COMPARAÇÃO DE ABORDAGENS EM UM SISTEMA INTELIGENTE DE CONTROLE SEMAFÓRICO: DETECÇÃO DE PEDESTRES COM VISÃO COMPUTACIONAL

Augusto Gonçalves Satiro[1], Marlon de Matos de Oliveira[2]

Resumo: O crescimento acelerado das cidades e o aumento da frota de veículos têm intensificado os desafios relacionados à mobilidade urbana, tornando necessária a adoção de soluções inteligentes para o gerenciamento do tráfego. Nesse contexto, a visão computacional e o aprendizado profundo destacam-se como tecnologias promissoras para a detecção automática de pedestres em sistemas semafóricos inteligentes. Este trabalho tem como objetivo comparar diferentes métodos de detecção de pedestres aplicados a esse contexto. Foram avaliados dez modelos de detecção de objetos pré-treinados no *dataset* COCO: YOLOv8, SSDLite320 MobileNetV3, Faster R-CNN, RetinaNet, FCOS, EfficientDet-D0, DETR, RT-DETR, *Conditional* DETR e *Deformable* DETR. Os experimentos foram realizados em sete vídeos contendo diferentes cenários de mobilidade urbana, totalizando 4.527 quadros anotados manualmente. Os resultados mostraram que Faster R-CNN, RetinaNet e FCOS obtiveram o maior *recall* no cenário principal de detecção de pedestres parados (80,85%), enquanto o YOLOv8 apresentou o melhor equilíbrio entre desempenho e velocidade, com tempo médio de inferência de 109,34 ms por quadro e F1-score de 79,9%. Em condições de baixa luminosidade, o RT-DETR destacou-se com F1-score de 46,9% e *recall* de 30,6%, superando os demais modelos. Conclui-se que o YOLOv8 apresenta maior viabilidade para aplicações em tempo real, enquanto o RT-DETR demonstra maior robustez em cenários adversos de iluminação.

Palavras-chave: visão computacional; detecção de pedestres; redes neurais convolucionais; semáforos inteligentes; mobilidade urbana.

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), Criciúma - Santa Catarina - Brasil. augustogsatiro@gmail.com

[2]Orientador, Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), Criciúma - Santa Catarina - Brasil. marlon.oliveira@unesc.net

ABