo *recall*, formando a *Average Precision* (AP).



Fonte: Imagem adaptada de Szeliski (2022).

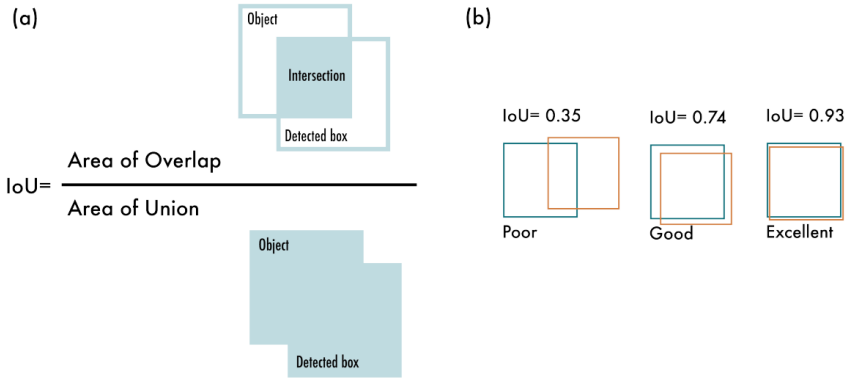
No entanto, na visão computacional de objetos 2D, definir se um modelo encontrou um objeto na cena pode ser ambíguo, pois existem diversas possíveis

delimitações do objeto. A *bounding box* pode estar cobrindo apenas parcialmente o objeto ou ainda englobando uma área muito maior do que a região efetivamente ocupada por ele na imagem. Por isso, para determinar se um modelo encontrou corretamente um objeto, é utilizado a métrica *Intersection over Union* (*IoU*).

A *IoU* representa a razão entre a área de sobreposição (*overlap*) e a área de união (*union*) das caixas delimitadoras. A área de sobreposição corresponde à interseção entre a *bounding box* real do objeto (*ground truth*) e a *bounding box* prevista pelo modelo. Já a área de união é definida como a soma das áreas dessas duas caixas, subtraída da área de interseção. Assim, quanto maior a sobreposição entre as caixas, maior será o valor da *IoU*, indicando uma previsão mais precisa (Terven *et al.*, 2023).

A Figura 13 ilustra visualmente o cálculo da *IoU*, onde em (a), observa-se a área de sobreposição (*Area of Overlap*) entre a caixa prevista e a caixa de verdade de terreno, representando o numerador da razão, enquanto a área total combinada das duas caixas (*Area of Union*) compõe o denominador. Em (b), são apresentadas interpretações qualitativas dos valores de *IoU*, categorizando-os como Ruim, Bom e Excelente, conforme o grau de sobreposição (Terven *et al.*, 2023).

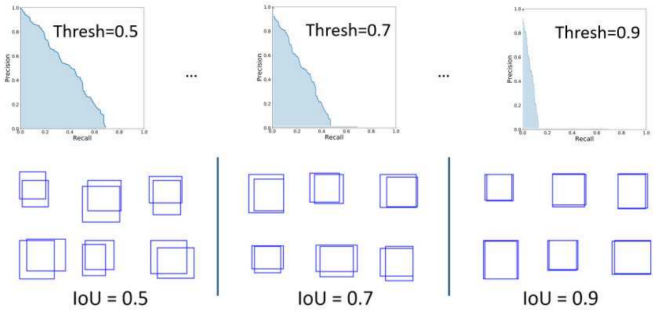
Figura 13 – Ilustração do cálculo da *Intersection over Union* (*IoU*).



Fonte: Terven *et al.* (2023).

A partir disso, é definida uma métrica ainda mais robusta que a AP vista anteriormente. Ao analisar a AP em diferentes limiares (*Threshold*) de *IoU*, é possível construir a métrica *Mean Average Precision 50-95* (*mAP50-95*). A *mAP50-95* calcula a AP em diferentes limiares de *IoU*, variando de 0,5 a 0,95, com incremento de 0,05. Dessa forma, a *mAP50-95* reflete não apenas a capacidade do modelo em identificar corretamente os objetos, mas também a precisão na localização desses objetos dentro da imagem a altos níveis de precisão.

Figura 14 – Cálculo da AP para diferentes limiares (*Threshold* de *IoU*).



Fonte: Szeliski (2022).

2.4.2.1 Matriz de Confusão

Uma das métricas para avaliação do modelo gerado automaticamente pela biblioteca da Ultralytics é a matriz de confusão. Essa matriz possui número de linhas e colunas igual ao total de classes utilizadas no treinamento, onde a linha representa os objetos previstos, e a coluna representa os objetos esperados (ou seja, os objetos verdadeiros do treinamento). O cruzamento entre a mesma classe nas linhas e colunas representa os verdadeiros positivos (acertos). Por outro lado, os cruzamentos entre classes diferentes, como entre a classe 2 e a classe 3 (Figura 15), indicam erros de classificação, ou seja, falsos positivos.

Além das classes do modelo, a matriz inclui uma linha e uma coluna adicionais correspondentes ao *background*, que representa o plano de fundo da imagem (ou seja, regiões onde não há objetos anotados). Valores presentes na coluna do *background* indicam que partes do cenário foram erroneamente classificadas como objetos, enquanto valores na linha do *background* indicam que objetos reais foram ignorados pelo modelo e tratados como fundo (Ultralytics, 2023). A Figura 15 ilustra essa estrutura, destacando em verde os acertos e, em amarelo, os erros de classificação entre classes e com o *background*.

Figura 15 – Exemplo de leitura de uma matriz de confusão: em verde, predições corretas; em amarelo, confusões entre classes e com o *background*.

Classes		Esperado				
		Classe 1	Classe 2	Classe 3	Classe 4	Background
Previsão	Classe 1					
	Classe 2					
	Classe 3					
	Classe 4					
	Background					

Fonte: Elaborado pelo autor (2025).

2.4.3 Processo de Treinamento

Segundo Redmon *et al.* (2016), o *backbone* da YOLOv1 foi pré-treinado por uma semana utilizando o conjunto de dados ImageNet, que contém 1 000 classes. Em seguida, esse modelo foi submetido a um novo treinamento utilizando o conjunto de dados PASCAL VOC, que possui apenas 20 classes.

Essa técnica de aplicar um novo treinamento sobre um modelo pré-treinado é conhecida como transferência de aprendizagem (*transfer learning*). Como explicado na Seção 2.2.4, a detecção e classificação são obtidas pelas características extraídas pelos *kernels*, que possuem parâmetros de pesos ajustáveis. Dessa forma, Redmon *et al.* (2016) utilizou o conjunto ImageNet para fazer um treinamento extenso e assim obter pesos otimizados para os *kernels* de sua rede. Posteriormente, foi feito o ajuste fino (*fine-tuning*) da rede, reutilizando os *kernels* pré-treinados e adaptando-os para um novo conjunto de classes.

No caso da YOLOv8, cada uma de suas versões foi pré-treinada utilizando o *dataset* COCO (Ultralytics, 2023). Similarmente à YOLOv1, após a escolha de um modelo base, este pode ser treinado por *transfer learning* para qualquer outro conjunto de dados, inclusive conjuntos de dados customizados desenvolvidos para aplicações específicas.

Durante o treinamento, o modelo percorre o conjunto de treino ajustando seus pesos ao tentar detectar e classificar corretamente os objetos nas imagens. Em seguida, ele testa essas predições no conjunto de validação. São aplicadas duas funções de perda: *Complete Intersection over Union* (*CIoU*) e *Distribution Focal Loss* (*DFL*) (Terven *et al.*, 2023). A partir desses resultados, os pesos são ajustados e um novo ciclo de tentativa ocorre. Esse processo se repete durante o número de épocas (*epochs*) definido previamente pelo treinador durante a configuração do treinamento.

Por padrão, o algoritmo de treinamento da Ultralytics (2023) realiza uma expansão do *dataset* de treino por meio da criação de novas imagens em mosaico, geradas a partir das imagens originais e reutilizando os mesmos *labels*. Essa técnica, que consiste em copiar imagens do próprio conjunto de dados e aplicar distorções ou modificações computacionais para gerar novas amostras, é conhecida como *data augmentation* no contexto de treinamento de detectores de objetos.

Ao final, dois arquivos principais são gerados: `last.pt`, que representa o modelo salvo na última época, e `best.pt`, que armazena o modelo com o melhor desempenho no conjunto de validação. Esse desempenho é medido automaticamente com base em uma métrica chamada de *mAP50-95*. Sempre que essa métrica apresenta uma melhora ao longo das épocas, um novo `best.pt` é salvo, garantindo que o modelo mais eficaz em detectar objetos seja preservado. Essa estratégia evita que o modelo final seja aquele que apenas treinou mais, mas sim aquele que performou melhor para dados nunca vistos (Ultralytics, 2023).

2.4.4 Desempenho da YOLO em Acurácia e Tempo de Inferência no *Dataset* MS COCO

Como explicado na Subseção 2.3.1, a Ultralytics lançou cinco variações do modelo YOLOv8, cada uma com diferentes números de parâmetros (quantidade de pesos e viéses) que simbolizam sua complexidade computacional. Essas variações (*nano, small, medium, large e extra-large*) apresentam um balanço entre acurácia e tempo de processamento das detecções.

O processamento das detecções, também chamado de inferência, pode ser realizado utilizando a Unidade Central de Processamento (*Central Processing Unit* - CPU) ou a Unidade de Processamento Gráfico (*Graphics Processing Unit* - GPU). As GPUs da empresa NVIDIA possuem a tecnologia CUDA (*Compute Unified Device Architecture*), que permite acelerar significativamente o processamento em comparação com CPUs convencionais (Harris, 2017). A *YOLOv8* possui compatibilidade tanto com CPUs quanto com GPUs NVIDIA, oferecendo flexibilidade na escolha do hardware de inferência.

A Tabela 1 apresenta uma comparação entre as diferentes variações do modelo YOLOv8 com base em métricas de desempenho. São exibidos o tamanho da imagem de entrada utilizada durante a inferência, a métrica *mean Average Precision* (mAP) calculada entre os limiares de *Intersection over Union* (*IoU*) de 0,5 a 0,95, o tempo médio de inferência em CPU e GPU (placa gráfica NVIDIA A100), o número de parâmetros (em milhões) e a quantidade de operações de ponto flutuante por segundo (FLOPs, em bilhões). Esses indicadores permitem avaliar de forma abrangente a precisão, a eficiência computacional e a complexidade de cada modelo, fornecendo orientação inicial para a escolha da versão mais adequada conforme os requisitos específicos da aplicação.

Tabela 1 – Desempenho dos modelos YOLOv8 para uma imagem com 640 *pixels* de tamanho.

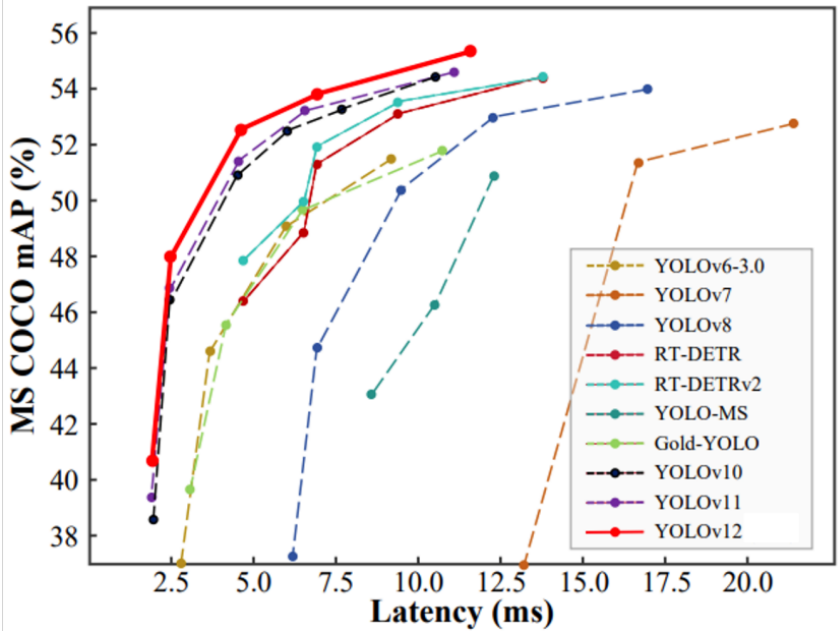
Modelo	**mAPval 50-95**	**Vel. CPU (ms)**	**Vel. A100 (ms)**	**Params (M)**	**FLOPs (B)**
YOLOv8n	37,3	80,4	0,99	3,2	8,7
YOLOv8s	44,9	128,4	1,20	11,2	28,6
YOLOv8m	50,2	234,7	1,83	25,9	78,9
YOLOv8l	52,9	375,2	2,39	43,7	165,2
YOLOv8x	53,9	479,1	3,53	68,2	257,8

Fonte: Ultralytics (2023).

Enquanto este trabalho foi desenvolvido, foram publicadas novas versões da YOLO, sendo a mais recente a YOLOv12 (Khanam; Hussain, 2025). No entanto, como mostra a Figura 16, a diferença percentual na métrica *mAPval 50-95* entre as versões YOLOv12 e YOLOv8 é de aproximadamente 2%. A principal evolução das versões mais recentes está relacionada à redução da latência de inferência, atingindo velocidades

progressivamente melhores. Portanto, apesar das publicações mais recentes, o modelo YOLOv8 mantém-se relevante em termos de acurácia de detecção.

Figura 16 – Comparação entre os modelos mais atuais da YOLO.



Fonte: Khanam e Hussain (2025).

2.5 Rastreadores de Múltiplos Objetos

O rastreamento de objetos (*Object Tracking*) é uma subárea da visão computacional que evolui o conceito de detecção de objetos, buscando acompanhar automaticamente um único objeto detectado ao longo do tempo em uma sequência de vídeo. A partir da detecção inicial, o sistema é capaz de estimar a trajetória desse objeto nos quadros seguintes. Quando a tarefa envolve o acompanhamento simultâneo de múltiplos objetos em cena, ela é denominada *Multiple Object Tracking* (MOT). Nesse caso, o sistema atribui um identificador único (ID) a cada instância detectada, garantindo a manutenção de suas identidades ao longo do tempo (Abouelyazid, 2023).

A associação de ID pode ser realizada por meio de diferentes técnicas, como o uso de modelos baseados em movimento, utilizando filtros de Kalman (Li *et al.*, 2015), ou por meio da extração de características visuais com redes neurais para comparar a aparência dos objetos entre os quadros. O principal desafio do MOT é garantir a consistência da identificação mesmo em situações complexas, como oclusões, mudanças abruptas de trajetória, variações de iluminação e densidade elevada de objetos em movimento (Abouelyazid, 2023).

A métrica *Multiple Object Tracking Accuracy* (MOTA) é amplamente utilizada para avaliar o desempenho de algoritmos de rastreamento de múltiplos objetos. Ela

combina três tipos de erros comuns: falsos positivos (FP), falsos negativos (FN) e trocas de identidade (*Identities Switch* - IDSW). A fórmula para calcular o MOTA é:

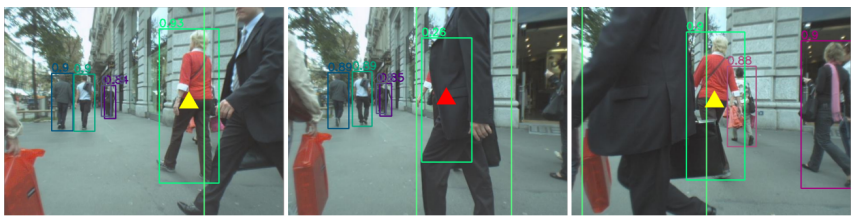
$$\text{MOTA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{IDSW}_t)}{\sum_t \text{GT}_t}$$

Onde FN_t, FP_t e IDSW_t representam, respectivamente, os falsos negativos, falsos positivos e trocas de identidade no quadro t, e GT_t é o número de objetos reais presentes nesse quadro. Um rastreador perfeito teria um MOTA de 100%, indicando ausência de erros. Essa métrica fornece uma visão abrangente do desempenho do rastreador, penalizando erros de detecção e de associação de identidades (Murray, 2017).

A partir de métricas como o MOTA, é possível realizar comparações padronizadas entre diferentes algoritmos de rastreamento, como ocorre no MOTChallenge (Desafio MOT), uma iniciativa que disponibiliza conjuntos de dados e protocolos de avaliação para promover a reprodutibilidade na área de rastreadores (MOTCHALLENGE. . . , 2015). Nesse contexto, Abouelyazid (2023) avaliou alguns dos principais algoritmos da área e destacou o desempenho superior do ByteTrack, que obteve um MOTA de 77,3%, superando de forma significativa o DeepSORT (Wojke *et al.*, 2017) (61,4%) e o SORT (Bewley *et al.*, 2016) (54,7%).

Diferentemente de métodos tradicionais que descartam detecções com baixa pontuação de confiança (como Deepsort e Sort), o ByteTrack (Zhang *et al.*, 2022) considera quase todas as detecções, incluindo aquelas com pontuações de confiança mais baixas, para melhorar a continuidade das trajetórias, especialmente em casos de oclusão. A associação de IDs é realizada em duas etapas: inicialmente, as detecções com alta pontuação são associadas às trajetórias existentes utilizando o filtro de Kalman e o algoritmo de associação húngaro (Kuhn, 1955); em seguida, as detecções restantes com pontuação mais baixa são associadas às trajetórias não correspondidas da primeira etapa. Essa estratégia permite recuperar objetos verdadeiros que poderiam ser ignorados devido a pontuações de detecção reduzidas, resultando em melhorias significativas nas métricas de rastreamento, ilustrado pela figura 17 (Zhang *et al.*, 2022).

Figura 17 – Exemplo do rastreamento Bytetrack, o triângulo vermelho sinaliza o objeto com obstrução pelo baixo *score* de confiança.



Fonte: Zhang *et al.* (2022).

2.6 Trabalhos Relacionados

O Quadro 3 apresenta os principais trabalhos relacionados que utilizam métodos de aprendizado profundo para desenvolver sistemas de detecção, classificação e contagem de veículos. O quadro organiza as informações em quatro aspectos fundamentais: o título do artigo, o resumo dos objetivos propostos pelos autores, a síntese dos principais resultados alcançados e uma análise crítica quanto à contribuição e limitações desses estudos em relação ao presente trabalho. Esta análise comparativa permite identificar tanto os aspectos que fundamentam teoricamente esta pesquisa quanto as lacunas metodológicas e técnicas que este estudo pode preencher no estado da arte atual.

Quadro 3 – Trabalhos relacionados.

Nome do artigo	**Resumo do objetivo**	**Resumo do resultado**	**Análise crítica**
Detecção, Rastreamento e Contagem de Veículos para Análise de Tráfego Urbano utilizando Métodos de Aprendizagem Profunda (Sampaio; Linhares, 2024)	Desenvolver um sistema de detecção (YOLOv8), rastreamento (Deepsort) e contagem em tempo real de veículos em ambientes urbanos	O experimento avaliou o desempenho sobre um vídeo de 10 segundos, obteve contagem total precisa (24 veículos), mas apresentou erros de classificação por classe, principalmente para ônibus, devido à visibilidade parcial e desbalanceamento do *dataset*.	Apresenta uma metodologia funcional do sistema (treinamento, detecção, rastreamento) que convergem com os objetivos desse trabalho. Porém, o banco de dados de *transfer learning* utilizado não é de vias brasileiras, e a amostra é de apenas 10 segundos, o que não demonstra a eficácia em um período mínimo de 15 minutos e nem em diferentes períodos do dia.
A YOLO-based Traffic Counting System (Lin, J.-P.; Sun, 2018)	Desenvolver um sistema de contagem de veículos em tempo real utilizando a tecnologia de reconhecimento de imagens baseada na arquitetura YOLO, aplicado a vídeos de monitoramento em Sistemas de Transporte Inteligente	O sistema atingiu 100% de acurácia na contagem de veículos durante o dia e 80% à noite, com falhas causadas por superexposição dos faróis. Em testes com melhor iluminação mantiveram 100% de acurácia mesmo à noite.	Utiliza YOLO básico sem *transfer learning* nem algoritmos de rastreamento, o que limita a contagem a movimentos lineares e reduz robustez em cenários urbanos complexos.

Nome do artigo	**Resumo do objetivo**	**Resumo do resultado**	**Análise crítica**
Vehicle Class Detection and Counting on a Malaysian Road Using YOLOv8 and OpenCV (Hassan *et al.*, 2024)	Aplicar o algoritmo YOLOv8 em conjunto com OpenCV para contagem e classificação de veículos em tempo real em uma rodovia na Malásia (Changlun–Kuala Perlis).	O modelo alcançou 96% de acurácia na contagem de veículos e 94,08% de acurácia na classificação de veículos entre carro, motocicleta, ônibus e caminhão.	Demonstra a viabilidade do modelo YOLOv8 para contagens de vídeos de rodovias em tempo real. No entanto, se limita à inferência com um modelo pré-treinado, sem realizar treinamento com um *dataset* personalizado. Além disso, não explora técnicas mais avançadas de rastreamento, o que restringe a capacidade da contagem direcional de veículos.

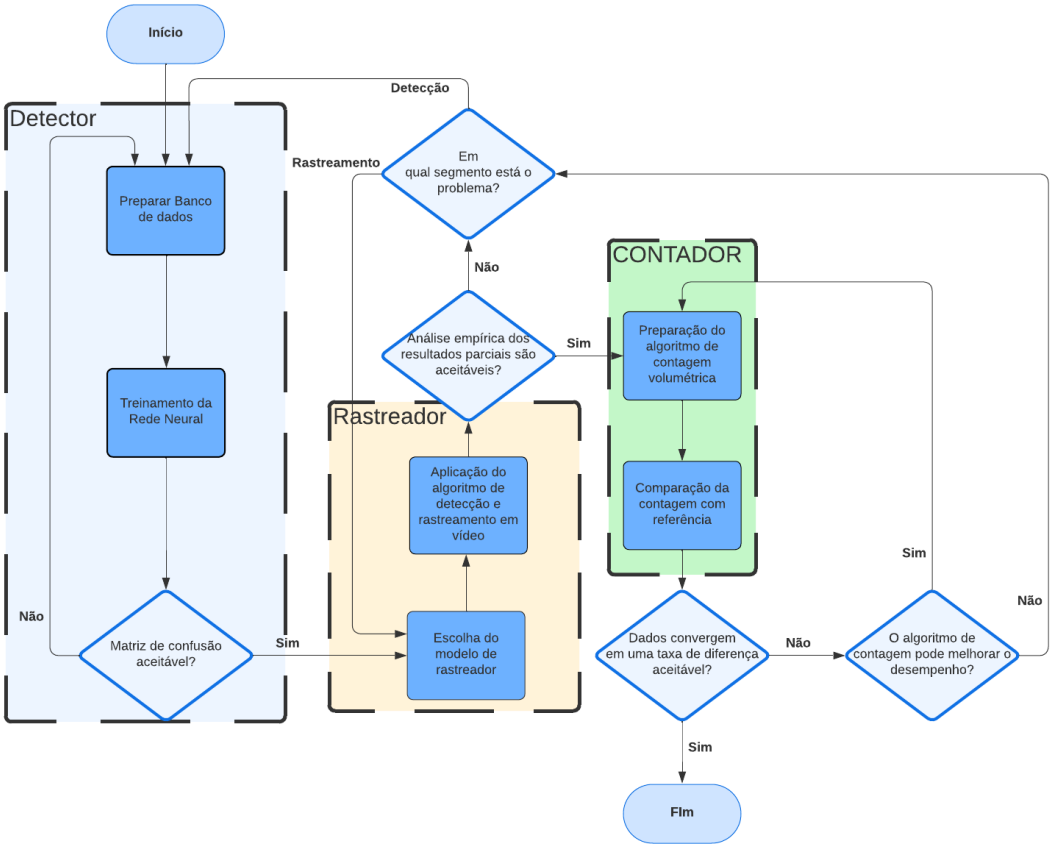
3 PROCEDIMENTOS METODOLÓGICOS E DESENVOLVIMENTO

Seguindo a estrutura adotada nos principais trabalhos relacionados, a metodologia proposta neste trabalho está organizada em três módulos principais: detecção, rastreamento e contagem. Cada uma dessas etapas desempenha um papel fundamental no processo de análise de tráfego veicular, desde a identificação e classificação dos objetos de interesse até o acompanhamento de suas trajetórias e a sua contabilização efetiva.

De forma geral, este trabalho foi desenvolvido com base em um ciclo iterativo entre esses três componentes. Ou seja, mesmo após a conclusão de uma etapa, caso os resultados preliminares não fossem satisfatórios, realizava-se uma reavaliação dos algoritmos utilizados, técnicas de treinamento ou lógicas de programação aplicadas na contagem. Esse processo se repetiu diversas vezes até que se atingissem os resultados apresentados no Capítulo 4.

A Figura 18 apresenta o fluxograma do desenvolvimento, indicando os módulos principais, o fluxo de execução e as decisões adotadas em cada etapa do projeto.

Figura 18 – Fluxograma de desenvolvimento.



Fonte: o Autor.

A seguir, cada componente será detalhado, destacando as técnicas, algoritmos e decisões adotadas ao longo do desenvolvimento.

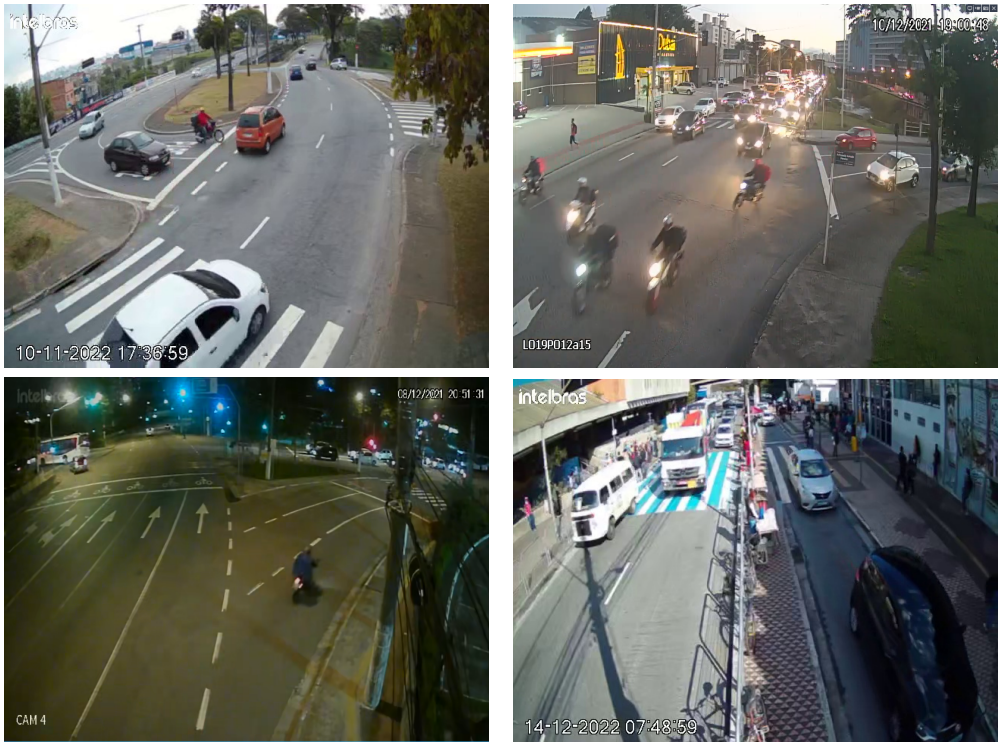
3.1 Detecção

A etapa de detecção tem como função localizar e classificar os veículos presentes em cada quadro do vídeo. Para essa tarefa, foi utilizada a rede neural YOLOv8. Embora essa rede esteja disponível com pesos pré-treinados no conjunto de dados COCO, optou-se por treinar um novo modelo a partir de um banco de imagens próprio, elaborado com base nas filmagens fornecidas pela empresa Consultran Engenharia. As seções a seguir apresentam em detalhes o processo de treinamento do modelo, bem como as configurações de ambiente e as técnicas empregadas.

3.1.1 Conjunto de Dados

Para montar o *dataset* de treinamento, foi utilizado imagens retiradas de vídeos disponibilizados pela Consultran Engenharia. Todos os vídeos tem o padrão descrito na Subseção 2.1.1, cujas principais características são a projeção oblíqua dos veículos, imagens coloridas durante a noite e resolução de 480p. A Figura 19 a seguir exemplificam a composição geral das imagens:

Figura 19 – Exemplo de imagens retiradas de vídeos fornecidos pela empresa Consultran Engenharia.



Fonte: Elaborado pelo autor (2025).

O objetivo da contagem classificada de veículos, conforme adotado pela empresa Consultran Engenharia, é categorizar todos os veículos que trafegam nas classes: automóvel, motocicleta, ônibus, caminhão e ciclista. Ou seja, é necessário compreender que veículos como vans, caminhonetes, carros de passeio, Kombi, minivans e até mesmo carros-fortes devem ser classificados como automóvel. A Figura 20 a seguir ilustra como cada silhueta veicular deve ser classificada:

Figura 20 – Principais silhuetas de veículos e como devem ser classificadas segundo a empresa Consultran Engenharia.

(a) Automovel (b) Automovel (c) Automovel

(d) Automovel (e) Automovel (f) Automovel

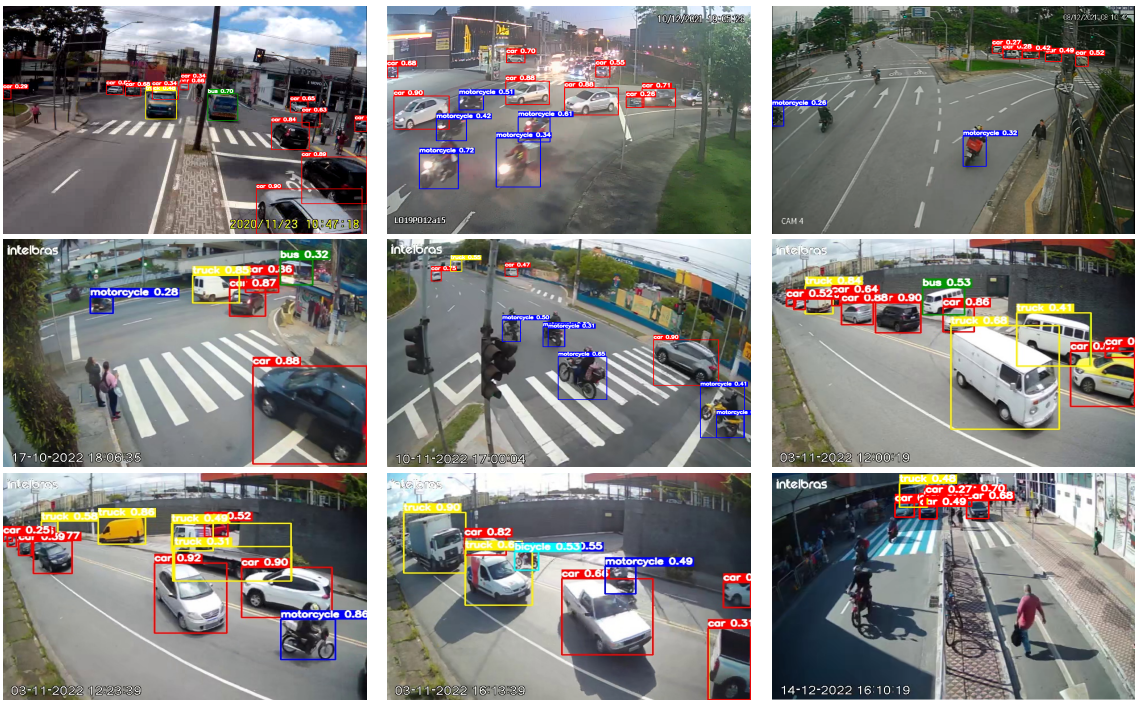
(g) Ônibus (h) Caminhão (i) Motocicleta

Fonte: Elaborado pelo autor (2025).

Após alguns testes iniciais com o modelos pré-treinado da YOLOv8m, foram identificados problemas na classificação de veículos. O *dataset* COCO, utilizado no pré-treinamento, possui 80 classes variadas, das quais apenas cinco são relevantes para a contagem veicular: *car*, *motorcycle*, *bus*, *truck* e *bicycle*. Essa limitação conceitual resultou em erros sistemáticos de categorização, uma vez que diversas silhuetas de veículos comuns no tráfego brasileiro não eram classificadas conforme a necessidade da empresa Consultran Engenharia. Verificaram-se inconsistências recorrentes, como veículos com silhueta de caminhonete sendo rotulados ora como *car*, ora como *truck*, enquanto a Kombi foi frequentemente classificada como *bus* ou *truck*. A Figura 21 apresenta um mosaico com algumas dessas confusões de classificação que não

atendem às demandas da Consultran Engenharia, evidenciando ainda a baixa confiança (inferior a 70%) na detecção da maioria das motocicletas.

Figura 21 – Inconsistências de classificação do modelo YOLOv8m pré-treinado. Legenda de cores das *bounding boxes*: vermelho (*car*), azul marinho (*motorcycle*), ciano (*bicycle*), verde (*bus*), amarelo (*truck*).



Fonte: Elaborado pelo autor (2025).

Diante disso, verificou-se a necessidade de preparar um novo *dataset* que agrupasse as silhuetas veiculares conforme a categorização apresentada na Figura 20. Para compor esse conjunto de dados, foram utilizadas imagens extraídas de filmagens com as mesmas condições técnicas e ambientais características das pesquisas realizadas pela Consultran Engenharia.

Sabe-se que a YOLO utiliza *feature maps* para reconhecer padrões visuais nas imagens durante o processo de classificação. Visualmente, certos veículos se distanciam da silhueta típica de um carro de passeio. Assim, concluiu-se que não seria adequado treinar o novo modelo considerando veículos como caminhonetes, vans, minivans, carros-fortes e Kombi diretamente na classe automóvel. Optou-se, portanto, por adotar dez categorias distintas no treinamento: *car*, *motorcycle*, van, caminhonete, *truck*, *bus*, minivan, *bicycle*, Kombi e carroforte.

Figura 22 – Classes de veículos motorizados definidos para o treinamento do modelo.

(a) *car* (b) caminhonete (c) *van*

(d) minivan (e) *carroforte* (f) Kombi

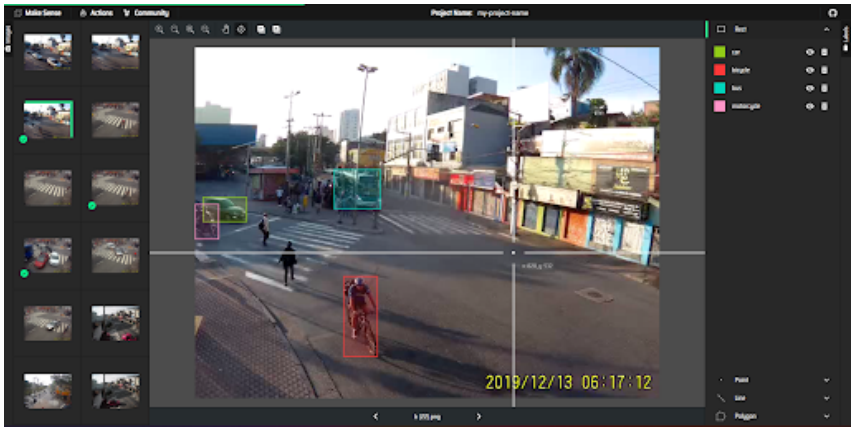
(g) *bus* (h) *truck* (i) *motorcycle*

Fonte: Elaborado pelo autor (2025).

Apesar de o objetivo deste trabalho ser a contagem efetiva apenas de veículos motorizados, a classe *bicycle* foi incluída no treinamento com o intuito de diferenciar corretamente motocicletas de bicicletas. Compreendeu-se que, caso as bicicletas fossem completamente ignoradas durante o treinamento, o modelo poderia confundi-las com motocicletas ao analisá-las nos vídeos, gerando assim falsos positivos na contagem.

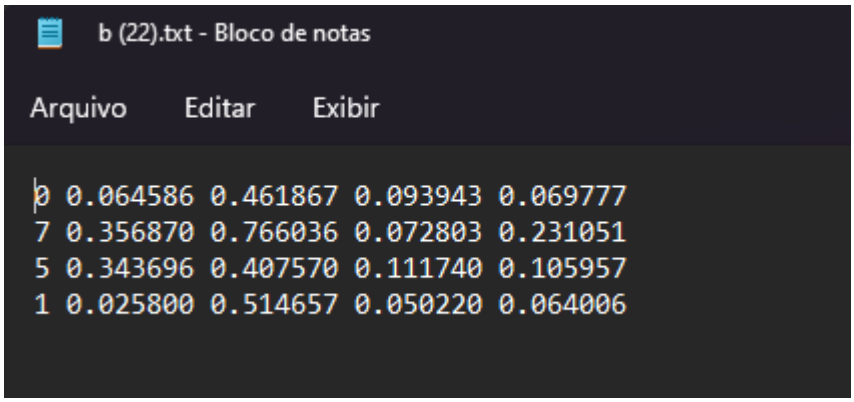
Com os *labels* definidos, o passo seguinte foi preparar o *dataset* para o treinamento. Para isso, utilizou-se a plataforma online Makesense (Skalski, 2019), que permite fazer o *upload* das imagens e segmentar manualmente cada objeto presente (Figura 23). Ao final do processo, a ferramenta exporta os dados em arquivos no formato `.txt`, compatíveis com o treinamento de modelos YOLOv5 e YOLOv8, conforme ilustrado pela Figura 24.

Figura 23 – Segmentação de objetos na plataforma Makesense.



Fonte: Elaborado pelo autor (2025).

Figura 24 – Exemplo dos *labels* retornados pela plataforma Makesense. Cada linha é um objeto diferente e a ordem das colunas (separadas por espaço) são: classe, posição x, posição y, largura e altura.



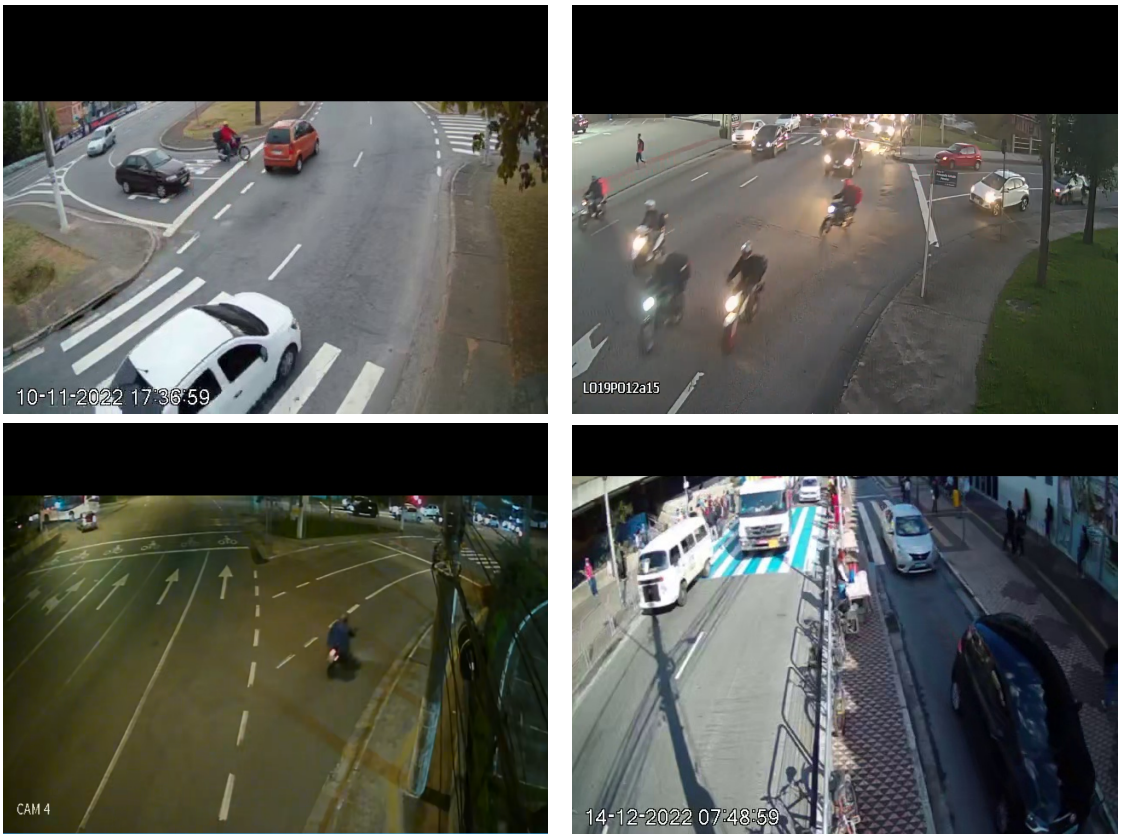
Fonte: Elaborado pelo autor (2025).

Outro aspecto importante na construção do *dataset* foi o tratamento das imagens. Devido ao ângulo de algumas filmagens, era comum a presença de veículos distantes da interseção de interesse para a contagem. Como a câmera utilizada não possui alta resolução, esses veículos distantes acabam ocupando poucos pixels na imagem, tornando-se quase indistinguíveis, mesmo para um observador humano. Dessa forma, durante a segmentação manual, frequentemente havia incerteza quanto à classe à qual o veículo realmente pertencia.

Para contornar esse problema e garantir que apenas objetos distinguíveis dentro da área de interesse fossem considerados, foi aplicado um procedimento denominado máscara. Cada imagem do *dataset* foi analisada individualmente, e regiões com veículos muito pequenos ou visualmente irreconhecíveis foram cobertas com blocos retangulares pretos, simulando um recorte da imagem. A Figura 25 apresenta

as mesmas imagens mostradas anteriormente na Figura 19, agora com a *máscara* aplicada.

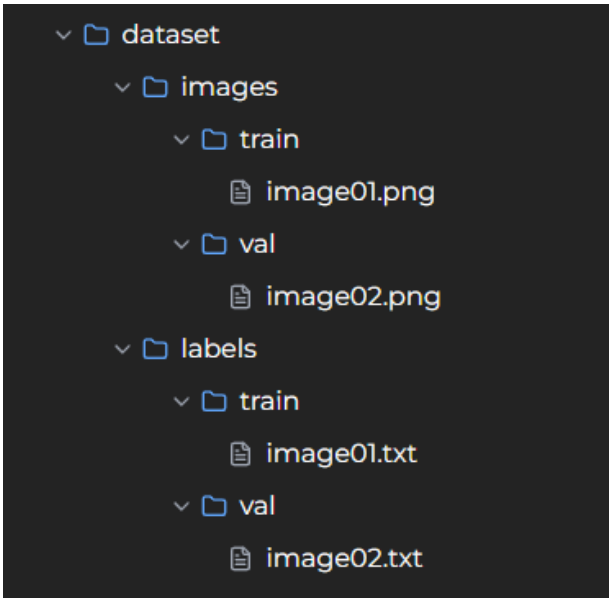
Figura 25 – Exemplo de imagens com *máscara* para evitar veículos indistínguiveis a olho humano.



Fonte: Elaborado pelo autor (2025).

Ao todo, foram preparadas 2.744 imagens. Destas, 2.187 imagens (79,71%) foram utilizadas no subconjunto de treinamento e 557 imagens (20,29%) no subconjunto de validação. A metodologia adotada para a divisão dos conjuntos foi uma seleção arbitrária, com o objetivo de balancear e diversificar adequadamente as amostras destinadas à validação. A Figura 26 apresenta a organização estrutural do conjunto de dados preparado.

Figura 26 – Estrutura do *dataset*.



Fonte: Elaborado pelo autor (2025).

Como a plataforma Makesense impõe um limite no número de imagens que podem ser segmentadas simultaneamente, optou-se por organizá-las em lotes de aproximadamente 50 imagens. Após a segmentação manual dos lotes, cerca de 10 imagens de cada lote foram destinadas à validação. Esse procedimento possibilitou manter tanto a diversificação quanto a proporção desejada de aproximadamente 80% das imagens para treinamento e 20% para validação.

3.1.2 Treinamento

Com o *dataset* definido, o passo seguinte foi configurar o ambiente, as bibliotecas e demais parâmetros necessários para o treinamento do modelo. Para essa etapa, utilizou-se a plataforma online Google Colab, que disponibiliza recursos computacionais, incluindo GPUs. Devido à alta demanda por processamento, foi contratado o plano mensal Colab Pro, que oferece acesso a até 100 unidades computacionais por mês.

A preparação do ambiente no Google Colab é relativamente simples: foi necessário apenas instalar a biblioteca Ultralytics, conforme disponibilizado em Ultralytics (2023). Em seguida, realizou-se o *upload* do *dataset*, acompanhado de um arquivo no formato `.yaml`, responsável por definir os caminhos dos subconjuntos de treinamento e validação, além de especificar os nomes de cada classe. Os parâmetros utilizados no treinamento são apresentados na Quadro 4 a seguir.

Quadro 4 – Parâmetros utilizados no *transfer learning*.

Parâmetro	**Valor**
Modelo	yolov8m.pt
Epochs	300
Batch size	16
Tamanho da imagem	640

Fonte: Elaborado pelo autor (2025).

O parâmetro "modelo" refere-se a um modelo pré-treinado e disponibilizado pela Ultralytics (2023). Para esse trabalho foi escolhido o modelo `yolov8m.pt` (Ultralytics, 2023), porque apresenta o desempenho mais equilibrado entre qualidade de detecção e velocidade de inferência, de acordo com a Tabela 1 da Seção 2.3.1.

3.2 Rastreamento

O repositório apresentado por Broström (2023) implementa os algoritmos da YOLOv8 em Python, integrando-os com diferentes modelos de rastreadores: BoT-SORT, ByteTrack, DeepOCSORT, OCSORT e StrongSORT. Com base nas principais métricas de avaliação de rastreamento, como o MOTA (Papers with Code, 2023), optou-se pela utilização do rastreador ByteTrack, devido à sua alta pontuação.

Além disso, foi necessário configurar o ambiente de inferência para as etapas de detecção e rastreamento. Considerando que o objetivo era processar vídeos longos e de grande volume, a utilização do Google Colab para a inferência não se mostrou viável. Assim, foi configurado um ambiente Python em um computador físico com GPU, disponibilizado pela empresa Consultran Engenharia.

Quadro 5 – Especificações do computador disponível.

Tipo	**Componente**
Sistema operacional	Windows 10
Processador	i5-10400 2.90 GHz
Memória RAM	16 GB 2400 Hz
GPU	gtx 1050 2GB

Fonte: Elaborado pelo autor (2025).

A Tabela 2 a seguir apresenta as bibliotecas necessárias para a configuração do ambiente.

Tabela 2 – Bibliotecas utilizadas e suas versões.

Biblioteca	**Versão**
Python	$\geq$ 3.8
filterpy	$\geq$ 1.4.5
gdown	$\geq$ 4.7.1
GitPython	$\geq$ 3.1.0
lapx	$\geq$ 0.5.4
loguru	$\geq$ 0.7.0
numpy	= 1.23.1
opencv-python	$\geq$ 4.6.0
pandas	$\geq$ 1.1.4
PyYAML	$\geq$ 5.3.1
tensorboard	$\geq$ 2.13.0
torch	$\geq$ 1.7.0
torchvision	$\geq$ 0.8.1

Fonte: Elaborado pelo autor (2025)

Para organizar os pacotes, é comum utilizar uma ferramenta gerenciadora de ambientes virtual. No caso desse trabalho foi utilizado a ferramenta Anaconda (Anaconda Inc., 2025), pois já possui pacotes pré instalados e é apropriada para desenvolvimento de projetos de aprendizado de máquina e ciência de dados.

Para iniciar o algoritmo, os principais parâmetros são apresentados no Quadro 6.

Quadro 6 – Principais parâmetros do algoritmo de rastreamento.

Parâmetro	**Descrição**
`track.py`	Script responsável pela execução do rastreamento
`save-vid`	Salva o vídeo com o desenho das detecções e rastreamento
`save-txt`	Salva os resultados de rastreamento em arquivos `.txt`
`tracking-method`	Define o modelo de rastreamento a ser utilizado
`yolo-weight`	Caminho para o modelo YOLOv8 previamente treinado
`img`	Define a resolução de inferência analisada pelo modelo
`conf-thres`	Define o limiar mínimo de confiança para considerar uma detecção válida
`source`	Caminho para o vídeo de entrada a ser processado
`show-vid`	Exibe o vídeo em tempo real durante a inferência

Fonte: Elaborado pelo autor (2025).

Os resultados salvos nos arquivos `.txt` possuem a estrutura de campos apresentada no Quadro 7.

Quadro 7 – Estrutura dos campos nos arquivos de saída do rastreamento.

Campo	**Descrição**
frame	Índice do quadro do vídeo em que a inferência foi realizada
coordenada x	Posição horizontal do canto superior esquerdo da *bounding box*, em *pixels*
coordenada y	Posição vertical do canto superior esquerdo da *bounding box*, em *pixels*
largura	Dimensão horizontal do retângulo delimitador, em *pixels*
altura	Dimensão vertical do retângulo delimitador, em *pixels*
confiança	Número decimal que indica a certeza da detecção e classificação
classe	Número inteiro que representa o índice da lista de *labels* definida durante o treinamento

Fonte: Elaborado pelo autor (2025).

Cada linha do arquivo corresponde a um objeto detectado. Caso haja múltiplos objetos no mesmo *frame*, o número do *frame* se repete nas linhas subsequentes.

3.3 Contagem

Os resultados brutos de detecção e rastreamento, por si só, não possuem utilidade prática sem um algoritmo capaz de interpretar as trajetórias estimadas e realizar a CVC. A importância desta etapa reside no fato de que ela consolida e traduz as trajetórias e classificações obtidas nas fases anteriores em dados quantitativos úteis para análise de tráfego.

Diferentemente dos trabalhos relacionados apresentados na Seção 2.6, este trabalho tem o objetivo de não apenas contar o fluxo e classificar veículos, mas também diferenciar os movimentos direcionais realizados em vias e interseções. Essa funcionalidade adicional exige uma abordagem específica que considere as particularidades dos cenários de aplicação.

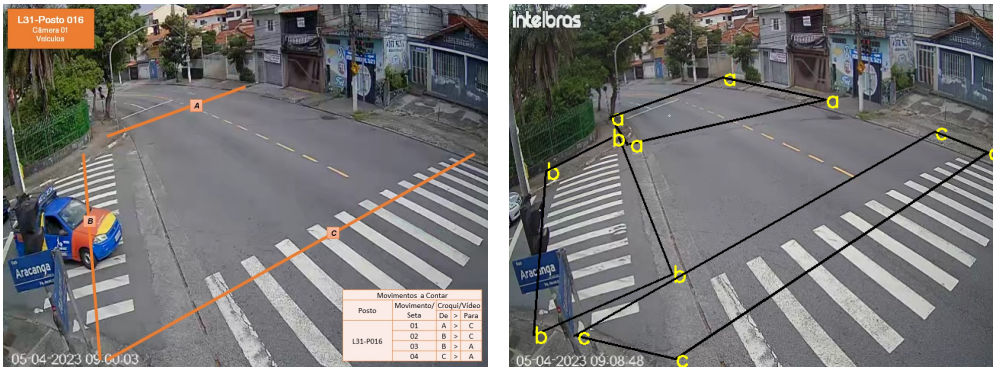
Portanto, esta seção apresenta a análise lógica deste desafio e propõe um método para atingir a contagem direcional de veículos. Posteriormente, é apresentada a implementação algorítmica desta metodologia, de modo a entregar a contagem volumétrica classificada e direcional de veículos organizada em intervalos de 15 minutos.

3.3.1 Método de Desenho de Regiões para Contagem Direcional

Para solucionar o desafio da contagem direcional de veículos, é necessário observar o croqui de movimentos elaborado pela empresa Consultran Engenharia na Seção 2.1. Basicamente, a interseção é segmentada em pontos de origem e destino (letras A, B, C, D...), em que as conexões entre essas origem e destinos formam os movimentos (por exemplo, origem no ponto A e destino no ponto C é o movimento 1).

Ou seja, o algoritmo de contagem pode seguir a mesma lógica para interpretar um movimento. Para representar esses pontos de origem e destino, optou-se por desenhar regiões quadriláteras sobre a via, como ilustra a Figura 27.

Figura 27 – Croqui de movimentos da Consultran Engenharia em (a) e em (b) um exemplo de três regiões quadriláteras de contagem.



(a) Croqui de movimentos da Consultran Engenharia

(b) Regiões quadriláteras de contagem propostas

Fonte: Elaborado pelo autor (2025).

O próximo passo é fazer com que o algoritmo consiga detectar que o veículo rastreado esteve dentro dessa região de interesse. Para representar o movimento de cada veículo, é desenhado um ponto sobre a imagem. A Figura 28 ilustra a escolha do ponto correspondente ao centro inferior da *bounding box*, definido como a base vertical da caixa e sua posição horizontal centralizada. Essa escolha foi feita em substituição ao ponto central tradicional da *bounding box*, pois oferece melhor aderência visual à posição real do veículo na via.

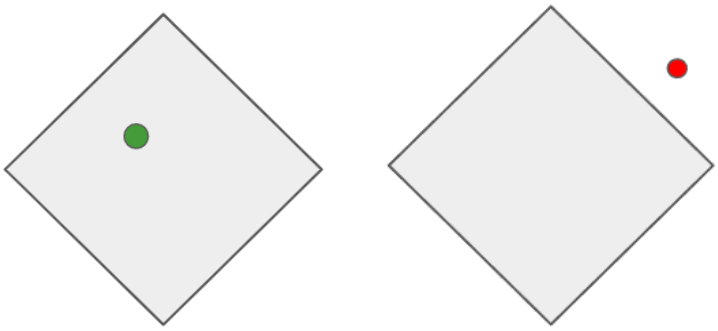
Figura 28 – Representações visuais de coordenadas das *bounding boxes* dos veículos. Em (a), a localização do ponto (x,y) e do ponto $(x+w,y+h)$. Em (b), o ponto de projeção padrão (em verde) e ponto de projeção escolhido (em amarelo).

(a) (b)

Fonte: Elaborado pelo autor (2025).

A partir do ponto de projeção do veículo e das coordenadas (vértices) de cada região, o cenário de análise passa a ser um problema de geometria analítica. Basicamente, pode-se simplificar esse cenário em um plano bidimensional, onde é preciso diferenciar se um ponto pertence ou não à uma região, como ilustrado na Figura 29.

Figura 29 – Ilustração de exemplo do cenário de verificação: o ponto verde em (a) está dentro do quadrilátero, e o ponto vermelho em (b) não está dentro.

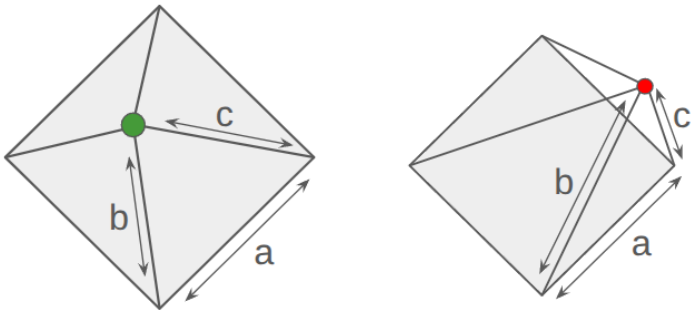


(a) Ponto dentro da região (b) Ponto fora da região

Fonte: Elaborado pelo autor (2025).

Para melhor se aderir ao formato da via, esse quadrilátero pode assumir diferentes formas e dimensões (como losangos ou trapézios, por exemplo), e o ponto do veículo pode estar posicionado em qualquer localização do plano. Para relacionar essas duas entidades (ponto e região), pode ser traçado 4 retas que conectam o ponto aos vértices da região, formando 4 triângulos, cada um definido por seus lados a, b e c, conforme ilustrado na Figura 30.

Figura 30 – Ilustração dos 4 triângulos formados ao conectar um ponto aos vértices de um quadrilátero.

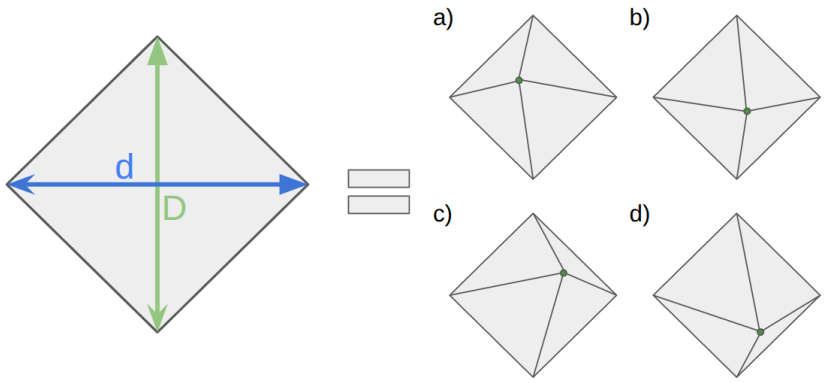


(a) Ponto dentro da região (b) Ponto fora da região

Fonte: Elaborado pelo autor (2025).

O critério de verificação baseia-se no seguinte princípio: se o ponto estiver contido na região, a soma das áreas desses quatro triângulos será igual à área total do quadrilátero. Em contrapartida, se o ponto estiver externo à região, essa soma será diferente da área do quadrilátero, conforme ilustrado na Figura 31.

Figura 31 – Qualquer uma das áreas calculadas nos pontos dos exemplos a), b), c) e d) ilustrados à direita deve ser igual à área total do quadrilátero calculado à esquerda.



Fonte: Elaborado pelo autor (2025).

A principal vantagem desse método de verificação reside em sua versatilidade, pois independentemente das dimensões ou da forma do quadrilátero, a verificação sempre será possível mediante a comparação das áreas calculadas.

A implementação desse método, baseia-se no cálculo da fórmula da distância euclidiana entre dois pontos, expressa pela Equação (3).

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3}$$

Dessa forma, é possível calcular a área total da região quadrilátera ao traçar duas retas diagonais que conectam os vértices opostos da região. O produto dessas distâncias dividido por dois fornece a área total (Equação (4)).

$$A = \frac{d * D}{2} \tag{4}$$

Em seguida, para calcular a área de cada um dos quatro triângulos formados, utiliza-se a fórmula de Heron, que permite calcular a área a partir apenas do conhecimento do comprimento dos três lados do triângulo. A fórmula é expressa por Equação (5).

$$A = \sqrt{s(s-a)(s-b)(s-c)} \tag{5}$$

Na Equação 5, *a*, *b* e *c* representam os lados do triângulo e *s* denota o semi-perímetro, calculado pela média aritmética da soma dos três lados, conforme a Equação 6.

$$s = \frac{a + b + c}{2} \tag{6}$$

Por fim, a Figura 32 ilustra a aplicação da proposta de verificação de detecção do veículo na região no cenário de uma via com um veículo.

Figura 32 – Exemplo visual da verificação do ponto projetado em duas situações diferentes: em (a) o veículo está sobre a região, e em (b) o veículo está fora da região.

(a) Região na mesma pista do veículo (b) Região em pista diferente do veículo

Fonte: Elaborado pelo autor (2025).

Portanto, o método de análise da trajetória dos veículos proposto permite realizar dois tipos de contagem: (i) o total de veículos que passaram por cada região, denominado método região; e (ii) os movimentos realizados entre regiões, definidos pela primeira e última região percorrida pelo veículo, denominado método movimento.

3.3.2 Algoritmo de Contagem Desenvolvido

Para a implementação dessa metodologia de contagem proposta, foi desenvolvido um algoritmo que utiliza as bibliotecas apresentadas no Tabela 3, que complementam as dependências já descritas na Seção 3.2.

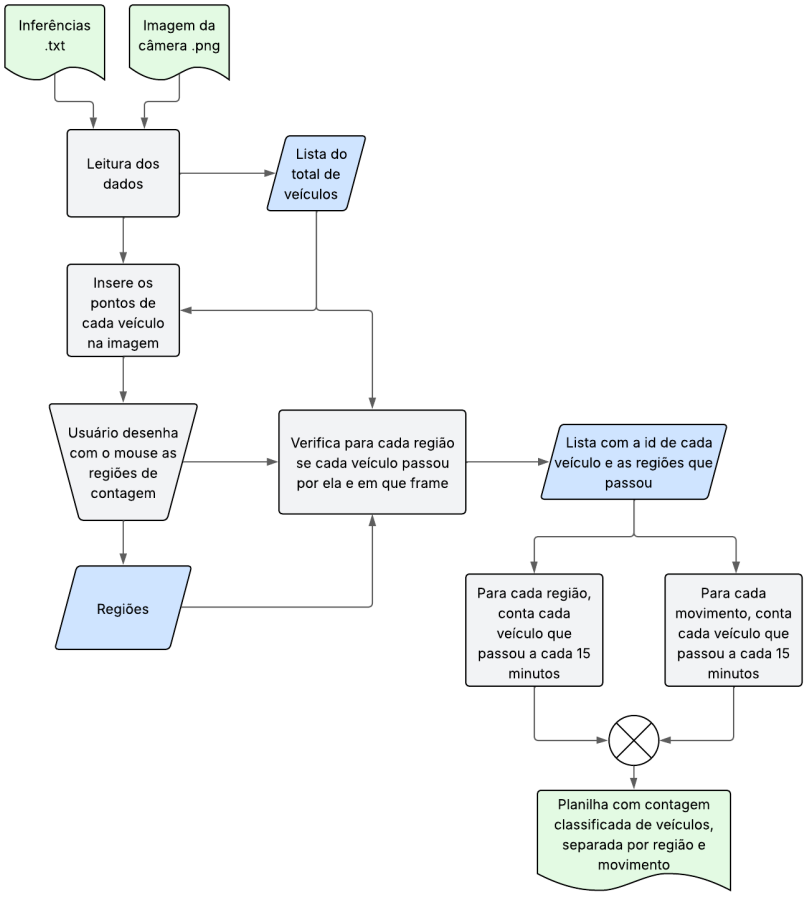
Tabela 3 – Bibliotecas utilizadas no algoritmo de contagem e suas versões.

Biblioteca	**Versão**
Python	$\geq$ 3.8
opencv-python	$\geq$ 4.6.0
XlsxWriter	$\geq$ 3.1.2
openpyxl	$\geq$ 3.1.2

Fonte: Elaborado pelo autor (2025).

A Figura 33 apresenta O diagrama geral do funcionamento desse código, ilustrando o fluxo de processamento desde a entrada de dados até a geração dos resultados de contagem.

Figura 33 – Diagrama geral do código de contagem. Em verde são documentos externos e em azul são dados relevantes gerados.



Fonte: Elaborado pelo autor (2025).

O algoritmo requer a configuração de parâmetros de entrada que definem tanto os arquivos a serem processados quanto as características temporais e espaciais

do vídeo analisado. O Quadro 8 apresenta esses parâmetros e suas respectivas descrições.

Quadro 8 – Parâmetros de entrada do algoritmo de contagem.

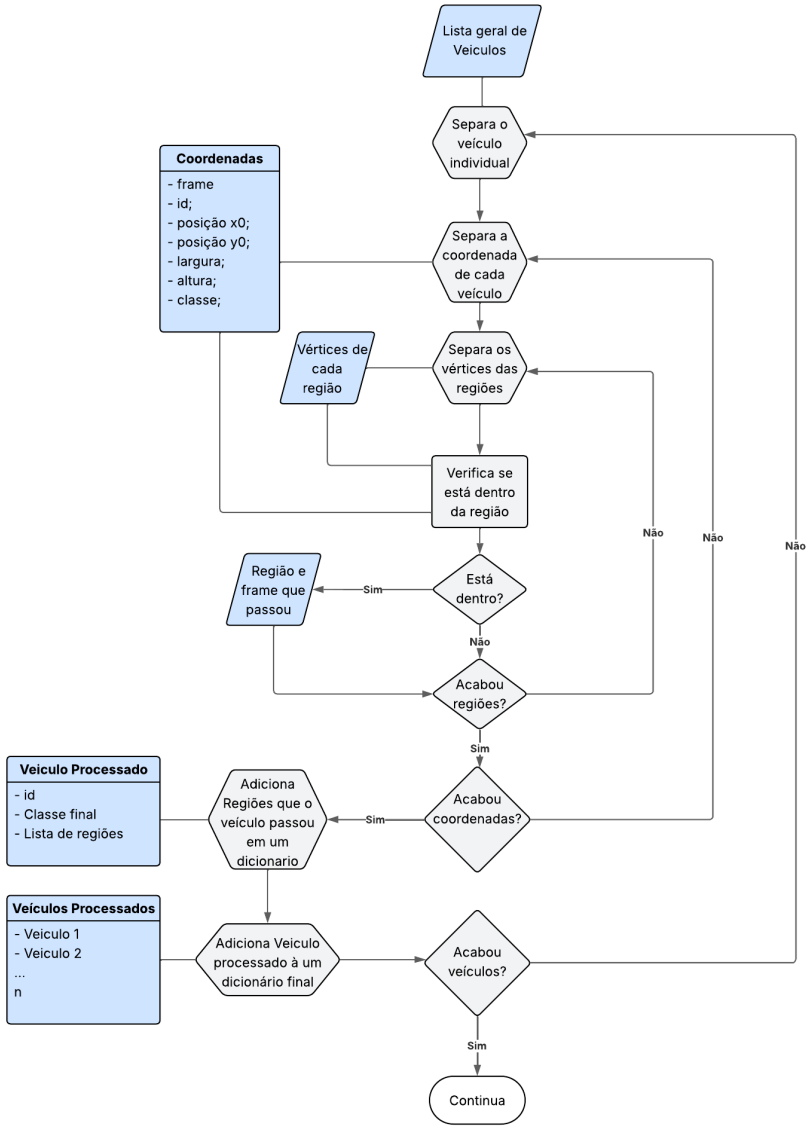
Parâmetro	**Descrição**
`file_name`	Arquivo `.txt` contendo os resultados do rastreamento (classe, ID, posição e confiança) gerados na etapa anterior
`img_name`	Imagem de referência (`.jpeg` ou `.png`) extraída do vídeo processado, representando a visada da câmera
`frames`	Número total de *frames* do vídeo processado
`tempo`	Duração total do vídeo em horas
`Resolucao`	Tupla contendo a resolução do vídeo no formato (largura, altura) em *pixels*

Fonte: Elaborado pelo autor (2025).

A partir desses parâmetros de entrada, o algoritmo organiza as informações, agrupando as coordenadas de cada objeto com base em seu identificador único (ID). Em seguida, é exibida uma janela com a trajetória de todos os veículos, permitindo que o usuário defina manualmente regiões de interesse. Essas regiões são representadas como quadriláteros, simbolizando origens e destinos possíveis na cena. Os vértices podem ser posicionados em qualquer lugar, podendo formar qualquer tamanho e angulação diferente entre as linhas.

Com as coordenadas dos veículos e as coordenadas das regiões definidas, o próximo passo do algoritmo é realizar a verificação da detecção do veículo sobre a região desenhada. Isso foi feito utilizando *loops* que iteram sobre cada coordenada de cada veículo para cada região, armazenando qual região cada veículo atravessou e o respectivo *frame* em que isso ocorreu. Durante esses *loops*, o algoritmo também registra todas as classificações atribuídas ao veículo ao longo do rastreamento, calculando ao final a moda da classe que mais se repetiu no período. A Figura 34 ilustra a lógica dessa verificação:

Figura 34 – Diagrama do processo de verificação de passagem por regiões.



Fonte: Elaborado pelo autor (2025).

A etapa final do algoritmo consiste em realizar os dois tipos de contagem propostos anteriormente: contagem pelo método região e contagem pelo método movimento. Ambos os métodos organizam os resultados em intervalos de 15 minutos. Para possibilitar essa divisão temporal, o algoritmo calcula a taxa de *frames* por minuto (FPM) com base nos parâmetros de *frames* e de tempo de vídeo fornecidos como parâmetro de entrada, conforme mostra a equação (7).

$$FPM = \frac{frames}{tempo \times 60} \tag{7}$$

Com esse valor, o algoritmo define os intervalos temporais em múltiplos de 15 minutos, convertendo-os em valores de *frame*. Assim, se o último *frame* em que o

veículo passou pela região estiver entre dois desses valores, ele é contabilizado no respectivo intervalo.

Na montagem do *dataset* (Subseção 3.1.1), foram utilizadas dez classes diferentes para o treinamento do modelo. No entanto, na etapa de contagem, os veículos são consolidados em apenas quatro categorias: motocicleta, automóvel, ônibus e caminhão. Dessa forma, as classes *car*, caminhonete, *van*, Kombi, minivan e carroforte são tratadas como automóvel pelo algoritmo de contagem.

Por fim, os dados finais da contagem são organizados em uma planilha, contendo abas separadas para cada região e para todos os possíveis movimentos entre as regiões definidas.

3.4 Métricas e Cenários Considerados Para Avaliação

Foram selecionados dois cenários distintos de filmagens reais, ambos já submetidos à contagem manual humana. O primeiro cenário refere-se ao posto P008, que compreende duas vias separadas de sentido único (Figura 35.a). Esse posto pode ser analisado exclusivamente com o método de contagem por região, descrito na Seção 3.3, sendo também um caso interessante para comparar os resultados com o método por movimento, já que ambos permitem contabilizar o mesmo volume de veículos por abordagens distintas.

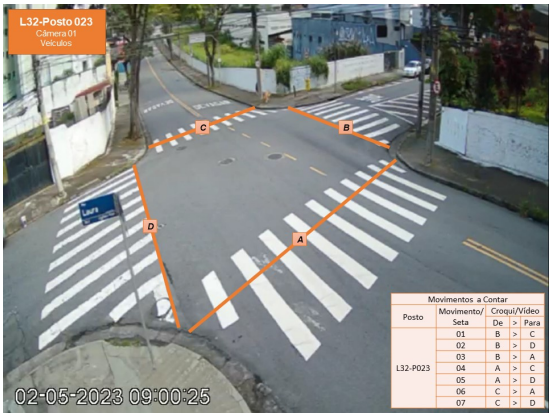
O segundo cenário corresponde ao posto P023, que representa um cruzamento com sete movimentos possíveis (Figura 35.b). Esse ambiente é apropriado para avaliar, principalmente, a capacidade do sistema em rastrear e contar múltiplos movimentos de conversão simultâneos, o que requer o uso do método por movimento, também descrito na Seção 3.3.

A avaliação de desempenho foi conduzida a partir da comparação entre os resultados obtidos pelo sistema e a contagem manual de referência, considerando as categorias de motocicleta, automóvel, ônibus e caminhão, ao longo de um período contínuo de 24 horas, segmentado em intervalos de 15 minutos (quartis).

Para a empresa Consultran Engenharia, o critério adotado na validação de contadores humanos estabelece como aceitável uma diferença de até 10% em relação à contagem de referência (classificação verde); diferenças entre 11% e 15% são consideradas em estado de alerta (classificação amarela); e valores superiores a 16% são classificados como inaceitáveis (vermelho). Dessa forma, os resultados de contagem obtidos neste projeto foram avaliados como se o sistema proposto fosse um contador humano contratado pela Consultran, devendo, portanto, atender aos mesmos parâmetros de validação com base na contagem de referência, neste caso, realizada por um operador previamente contratado pela empresa.

Figura 35 – Croqui de movimento dos dois tipos de cenários a serem avaliados.

(a) Filmagem Posto 008



(b) Filmagem Posto 023

Fonte: Elaborado pelo autor (2025).

4 RESULTADOS OBTIDOS E DISCUSSÃO

Nesta seção, serão apresentados e discutidos os resultados obtidos a partir da metodologia proposta, com foco nos três componentes principais do sistema: detecção, rastreamento e contagem. A adoção dessa estrutura segmentada de análise decorre da própria organização modular descrita no Capítulo 3, uma vez que o desempenho global da solução depende do funcionamento adequado de cada módulo, tanto individualmente quanto de forma integrada.

A interdependência entre os componentes torna relevante sua avaliação separada: se a etapa de detecção e classificação não apresentar resultados satisfatórios, o módulo de rastreamento terá dificuldades para manter a identidade dos objetos ao longo dos quadros, comprometendo diretamente a contagem. Por outro lado, ainda que a detecção funcione corretamente, um rastreador ineficaz impossibilitará o acompanhamento consistente dos veículos, o que inviabiliza qualquer tentativa de contagem precisa.

A etapa de contagem, por sua vez, é aquela que reflete, na prática, a efetividade combinada das etapas anteriores, uma vez que nela são aplicados filtros responsáveis por consolidar as informações provenientes da detecção e do rastreamento. Um exemplo é o uso da moda das classes atribuídas a um mesmo objeto rastreado, estratégia que reduz erros de classificação durante o processo de contagem, mesmo que em determinados *frames* o modelo apresente uma classificação incorreta para aquele veículo.

Dessa maneira, a análise da etapa de Detecção será focada na qualidade do treinamento e nas métricas quantitativas associadas ao modelo gerado. A avaliação do Rastreamento envolverá a verificação visual de trajetórias e a observação visual das detecções e classificações ao longo do vídeo, além do desempenho de processamento. Por fim, a etapa de Contagem será avaliada a partir do processamento e contagem de vídeos de 24 horas, comparando seus resultados com os obtidos por contagem manual realizada por um operador humano.

4.1 Resultados do Treinamento

Os recursos provisionados pelo Google Colab no momento do treinamento incluíam uma GPU T4 com 16 GB de memória, que processou cada época de treinamento em aproximadamente 18 segundos, totalizando cerca de 1 hora e 30 minutos de execução. Ao término do processo, o algoritmo gerou automaticamente uma série de imagens contendo métricas de desempenho do melhor modelo obtido, bem como da evolução do treinamento ao longo das épocas.

A quantidade total de instâncias (objetos rotulados) utilizadas para o treinamento é apresentada na Tabela 4. Como as imagens do *dataset* foram retiradas de visadas completas das câmeras, a proporção de instâncias treinadas refletiu a proporção de veículos que trafegam nas vias urbanas filmadas pela Consultran Engenharia. Observou-se uma grande quantidade de carros (*car*), seguidos por motocicletas (*motorcycle*), caminhões (*truck*) e ônibus (*bus*). As demais classes apresentaram quantidades ainda menores, principalmente por se tratarem de subcategorias de automóveis.

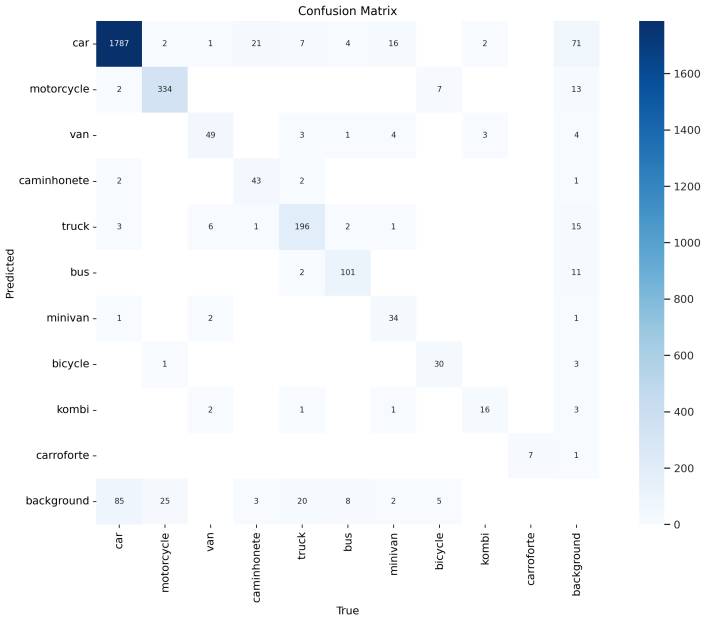
Tabela 4 – Distribuição de instâncias por classe no conjunto de dados

Classe	**Treinamento**	**Validação**	**Total**	**Percentual (%)**
car	7 350	1 856	9 206	66,78
motorcycle	1 460	359	1 819	13,20
truck	755	232	987	7,16
bus	540	113	653	4,74
van	233	58	291	2,11
caminhonete	199	68	267	1,94
minivan	158	57	215	1,56
bicycle	182	32	214	1,55
Kombi	73	20	93	0,67
carroforte	33	7	40	0,29
Total	**10 983**	**2 802**	**13 785**	**100,00**

Fonte: Elaborado pelo autor (2025).

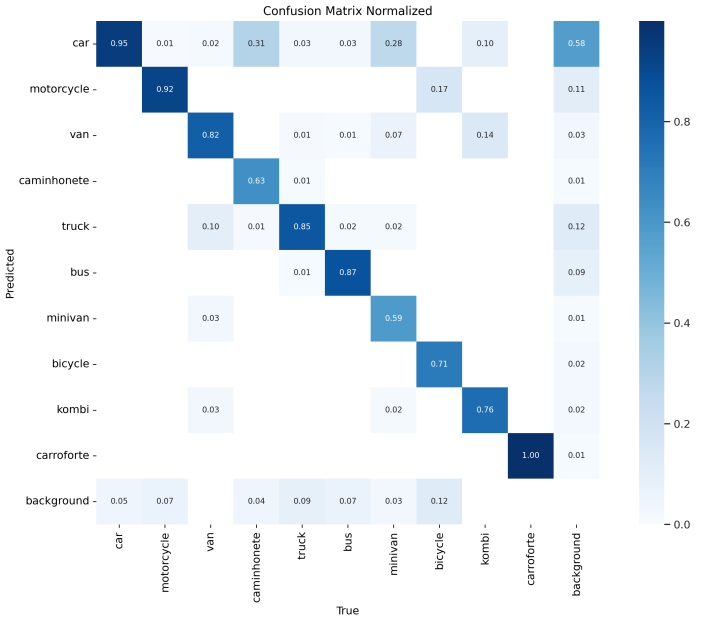
A matriz de confusão gerada a partir do melhor modelo treinado é apresentada na Figura 36. Devido ao desbalanceamento entre as classes, evidenciado pelo número significativamente maior de instâncias da classe *car*, torna-se essencial analisar também a versão normalizada da matriz de confusão (Figura 37), que permite avaliar percentualmente as confusões entre as classes, proporcionando uma análise mais equilibrada dos resultados de classificação.

Figura 36 – Matriz de confusão.



Fonte: Elaborado pelo autor (2025).

Figura 37 – Matriz de confusão normalizada.



Fonte: Elaborado pelo autor (2025).

Primeiramente, destacam-se os resultados positivos das inferências de verdadeiros positivos, em que mais de 93% das instâncias das classes *car*, *motorcycle* e carroforte foram detectadas e classificadas corretamente. Também foram observados altos índices de verdadeiros positivos para as classes *truck* e *bus*, alcançando mais de 85% de acerto. Quanto às demais classes, verificou-se um percentual elevado de

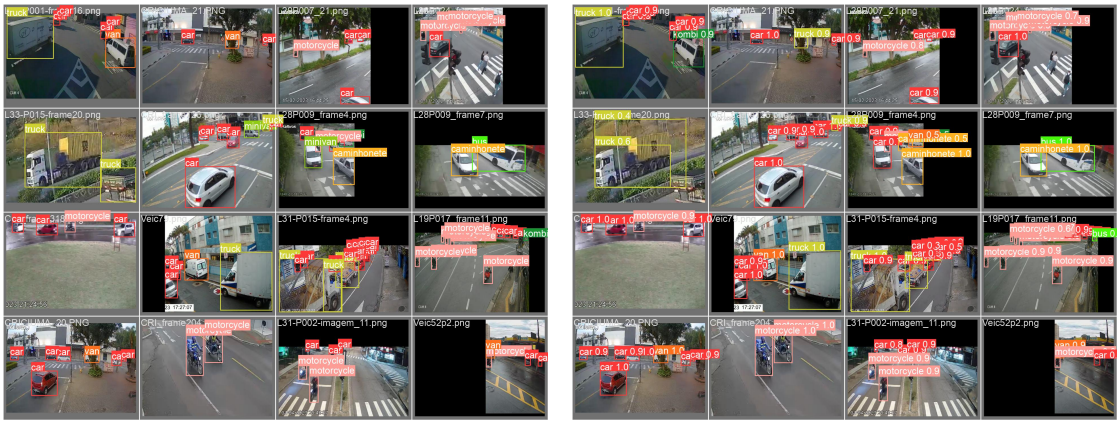
falsos positivos da classe *car* para objetos rotulados como caminhonete e minivan, atingindo 31% e 28%, respectivamente. Nota-se que, exceto pela classe van (com 82% de acerto), as demais subcategorias de automóveis apresentaram uma taxa de acerto insatisfatória, com ocorrência significativa de falsos positivos em outras classes, indicando uma dificuldade do modelo em distinguir corretamente essas categorias.

No entanto, essas dificuldades de classificação não representam, inicialmente, um problema para este trabalho, pois as classes *car*, van, caminhonete, minivan, Kombi e carroforte são tratadas como subcategorias de automóvel, conforme explicado na Seção 3.1.1. Portanto, qualquer confusão entre essas classes não afeta o valor da contagem, uma vez que pouco importa se passou, de fato, um *car* ou uma caminhonete, pois ao final ambos são contabilizados como automóvel.

Quanto ao alto percentual (58%) de falsos positivos da classe *car* em *background*, esse valor não se traduz, numericamente, em uma quantidade elevada, considerando que são apenas 71 falsos positivos em um universo de 1880 instâncias rotuladas da classe *car*. Dessa forma, a matriz de confusão apresentou resultados preliminares bastante satisfatórios, nas quais todas as classes (com exceção de *bicycle*) apresentaram menos de 10% de falsos negativos. Ou seja, a maior parte dos veículos presentes no *dataset* de validação foi corretamente localizada.

Uma forma de visualizar, na prática, o comportamento evidenciado pela matriz de confusão é por meio da análise dos pares de lote de validação, que são amostras de imagens do subconjunto de validação do *dataset*, acompanhadas dos respectivos *labels* e das *predictions* realizadas pelo modelo. Destaca-se o comportamento de *data augmentation* do algoritmo, que não é aplicado ao subconjunto de validação, como pode ser observado na Figura 38 a seguir.

Figura 38 – *Labels* manuais e predições do modelo durante validação.



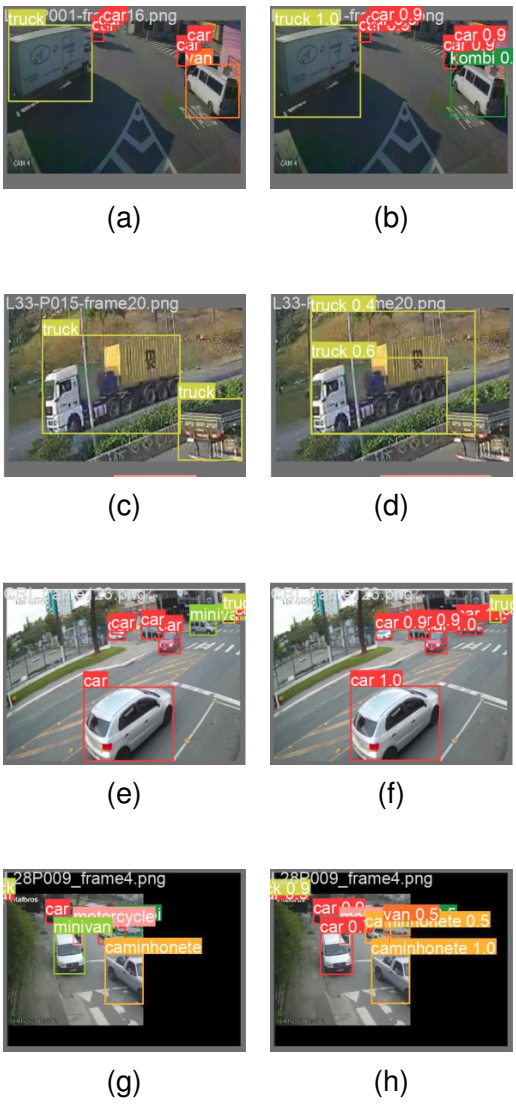
(a) *Labels* de algumas imagens do dataset de validação.

(b) Predição do modelo em algumas imagens do dataset de validação.

Fonte: Elaborado pelo autor (2025).

A seguir, na Figura 39 destacam-se alguns pares de imagens individuais, com o objetivo de evidenciar determinados comportamentos observados durante a validação.

Figura 39 – *Labels* manuais e predições do modelo durante validação. Imagens apresentadas em pares ((a) e (b), (c) e (d), ...): à esquerda os *labels* manuais; à direita as predições do modelo.



(a) (b)

(c) (d)

(e) (f)

(g) (h)

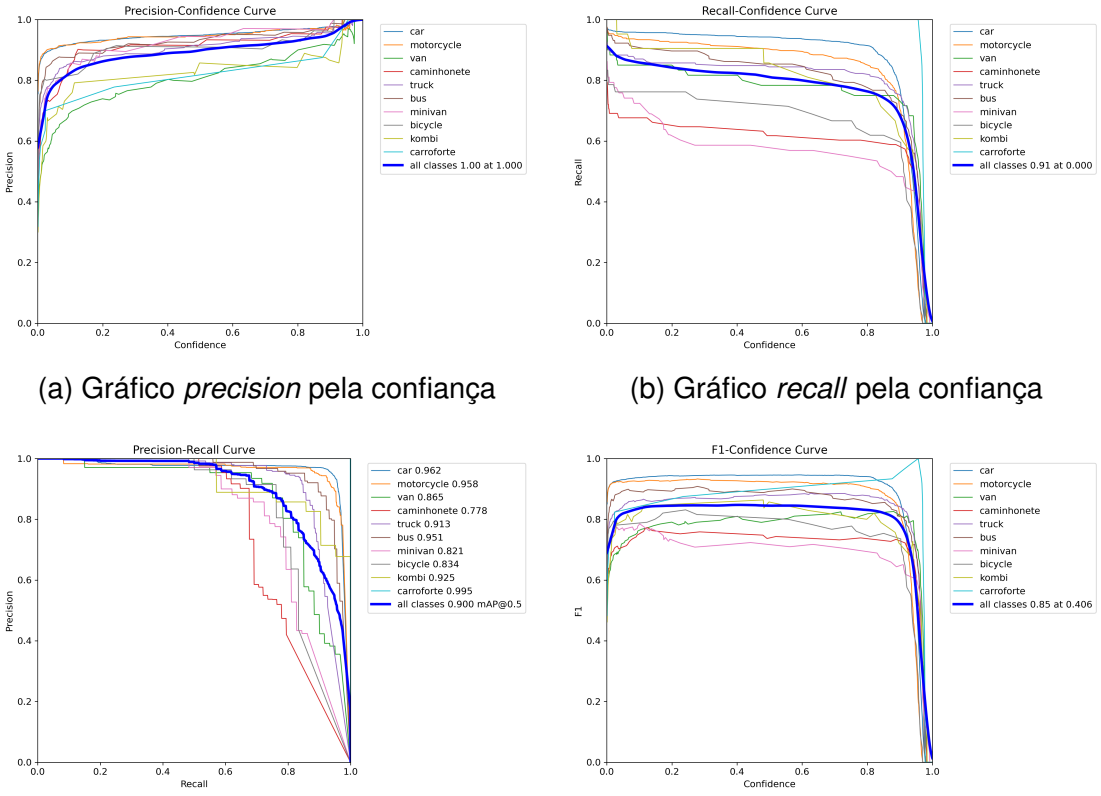
Fonte: Elaborado pelo autor (2025).

O primeiro par de imagens (Figuras 39.a e 39.b) ilustra uma confusão entre as classes van e Kombi. O segundo par (Figuras 39.c e 39.d) evidencia um erro de detecção relacionado à carreta de um caminhão: em vez de reconhecer o veículo articulado e o caminhão da imagem, o modelo identificou dois caminhões distintos no mesmo caminhão. O terceiro par (Figuras 39.e e 39.f) apresenta uma confusão entre minivan e *car*. Por fim, o quarto par (Figuras 39.g e 39.h) mostra dois problemas simultâneos: uma nova confusão entre minivan e *car*, além de uma detecção correta como caminhonete de um veículo que, na verdade, havia sido rotulado erroneamente

como *car* durante a preparação do *dataset*.

Uma análise mais profunda pode ser realizada por meio dos gráficos das curvas de *precision*, *recall*, *precision-recall* e *F1*, apresentados na Figura 40 a seguir.

Figura 40 – Gráficos das principais métricas do melhor modelo gerado.



(a) Gráfico *precision* pela confiança

(b) Gráfico *recall* pela confiança

(c) Gráfico *precision* pelo *recall*

(d) Gráfico *score F1* pela confiança

Fonte: Elaborado pelo autor (2025).

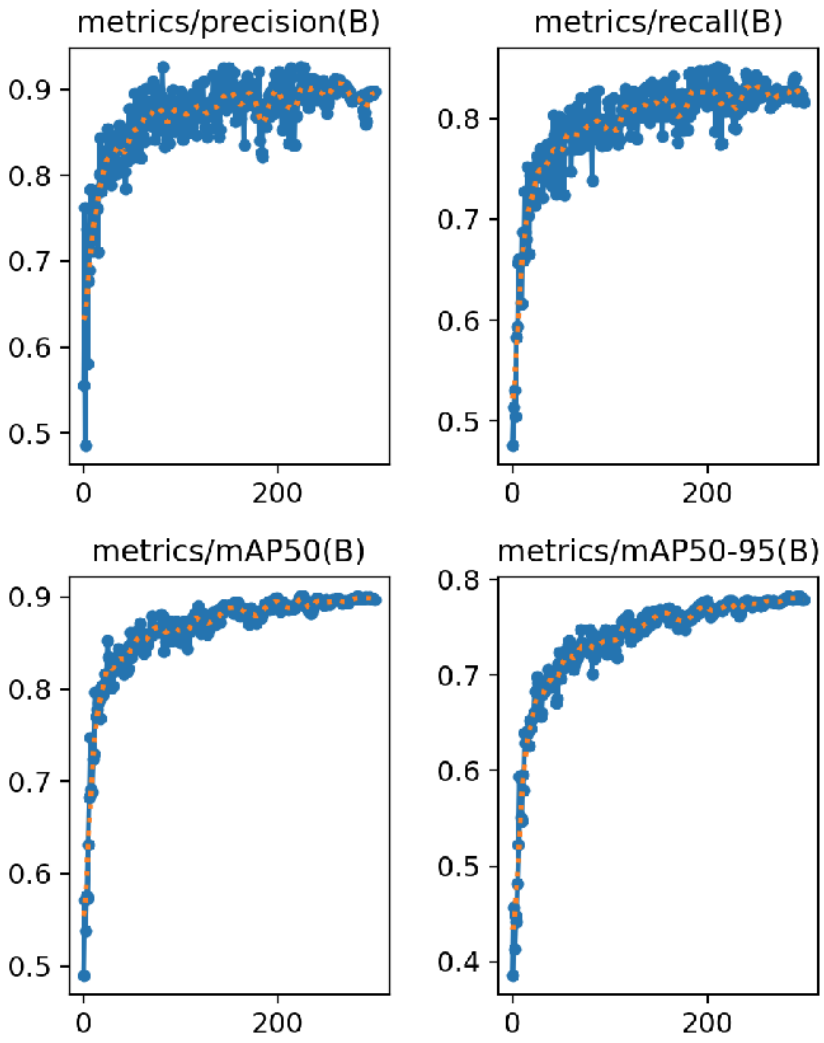
De forma geral, as curvas de *precision* e *recall* apresentam o comportamento esperado, com poucos falsos positivos e falsos negativos mesmo para níveis baixos de confiança, ou seja, pontuações maiores que 0,8 em ambas as métricas, mesmo com apenas 20% de confiança na média de todas as classes. No entanto, algumas classes apresentam desempenho muito abaixo da média geral, como van e Kombi na curva de *precision*, e minivan e caminhonete na curva de *recall*.

A Figura 40.c mostra a relação entre *precision* e *recall*, considerando o *mAP50*, em que foi atingido um *score* de aproximadamente 0,9. A Figura 40.d, que utiliza a métrica de *F1 score*, também confirma um bom desempenho para a média de todas as classes, com valor próximo de 0,85 para uma confiança de apenas 40,6%. Por ser uma pontuação única que relaciona *precision* e *recall*, percebe-se que as mesmas classes minivan e caminhonete também estão significativamente abaixo do desempenho. Como essas são as mesmas classes que performaram mal em *recall* e bem em *precision*, significa que o modelo teve dificuldades em classificar essas categorias. A principal

causa dessa limitação é a confusão com a classe *car*, evidenciada pela matriz de confusão, onde veículos dessas categorias foram incorretamente classificados como *car*.

Na Figura 41 é possível visualizar o comportamento das métricas de *precision* e *recall*, incluindo a métrica *mAP50-95*, responsável por definir o melhor modelo durante o treinamento.

Figura 41 – Precisão ao longo das épocas de treinamento



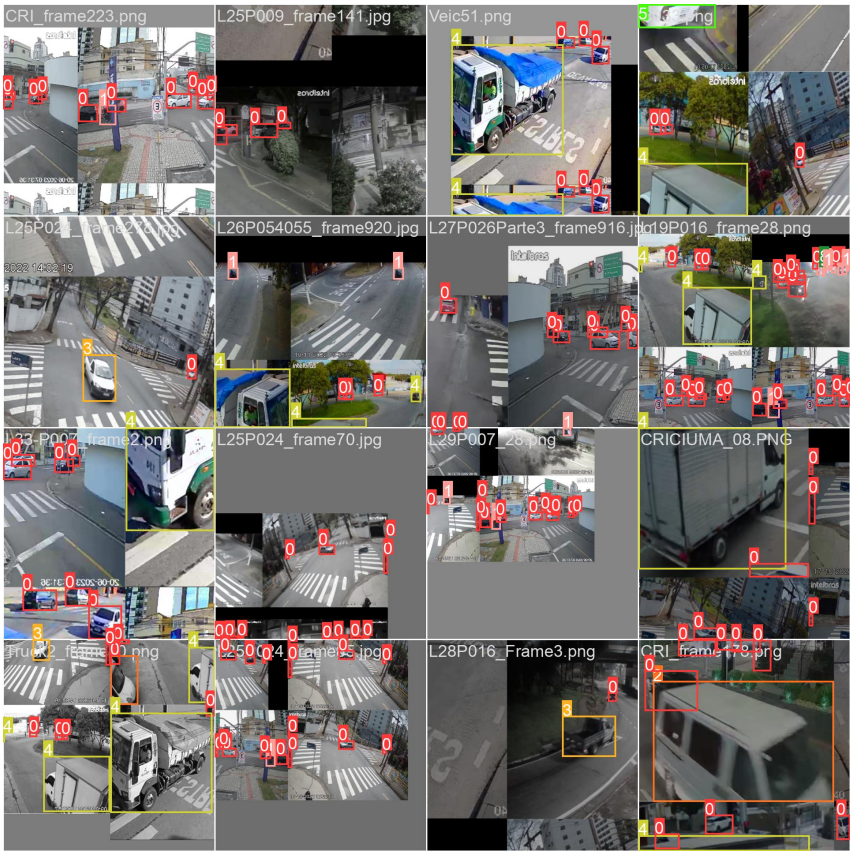
Fonte: Elaborado pelo autor (2025).

Percebe-se que o desempenho do modelo aumentou consideravelmente até a época número 200, quando sua melhora passou a ser gradativa. A partir dos dados contidos no arquivo chamado *results*, verificou-se que o melhor modelo, segundo a métrica *mAP50-95*, foi gerado na época 283.

Por fim, cabe mencionar a Figura 42, uma imagem interessante gerada pelo algoritmo, que exemplifica visualmente o funcionamento de um *batch* de imagens de treinamento recebido como parâmetro pelo modelo. Nessa imagem, é possível

compreender melhor como ocorre o *data augmentation* do tipo mosaico, configurado como padrão pelo pacote da Ultralytics (2023). Como foi definido o valor 16 para o parâmetro *batchsize*, a Figura 42 apresenta 16 imagens, sendo cada uma delas um “segmento” de uma ou mais imagens do *dataset* original. Nota-se que, quando a imagem não cobre o quadro completo de 640 × 480 *pixels*, o restante é preenchido com *pixels* cinzas, de modo similar ao método de máscara utilizado na montagem do *dataset*, em que regiões indesejadas da imagem foram ocultadas por preenchimento com *pixels* pretos.

Figura 42 – Imagens treinadas com *data augmentation* feito automaticamente pelo algoritmo da Ultralytics.



Fonte: Elaborado pelo autor (2025).

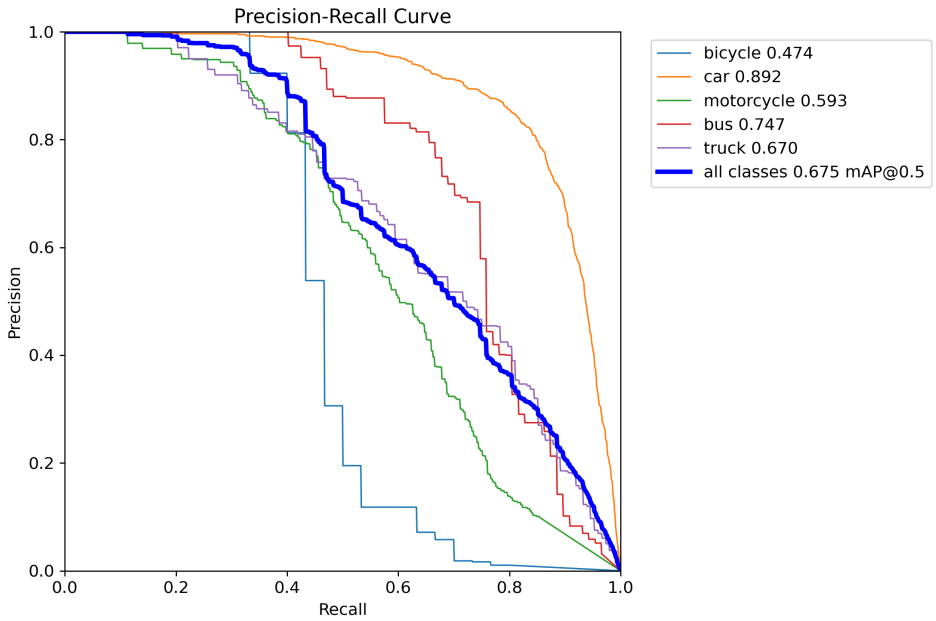
4.1.1 Comparação com Modelo Pré-Treinado

Para avaliar o desempenho do modelo customizado em relação ao modelo YOLOv8m pré-treinado, foi realizado uma adaptação do conjunto de dados de validação. Os rótulos do *dataset* customizado, originalmente estruturados em 10 classes (índices 0 a 9), foram mapeados para o formato COCO. Basicamente, uma função em Python muda os índices das subcategorias de automovel (van, caminhonete, minivan, Kombi e carroforte) para a classe *car* do COCO. Assim, todas as classes rotuladas no *dataset*

customizado foram mapeadas para os índices COCO originais: *bicycle* (1), *car* (2), *motorcycle* (3), *bus* (5) e *truck* (7).

Após a conversão dos rótulos e a preparação do arquivo de configuração, ambos os modelos foram submetidos ao processo de validação através do método `val()` da biblioteca Ultralytics, utilizando os mesmos parâmetros: tamanho de imagem de 640 *pixels*, *threshold* de confiança de 0,001 e IoU de 0,5. Esse método gera automaticamente as curvas de *Precision-Recall* e demais métricas de desempenho, permitindo uma comparação direta e objetiva entre os modelos, conforme apresentado na Figura 43.

Figura 43 – Curva *Precision-Recall* do modelo YOLOv8m pré-treinado.



Fonte: Elaborado pelo autor (2025).

Os resultados demonstram que o modelo customizado apresentou desempenho significativamente superior ao modelo pré-treinado, com uma diferença de 0,225 na métrica *mAP50* de forma geral. Essa diferença expressiva pode ser atribuída à inserção das subcategorias no modelo customizado, o que permitiu a classificação adequada de cada silhueta comumente encontrada em trânsitos brasileiros.

Por outro lado, destaca-se o desempenho ainda mais baixo nas classes *bicycle* e *motorcycle* no modelo pré-treinado. É importante notar que essas classes não apresentam "silhuetas de confusão", ao contrário do que ocorre com *car* e *truck*, que podem ser confundidas com veículos como caminhonete, Kombi e outros. Ou seja, o principal fator responsável pelo desempenho insatisfatório na detecção de bicicletas e motocicletas não pode ser atribuído a confusões entre classes similares. Nesse contexto, o treinamento do modelo customizado com imagens em condições e caracte-

rísticas mais similares às do cenário de aplicação (descritas na Seção 2.1.1) contribuiu significativamente para a melhoria de desempenho. O modelo customizado alcançou ganhos superiores a 0,36 e 0,445 nas classes bicicleta e motocicleta, respectivamente, evidenciando a importância da especialização do conjunto de treinamento.

Esses resultados comprovam a observação empírica realizada na Subseção 3.1.2, na qual foi constatado que o modelo pré-treinado não seria suficiente para atingir os objetivos estabelecidos para este trabalho.

4.2 Resultados do Rastreamento

A partir do melhor modelo gerado na Seção 4.1, foi possível aplicar o algoritmo da Seção 3.2 com a seguinte linha de comando:

Quadro 9 – Comando para execução do algoritmo de rastreamento.

Comando de execução no terminal
`python track.py -tracking-method bytetrack -yolo-model weights/best.pt -img 640 -conf 0.7 -source videos/L34-P010.mp4 -show -save-txt`

Fonte: Elaborado pelo autor (2025).

O tempo de processamento de cada *frame* foi aproximadamente 60 ms. Ou seja, para rodar completamente as 4 partes dos vídeos de aproximadamente 216 000 *frames*, que compõe 24 horas de filmagem por posto, levou cerca de 14 horas e 24 minutos.

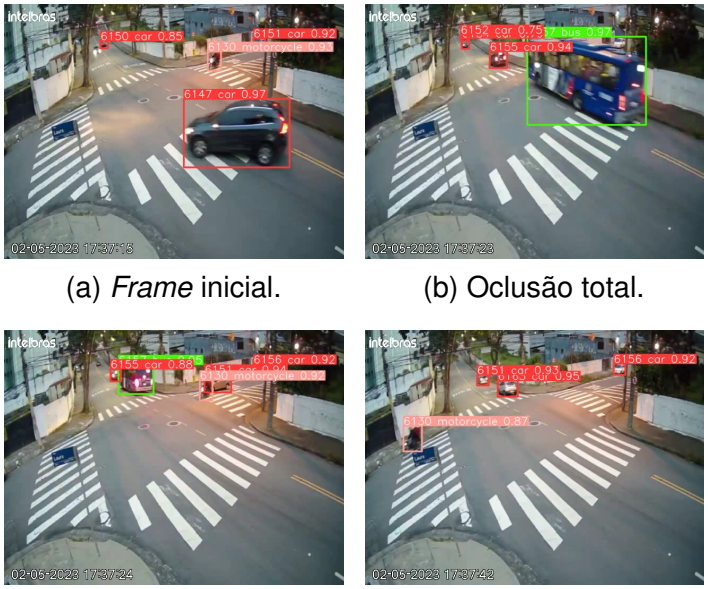
Análises visuais preliminares mostram o comportamento esperado para o rastreamento dos veículos, como no exemplo a seguir em que mesmo após a oclusão parcial do motociclista pela placa, a identificação permaneceu igual (figura 44). No exemplo seguinte, uma motocicleta permaneceu sendo rastreada com o mesmo ID por aproximadamente 30 segundos, onde houve oclusões totais durante alguns *frames*, causadas por um ônibus (figura 45).

Figura 44 – Rastreamento motocicleta com oclusão parcial.

(a) *Frame* inicial. (b) Oclusão parcial. (c) *Frame* final.

Fonte: Elaborado pelo autor (2025).

Figura 45 – Rastreamento m com oclusão total.



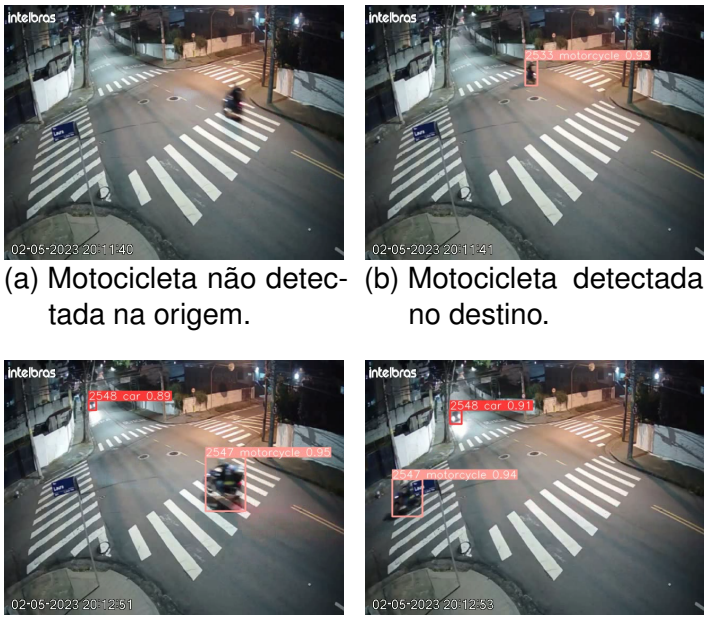
(a) *Frame* inicial. (b) Oclusão total.

(c) Reaparecimento. (d) Destino final.

Fonte: Elaborado pelo autor (2025).

Durante a noite, foi possível perceber o mesmo bom comportamento de rastreamento para todos os veículos, exceto a motocicleta. Movimentos de curva tiveram um bom rastreamento, porém em movimentos em linha reta não foi possível detectar a origem da motocicleta.

Figura 46 – Rastreamento motocicleta em dois movimentos distintos.



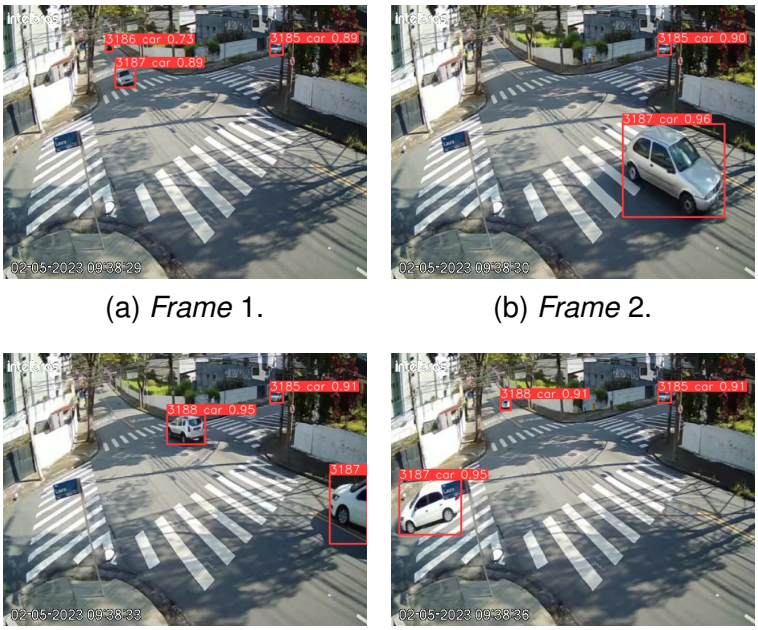
(a) Motocicleta não detectada na origem. (b) Motocicleta detectada no destino.

(c) Motocicleta detectada na origem. (d) Moto rastreada no destino.

Fonte: Elaborado pelo autor (2025).

Outro comportamento negativo do rastreamento, é que dependendo do ângulo da câmera e do objeto, pode ocorrer reidentificações (re-IDs) entre veículos que sairam da cena e veículos que surgiram. Por exemplo o caso a seguir da figura 47, em que um carro que seguiu reto, foi reidentificado como o veículo que surgiu na direção oposta, que após isso, realizou uma conversão à sua esquerda. Esse comportamento pode afetar a contagem por movimento, ocorrendo um falso positivo de contagem, como se o veículo daquela origem, tivesse finalizado seu movimento na cena em um destino diferente do que de fato aconteceu.

Figura 47 – Caso de re-ID entre dois veículos diferentes (Id 3187).



(a) *Frame* 1. (b) *Frame* 2.

(c) *Frame* 3. (d) *Frame* 4.

Fonte: Elaborado pelo autor (2025).

4.3 Resultados da Contagem

Nesta seção, são apresentados e discutidos os resultados obtidos a partir da contagem de veículos realizada pelo método proposto neste trabalho de conclusão de curso. O algoritmo desenvolvido recebe como entrada os dados de rastreamento e classificação gerados durante o processo de inferência e aplica os procedimentos de contagem definidos nesta pesquisa. A análise contempla os diferentes métodos de contagem implementados, bem como as categorias de veículos previamente estabelecidas. Os resultados são avaliados com base nas métricas descritas na Seção 3.4, permitindo uma comparação direta entre os métodos propostos e a referência humana.

Para fins de representação, o algoritmo é referido ao longo desta seção apenas como IA ou YOLO, visto que sintetiza o desempenho agregado do modelo YOLOv8 treinado em conjunto com o mecanismo de rastreamento. Dessa forma, nas legendas

das figuras a seguir, as contagens realizadas pelo algoritmo serão identificadas por esses nomes, enquanto as contagens manuais serão referidas como operador humano ou contador.

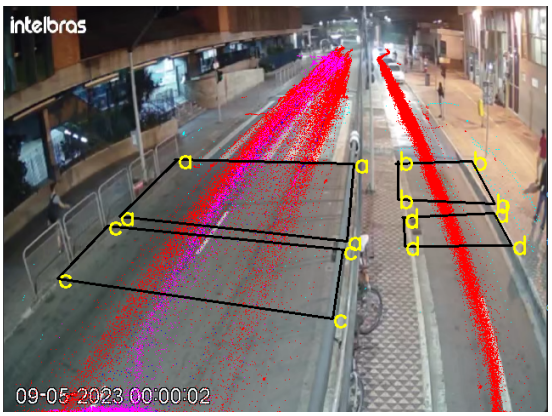
Para facilitar a análise do comportamento ao longo do tempo, foi elaborado um gráfico para cada região e método de contagem. Nele, o contorno preto das barras representa a contagem realizada pelo operador humano, enquanto o preenchimento corresponde à contagem da IA (identificada como YOLO) no gráfico). A cor da barra indica a diferença percentual entre as duas contagens (verde, amarelo e vermelho, de acordo com a Seção 3.4).

Além disso, estão sobrepostos aos valores por hora os dados correspondentes aos intervalos de 15 minutos, representando os quatro quartis que compõem cada hora. Isso permite visualizar não apenas o total por hora, mas também a distribuição intra-horária das contagens.

4.3.1 Resultados Posto P008

As regiões desenhadas para contagem do posto P008 estão apresentadas na Figura 48.

Figura 48 – Regiões desenhadas e croqui de movimentos do Posto 008.



(a) Regiões desenhadas Posto 008.

(b) Croqui Posto 008.

Fonte: Elaborado pelo autor (2025).

Como mencionado na seção 3.4, os 2 movimentos de cada via foram contados utilizando os dois métodos região e movimento. A Figura 49 a seguir apresenta o resumo da contagem obtida pela "IA". Os resultados estão dispostos em conjuntos de 6 em 6 horas e com o total, para cada movimento possível, comparada com a contagem do operador humano (chamado de contador).

Figura 49 – Resumo da contagem do posto 008.

Resumo Resultado Movimento 1 (Todos em A):				
Tipo:	Moto	Automovel	Onibus	Caminhao
0:00 até 6:00				
IA	74	428	171	25
Contador	68	428	171	26
Dif %	8%	0%	0%	4%
6:00 até 12:00				
IA	572	4211	620	227
Contador	569	4179	613	204
Dif %	1%	1%	1%	10%
12:00 até 18:00				
IA	819	4341	594	220
Contador	796	4369	588	239
Dif %	3%	1%	1%	8%
18:00 até 0:00				
IA	556	3370	541	33
Contador	528	3279	537	40
Dif %	5%	3%	1%	18%
Total				
IA	2021	12350	1926	505
Humano	1961	12255	1909	509
Dif %	3%	1%	1%	1%

(a) Contagem por região.

Resumo Resultado Movimento 2 (Todos em B):				
Tipo:	Moto	Automovel	Onibus	Caminhao
0:00 até 6:00				
IA	10	407	0	0
Contador	11	412	0	0
Dif %	9%	1%	0%	0%
6:00 até 12:00				
IA	31	1602	0	0
Contador	35	1647	0	0
Dif %	11%	3%	0%	0%
12:00 até 18:00				
IA	40	1469	0	0
Contador	40	1475	0	0
Dif %	0%	0%	0%	0%
18:00 até 0:00				
IA	41	1646	0	1
Contador	41	1648	0	1
Dif %	0%	0%	0%	0%
Total				
IA	122	5124	0	1
Humano	127	5182	0	1
Dif %	4%	1%	0%	0%

(b) Contagem por região.

Resumo Resultado Movimento 1 (A para C):				
Tipo:	Moto	Automovel	Onibus	Caminhao
0:00 até 6:00				
IA	65	425	171	25
Contador	68	428	171	26
Dif %	4%	1%	0%	4%
6:00 até 12:00				
IA	526	4192	617	224
Contador	569	4179	613	204
Dif %	8%	0%	1%	9%
12:00 até 18:00				
IA	737	4317	593	219
Contador	796	4369	588	241
Dif %	7%	1%	1%	9%
18:00 até 0:00				
IA	506	3356	540	32
Contador	528	3279	537	40
Dif %	4%	2%	1%	20%
Total				
IA	1834	12290	1921	500
Humano	1961	12255	1909	511
Dif %	6%	0%	1%	2%

(c) Contagem por movimento.

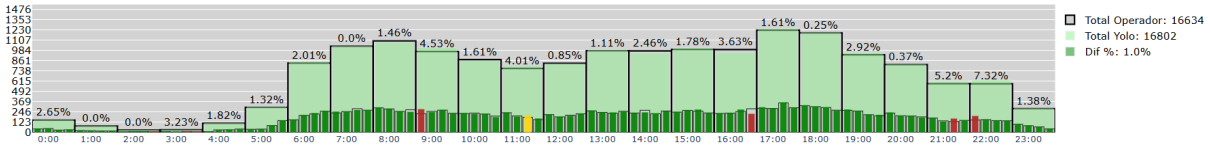
Resumo Resultado Movimento 2 (B para D):				
Tipo:	Moto	Automovel	Onibus	Caminhao
0:00 até 6:00				
IA	10	407	0	0
Contador	11	412	0	0
Dif %	9%	1%	0%	0%
6:00 até 12:00				
IA	27	1602	0	0
Contador	35	1647	0	0
Dif %	23%	3%	0%	0%
12:00 até 18:00				
IA	36	1468	0	0
Contador	40	1475	0	0
Dif %	10%	0%	0%	0%
18:00 até 0:00				
IA	36	1646	0	1
Contador	41	1648	0	1
Dif %	12%	0%	0%	0%
Total				
IA	109	5123	0	1
Humano	127	5182	0	1
Dif %	14%	1%	0%	0%

(d) Contagem por movimento.

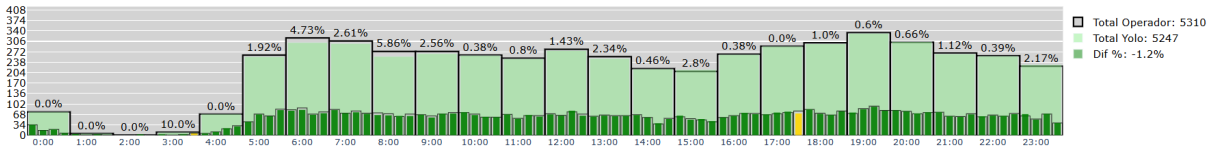
Fonte: Elaborado pelo autor (2025).

A Figura 50 a seguir, apresenta os 4 gráficos da contagem de todos os veículos do posto P008.

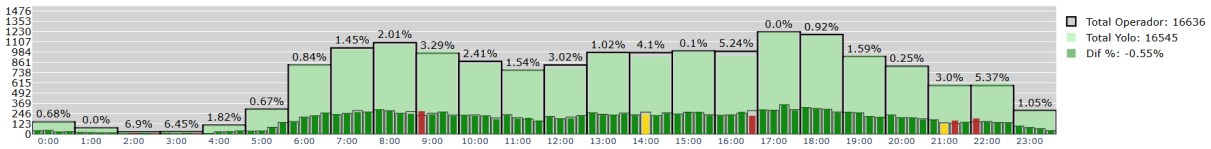
Figura 50 – Gráfico de 24 horas de contagem de todos os veículos do Posto 008 por movimento.



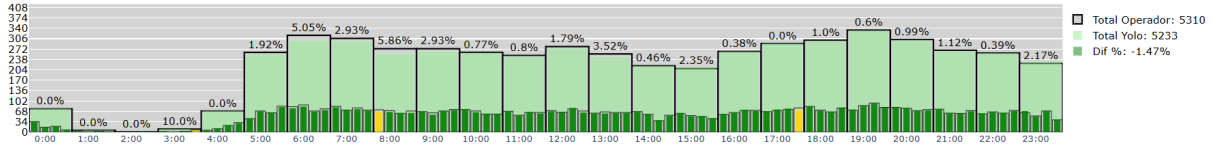
(a) Movimento 1, método região.



(b) Movimento 2, método região.



(c) Movimento 1, método movimento.



(d) Movimento 2, método movimento.

Fonte: Elaborado pelo autor (2025).

Os resultados de contagem obtidos neste posto apresentaram um desempenho muito elevado, alcançando uma acurácia aceitável (indicada em verde) para todos os horários, considerando o total de veículos (sem distinção por classe), em ambos os métodos avaliados.

Ao analisar a contagem por classe, observa-se que as categorias automóvel e ônibus apresentaram excelente desempenho em ambos os métodos, com diferenças inferiores a 1% no total acumulado das 24 horas. Esse resultado corrobora a análise apresentada na Seção 4.1, em que, apesar de haver confusões entre subcategorias de automóvel, não houve impacto relevante na contagem agregada.

Nota-se também que a variação de horário (como períodos diurnos e noturnos) não impactou significativamente os resultados neste posto, visto que não há deterioração perceptível nos diferentes intervalos analisados.

A contagem de caminhões revela um comportamento interessante: no movimento 1, a IA contabilizou 23 caminhões a mais que o operador humano entre 6 h e 12 h, enquanto entre 18 h e 0 h deixou de contabilizar 7 caminhões (utilizando o método por região). Apesar dessas discrepâncias pontuais, as diferenças se compensaram,

resultando em um total diário muito próximo entre os métodos.

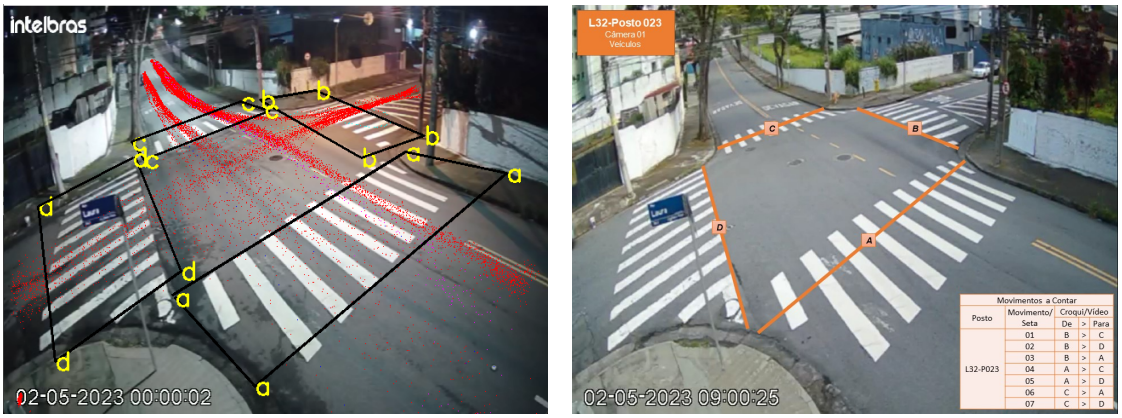
Em relação às motocicletas, observa-se a maior diferença percentual entre os métodos região e movimento. A contagem por região apresentou valores acima aos da contagem humana, enquanto o método por movimento os valores estavam sempre abaixo para essa categoria, embora ambas as diferenças permaneçam em níveis aceitáveis (inferiores a 10% no movimento 1 em todos os conjuntos de horário).

De modo geral, o método por região tende a superestimar as contagens em relação ao operador humano, enquanto o método por movimento tende a subestimar. Essa tendência está diretamente relacionada à dependência da qualidade do rastreamento. Quando o rastreamento falha e a identificação do veículo se perde ao longo da trajetória das regiões de análise, o método por movimento não consegue associar origem e destino da mesma ID, resultando em uma contagem falso-negativa. Por outro lado, no método por região, uma nova identificação atribuída ao mesmo veículo enquanto ele ainda se encontra sobre a região resulta em contagem duplicada, caracterizando uma falsa-positiva.

4.3.2 Resultados Posto P023

As regiões desenhadas para contagem do posto P023 estão apresentadas na Figura 51.

Figura 51 – Regiões desenhadas e croqui de movimentos do Posto 023.



(a) Regiões desenhadas Posto 023. (b) Croqui Posto 023.

Fonte: Elaborado pelo autor (2025).

Diferentemente do Posto P008, discutido na Seção 4.3.1, o Posto P023 foi analisado exclusivamente por meio do método de contagem por movimento, uma vez que todos os seus fluxos são multidirecionais. Não há nenhuma via unidirecional que permita a aplicação do método por região. A Figura 52 apresenta o resumo da contagem realizada pelo algoritmo (IA), comparada com a contagem de referência

do operador humano (contador). Os resultados estão organizados em intervalos de 6 horas, além do total acumulado, para cada um dos movimentos possíveis no posto.

Figura 52 – Resumo da contagem do posto 023.

Resumo Resultado Movimento 1:				
Tipo:	Moto	Automovel	Onibus	Caminhao
0:00 até 6:00				
IA	1	13	0	0
Contador	2	14	0	0
Dif %	50%	7%	0%	0%
6:00 até 12:00				
IA	18	312	0	2
Contador	23	303	0	5
Dif %	22%	3%	0%	60%
12:00 até 18:00				
IA	33	377	0	1
Contador	35	363	0	1
Dif %	6%	4%	0%	0%
18:00 até 0:00				
IA	17	182	1	2
Contador	20	181	0	1
Dif %	15%	1%	100%	50%
Total				
IA	69	884	1	5
Humano	80	861	0	7
Dif %	14%	3%	100%	29%

(a) Movimento 1.

Resumo Resultado Movimento 2:				
Tipo:	Moto	Automovel	Onibus	Caminhao
0:00 até 6:00				
IA	3	28	0	0
Contador	3	28	0	0
Dif %	0%	0%	0%	0%
6:00 até 12:00				
IA	43	602	0	3
Contador	44	601	0	6
Dif %	2%	0%	0%	50%
12:00 até 18:00				
IA	80	692	0	4
Contador	78	688	0	7
Dif %	3%	1%	0%	43%
18:00 até 0:00				
IA	41	345	0	1
Contador	42	344	0	2
Dif %	2%	0%	0%	50%
Total				
IA	167	1667	0	8
Humano	167	1661	0	15
Dif %	0%	0%	0%	47%

(b) Movimento 2.

Resumo Resultado Movimento 3:				
Tipo:	Moto	Automovel	Onibus	Caminhao
0:00 até 6:00				
IA	0	2	0	0
Contador	0	2	0	0
Dif %	0%	0%	0%	0%
6:00 até 12:00				
IA	4	134	0	0
Contador	6	137	0	1
Dif %	33%	2%	0%	100%
12:00 até 18:00				
IA	9	199	0	0
Contador	10	206	0	0
Dif %	10%	3%	0%	0%
18:00 até 0:00				
IA	5	84	0	0
Contador	6	85	0	0
Dif %	17%	1%	0%	0%
Total				
IA	18	419	0	0
Humano	22	430	0	1
Dif %	18%	3%	0%	100%

(c) Movimento 3.

Resumo Resultado Movimento 4:				
Tipo:	Moto	Automovel	Onibus	Caminhao
0:00 até 6:00				
IA	4	70	13	1
Contador	7	69	13	1
Dif %	43%	1%	0%	0%
6:00 até 12:00				
IA	87	1235	57	26
Contador	124	1260	60	30
Dif %	30%	2%	5%	13%
12:00 até 18:00				
IA	119	1499	64	18
Contador	156	1522	61	20
Dif %	24%	2%	5%	10%
18:00 até 0:00				
IA	73	786	52	3
Contador	103	793	53	2
Dif %	29%	1%	2%	33%
Total				
IA	283	3590	186	48
Humano	390	3644	187	53
Dif %	27%	1%	1%	9%

(d) Movimento 4.

Resumo Resultado Movimento 5:				
Tipo:	Moto	Automovel	Onibus	Caminhao
0:00 até 6:00				
IA	1	24	0	0
Contador	2	24	0	0
Dif %	50%	0%	0%	0%
6:00 até 12:00				
IA	27	501	0	5
Contador	31	516	0	6
Dif %	13%	3%	0%	17%
12:00 até 18:00				
IA	46	607	0	9
Contador	52	619	0	10
Dif %	12%	2%	0%	10%
18:00 até 0:00				
IA	29	368	0	3
Contador	32	377	0	3
Dif %	9%	2%	0%	0%
Total				
IA	103	1500	0	17
Humano	117	1536	0	19
Dif %	12%	2%	0%	11%

(e) Movimento 5.

Resumo Resultado Movimento 6:				
Tipo:	Moto	Automovel	Onibus	Caminhao
0:00 até 6:00				
IA	2	35	0	0
Contador	4	35	0	2
Dif %	50%	0%	0%	100%
6:00 até 12:00				
IA	41	1142	0	5
Contador	63	1181	0	6
Dif %	35%	3%	0%	17%
12:00 até 18:00				
IA	85	1505	0	9
Contador	126	1542	0	16
Dif %	33%	2%	0%	44%
18:00 até 0:00				
IA	49	874	2	1
Contador	67	910	2	1
Dif %	27%	4%	0%	0%
Total				
IA	177	3556	2	15
Humano	260	3668	2	25
Dif %	32%	3%	0%	40%

(f) Movimento 6.

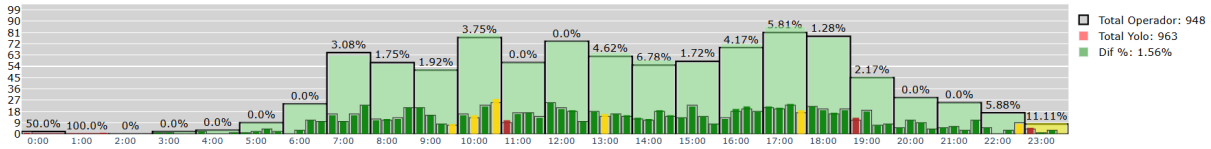
Resumo Resultado Movimento 7:				
Tipo:	Moto	Automovel	Onibus	Caminhao
0:00 até 6:00				
IA	0	1	0	0
Contador	0	1	0	0
Dif %	0%	0%	0%	0%
6:00 até 12:00				
IA	0	91	0	1
Contador	3	84	0	0
Dif %	100%	8%	0%	100%
12:00 até 18:00				
IA	3	98	0	2
Contador	6	83	1	1
Dif %	50%	15%	100%	50%
18:00 até 0:00				
IA	2	52	0	1
Contador	11	43	0	2
Dif %	82%	17%	0%	50%
Total				
IA	5	242	0	4
Humano	20	211	1	3
Dif %	75%	13%	100%	25%

(g) Movimento 7.

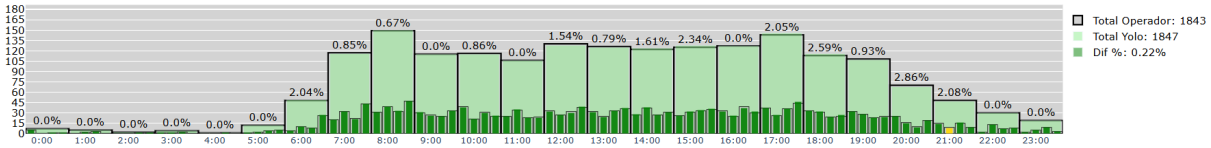
Fonte: Elaborado pelo autor (2025).

A Figura 53 apresenta os sete gráficos de contagem de veículos do posto P023, separados por movimento.

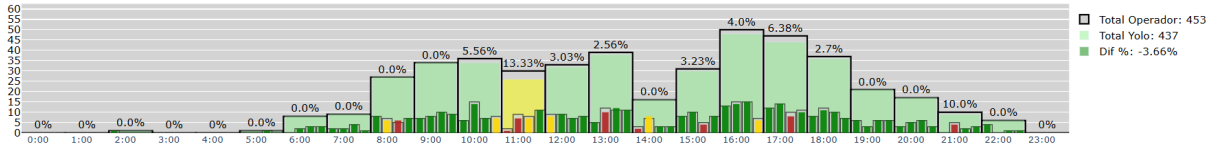
Figura 53 – Gráfico 24 horas de contagem todos os veículos do P023 por movimento.



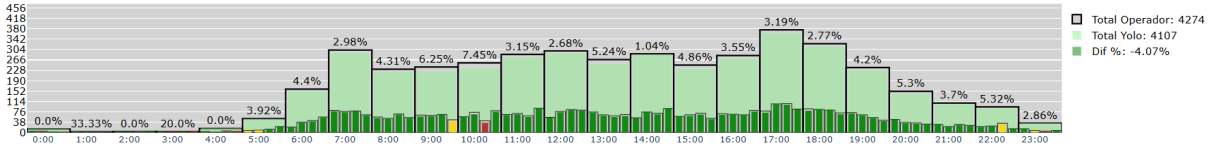
(a) Movimento 1.



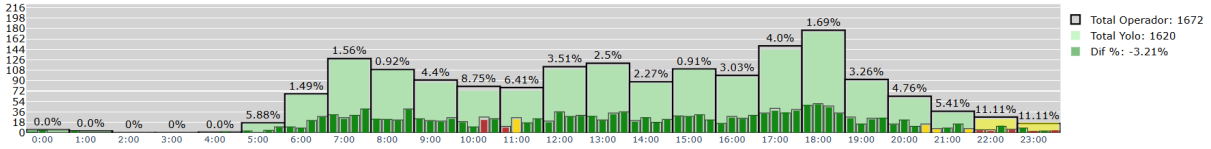
(b) Movimento 2.



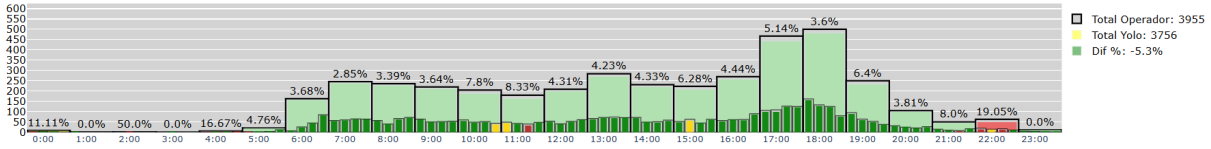
(c) Movimento 3.



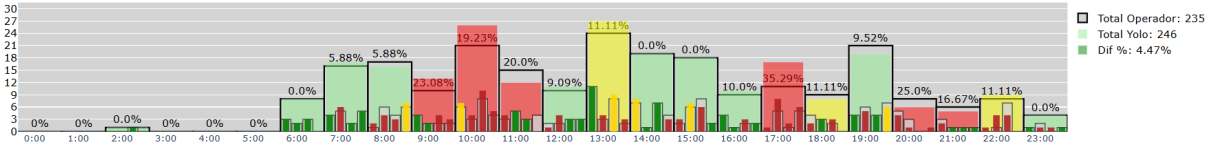
(d) Movimento 4.



(e) Movimento 5.



(f) Movimento 6.



(g) Movimento 7.

Primeiramente, diferentemente do posto P008, nesse posto observam-se diferenças percentuais maiores por cada classe, especialmente para motocicletas e caminhões. No entanto, em alguns movimentos, como os de número 1, 2, 3 e 5, essas diferenças percentuais não correspondem a um número absoluto elevado de erros de contagem. Por exemplo, no Movimento 1, a IA deixou de contabilizar apenas um ônibus

e dois caminhões. Os gráficos desses movimentos refletem bem essa situação, evidenciando que, na maioria dos quartis e faixas horárias, a contagem total permaneceu dentro da margem de erro aceitável (inferior a 10%).

No caso dos movimentos 4 e 6, a contagem de motocicletas ficou significativamente abaixo da referência humana, indicando uma limitação do algoritmo em rastrear esse tipo de veículo nesses fluxos. Curiosamente, ambos os movimentos representam trajetórias de avanço reto, o que, em princípio, sugeriria uma dinâmica mais estável e previsível, tornando mais fácil de rastrear. Essa facilidade em rastreamento esteve presente na contagem de automóveis, que apresentou excelente desempenho, com diferenças inferiores a 6%. Essa acurácia compensa a deficiência na contagem de motocicletas, devido à maior proporção de automóveis no agregado de veículos. Esse comportamento é evidenciado nos gráficos correspondentes, onde os valores totais mantêm-se próximos da referência.

Uma possível explicação para esse fenômeno está relacionado aos fatores discutidos na Seção 4.2. As motocicletas, por serem menores e mais rápidas, frequentemente aparecem borradas nos quadros de vídeo, em função da qualidade da captura. Isso compromete tanto a detecção quanto o rastreamento. Além disso, o Movimento 6 ocorre mais próximo da posição da câmera, o que pode gerar obstruções na linha de visão dos veículos menores que transitam no sentido oposto, que corresponde ao Movimento 4, impactando negativamente sua contagem.

Entre todos os movimentos analisados, o Movimento 7 apresentou o pior desempenho. Mesmo a contagem de automóveis, que geralmente apresenta baixos índices de erro, registrou uma diferença percentual de até 17%. Os gráficos indicam a ocorrência de numerosos falsos positivos. Uma possível justificativa para esse resultado encontra-se no comportamento ilustrado na Figura 47, também discutido na Seção 4.2. Nessa situação, alguns veículos que realizaram o Movimento 4 podem ter sido erroneamente reatribuídos a trajetórias compatíveis com o Movimento 5.

Com isso, tanto os movimentos 4 quanto 5 apresentaram perdas parciais de contagem, atríbuidas indevidamente à contabilização do Movimento 7. Como os volumes de tráfego nos movimentos 4 e 5 são elevados, essas perdas permaneceram dentro da margem aceitável. Contudo, no caso do Movimento 7, que possui menor volume, essas inconsistências foram suficientemente significativas para comprometer a contagem acumulada nas 24 horas e sua distribuição no período.

5 CONCLUSÕES E CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo desenvolver e avaliar um sistema automatizado para contagem volumétrica classificada e direcional de veículos em vias urbanas, utilizando técnicas de aprendizado profundo e visão computacional. A solução proposta integrou três componentes principais: detecção de objetos mediante rede neural convolucional YOLOv8 customizada, rastreamento de múltiplos objetos através do algoritmo ByteTrack e contagem direcional por meio de dois métodos complementares (região e movimento) propostos neste trabalho de pesquisa.

O modelo treinado neste trabalho apresentou desempenho superior ao modelo pré-treinado YOLOv8m disponibilizado pela Ultralytics, com diferença de 22,5% na métrica mAP50-95. Essa melhoria foi atribuída em partes à estratégia de subdividir a categoria automóvel em classes mais específicas durante o treinamento (*car*, van, caminhonete, minivan, Kombi e carroforte), e à utilização de um conjunto de imagens mais similar com o cenário de aplicação. Isso permitiu que o modelo aprendesse adequadamente os padrões visuais de cada silhueta veicular comum no contexto brasileiro.

Outro fator do treinamento a se destacar, é que o *dataset* utilizado é substancialmente menor que o COCO, e ainda assim contribuiu bastante para melhorar a precisão no contexto das filmagens deste trabalho. Isso mostra que o modelo pré-treinado da Ultralytics (2023) aprendeu bem a extrair características de veículos, mas necessita de *fine-tuning* para atingir precisões de alto nível em cenários específicos.

A avaliação em dois cenários urbanos distintos, abrangendo períodos contínuos de 24 horas, revelou desempenhos diferenciados. No Posto P008, caracterizado por vias separadas de sentido único, o sistema alcançou acurácia aceitável (diferenças inferiores a 10%) para a contagem total de veículos em ambos os métodos (região e movimento), com desempenho particularmente elevado para as categorias automóvel e ônibus (diferenças inferiores a 1%). No Posto P023, representando um cruzamento com sete movimentos possíveis, o sistema manteve desempenho satisfatório na maioria dos fluxos, embora tenha apresentado limitações na contagem de motocicletas em movimentos específicos (movimentos 4 e 6) e falsos positivos no movimento de menor volume (movimento 7).

São os problemas encontrados na contagem do Posto P023 que evidenciam as principais limitações do Detector e Rastreador. Apesar da evolução em relação ao modelo pré-treinado, o modelo treinado apresenta dificuldades em detectar motocicletas à noite. Isso pode ser atribuído ao borrão da câmera em capturar esses veículos em alta velocidade e com baixa luminosidade. Outro desafio observado foi as reatribuições incorretas dos identificadores (re-IDs) em situações específicas da via, gerando falsos

positivos na contagem por movimento.

Ainda assim, os resultados evidenciam que o sistema proposto atende aos critérios de validação estabelecidos pela Consultran Engenharia para os cenários avaliados, apresentando-se como uma alternativa viável para automatizar parcial ou totalmente o processo de contagem volumétrica classificada e direcional. A metodologia desenvolvida, estruturada em módulos independentes (detecção, rastreamento e contagem), confere flexibilidade ao sistema, permitindo aprimoramentos futuros em cada componente sem necessidade de reestruturação completa.

O desenvolvimento desse trabalho trouxe uma ampla gama de campos para trabalhos futuros. Sugere-se uma investigação aprofundada nas vantagens da subdivisão das categorias de veículos, para entender até que ponto essa estratégia agrega um ganho real na acurácia de classificação. Além disso, é promissor quantificar a responsabilidade na qualidade de detecção de veículos entre o modelo detector e as condições da imagem de entrada. Assim, pode-se chegar a uma solução ao problema das detecções de motocicletas à noite capturadas por câmeras convencionais com resoluções de 480p. O comportamento das Re-IDs pode ser melhor explorado, já que a inserção de outros parâmetros, como inércia e sentido das vias, poderiam evitar Re-IDs entre veículos que estão trafegando em sentidos opostos. O método de contagem atual depende totalmente dos locais desenhados, sendo que uma má escolha de posição das regiões pode inviabilizar completamente a contagem automática proposta. Portanto, formas de induzir a escolha de regiões apropriadas ou ainda uma escolha automática de regiões de origem e destino na via agregariam muito a esse tipo de ferramenta.

REFERÊNCIAS

ABOUELYAZID, Mahmoud. Comparative Evaluation of SORT, DeepSORT, and ByteTrack for Multiple Object Tracking in Highway Videos. **International Journal of Sustainable Infrastructure for Cities and Societies**, v. 8, n. 11, p. 42–52, 2023. Disponível em: https://vectoral.org/index.php/IJSICS/article/view/97.

ABREU, Diego. **36% dos brasileiros de grandes cidades passam mais de 1 hora por dia no trânsito**. Confederação Nacional da Indústria. 2023. Disponível em: https://noticias.portaldaindustria.com.br/noticias/infraestrutura/36-dos-brasileiros-de-grandes-cidades-passam-mais-de-1-hora-por-dia-no-transito/. Acesso em: 5 set. 2024.

ALBUQUERQUE, Leonardo Sokolowski de. **Diagnóstico automático de COVID-19 baseado em redes neurais profundas usando imagens de raio-X**. 2022. f. 64. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) – Instituto Federal de Santa Catarina, Itajaí, SC.

ANACONDA INC. **Anaconda**. Disponível em: https://anaconda.org. Acesso em: 11 dez. 2025.

BEWLEY, Alex; GE, Zongyuan; OTT, Lionel; RAMOS, Fabio; UPCROFT, Ben. Simple online and realtime tracking. *In:* 2016 IEEE International Conference on Image Processing (ICIP). Phoenix, AZ, USA: IEEE, 2016. p. 3464–3468. DOI: 10.1109/ICIP.2016.7533003.

BROSTRÖM, Mikel. **BoxMOT: A collection of SOTA real-time, multi-object trackers for object detectors**. Versão 10.0.17. 2023. DOI: 10.5281/zenodo.7629840. Disponível em: https://github.com/mikel-brostrom/yolo_tracking. Acesso em: 10 jun. 2023.

CONSULTRAN ENGENHARIA LTDA. Material técnico interno fornecido para fins acadêmicos. Documento não publicado, cedido pela empresa ao autor para elaboração deste trabalho. Itajaí, SC, 2024.

DEPARTAMENTO NACIONAL DE INFRAESTRUTURA DE TRANSPORTES (DNIT). **Manual de Estudos de Tráfego**. Rio de Janeiro, RJ: DNIT, 2006. 384 p.

FEDERAL HIGHWAY ADMINISTRATION. **Traffic Monitoring Guide**. Washington, DC, 2022. p. 1–250. Disponível em: https://www.fhwa.dot.gov/policyinformation/tmguide/2022_TMG_Final_Report.pdf. Acesso em: 24 ago. 2024.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. Cambridge, MA: MIT Press, 2016. p. 800. Disponível em: http://www.deeplearningbook.org.

HARRIS, Mark. **An Even Easier Introduction to CUDA**. NVIDIA Developer Blog. Jan. 2017. Disponível em: https://developer.nvidia.com/blog/even-easier-introduction-cuda. Acesso em: 21 nov. 2025.

HASSAN, Mahbub; MAHIN, Hridoy Deb; JUSOH, Muzammil; NAFEES, Abdullah Al; PAUL, Arpita; ISLAM, Md Ashequl. Vehicle Class Detection and Counting on a Malaysian Road Using YOLOv8 and OpenCV. *In:* IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET). Kota Kinabalu, Malaysia: IEEE, 2024. p. 271–275. DOI: 10.1109/IICAIET62352.2024.10729957.

HUSSAIN, Muhammad. YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. **Machines**, MDPI, Basel, Switzerland, v. 11, n. 7, p. 677, 2023. DOI: 10.3390/machines11070677.

INTELBRAS S.A. **Datasheet do modelo IM5 S**. 2024. Disponível em: https://www.intelbras.com/pt-br/camera-externa-inteligente-wi-fi-im5-sc. Acesso em: 18 ago. 2024.

KHANAM, Rahima; HUSSAIN, Muhammad. **A Review of YOLOv12: Attention-Based Enhancements vs. Previous Versions**. 2025. Disponível em: https://doi.org/10.48550/arXiv.2504.11995. Acesso em: 29 set. 2025.

KUHN, Harold W. The Hungarian method for the assignment problem. **Naval Research Logistics Quarterly**, Wiley, v. 2, n. 1-2, p. 83–97, 1955. DOI: 10.1002/nav.3800020109.

LI, Qian; RANYANG, Li; JI, Kaifan; DAI, Wei. **Kalman Filter and Its Application**. 2015. Disponível em: https: //www.researchgate.net/publication/305871722_Kalman_Filter_and_Its_Application. Acesso em: 18 ago. 2024.

LIN, Jia-Ping; SUN, Min-Te. A YOLO-based Traffic Counting System. *In:* 2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI). Taoyuan, Taiwan: IEEE, 2018. p. 104–107. DOI: 10.1109/TAAI.2018.00027.

LIN, Tsung-Yi; MAIRE, Michael; BELONGIE, Serge; BOURDEV, Lubomir; GIRSHICK, Ross; HAYS, James; PERONA, Pietro; RAMANAN, Deva; ZITNICK, C. Lawrence; DOLLÁR, Piotr. **Microsoft COCO: Common Objects in Context**. 2015. Disponível em: https://doi.org/10.48550/arXiv.1405.0312. Acesso em: 5 set. 2023.

MOTCHALLENGE. Technical University of Munich. 2015. Disponível em: https://motchallenge.net/. Acesso em: 21 abr. 2025.

MURRAY, Samuel. **Real-Time Multiple Object Tracking – A Study on the Importance of Speed**. 2017. Disponível em: https://arxiv.org/abs/1709.03572. Acesso em: 10 out. 2024.

NETO, Orlirio de Souza Tourinho. **Nota técnica: contagem volumétrica manual classificada com uso de aplicativo no telefone celular**. São Paulo, SP, 2015. Disponível em: https://www.cetsp.com.br/media/419681/nt238.pdf. Acesso em: 18 ago. 2024.

PAPERS WITH CODE. **Multi-Object Tracking on MOT20 (MOTA)**. 2023. Disponível em: https://paperswithcode.com/sota/multi-object-tracking-on-mot20-1?metric=MOTA. Acesso em: 6 jul. 2023.

RAZAVI, Saman. Deep learning, explained: Fundamentals, explainability, and bridge ability to process-based modelling. **Environmental Modelling & Software**, Elsevier, v. 144, p. 105159, 2021. DOI: 10.1016/j.envsoft.2021.105159. Disponível em: https://www.sciencedirect.com/science/article/pii/S1364815221002024.

REDMON, Joseph; DIVVALA, Santosh; GIRSHICK, Ross; FARHADI, Ali. You Only Look Once: Unified, Real-Time Object Detection*. In:* IEEE. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, NV, USA: [*s. n.*], 2016. p. 779–788. DOI: 10.1109/CVPR.2016.91.

RUSSELL, Stuart; NORVIG, Peter. **Artificial Intelligence: A Modern Approach**. 4. ed. Hoboken, NJ: Pearson, 2020. p. 1136.

SAMPAIO, Guilherme de S.; LINHARES, José E. B. de S. Detecção, Rastreamento e Contagem de Veículos para Análise de Tráfego Urbano utilizando Métodos de Aprendizagem Profunda. *In:* XV Computer on the Beach. Balneário Camboriú, SC, Brasil: [*s. n.*], abril 2024. p. 10–13.

SCARINGELLA, Roberto. A Crise da Mobilidade Urbana em São Paulo. **São Paulo em Perspectiva**, Fundação SEADE, São Paulo, SP, v. 15, n. 1, p. 55–59, 2001. DOI: 10.1590/S0102-88392001000100007.

SKALSKI, Piotr. **MakeSense: Annotations Made Easy**. 2019. Disponível em: https://www.makesense.ai/. Acesso em: 6 jul. 2023.

SZELISKI, Richard. **Computer Vision: Algorithms and Applications**. 2. ed. New York, NY: Springer Nature, 2022. p. 925.

TERVEN, Juan; CORDOVA-ESPARZA, Diana-Margarita; ROMERO-GONZÁLEZ, Julio-Alejandro. A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. **Machine Learning and Knowledge Extraction**, MDPI, Basel, Switzerland, v. 5, n. 4, p. 1680–1716, 2023. DOI: 10.3390/make5040083.

ULTRALYTICS. **Ultralytics GitHub Repository**. 2023. Disponível em: https://github.com/ultralytics/ultralytics. Acesso em: 18 ago. 2024.

WOJKE, Nicolai; BEWLEY, Alex; PAULUS, Dietrich. **Simple Online and Realtime Tracking with a Deep Association Metric**. 2017. Disponível em: https://arxiv.org/abs/1703.07402. Acesso em: 10 out. 2023.

ZAREEN, Iffat; KHATUN, Ayesha; HASSAN, Khondekar. Evolution of YOLO Architectures: Trends, Applications and Future Research Directions for Object Detection. **TechRxiv**, nov. 2025. DOI: 10.36227/techrxiv.176315286.66846573/v1. Disponível em: https://www.researchgate.net/publication/397626863_Evolution_of_Yolo_Architectures_Trends_Applications_and_Future_Research_Directions_for_Object_Detection. Acesso em: 21 nov. 2025.

ZHANG, Yifu; SUN, Peize; JIANG, Yi; YU, Dongdong; WENG, Fucheng; YUAN, Zehuan; LUO, Ping; LIU, Wenyu; WANG, Xinggang. **ByteTrack: Multi-Object Tracking by Associating Every Detection Box**. 2022. Disponível em: https://arxiv.org/abs/2110.06864. Acesso em: 22 set. 2023.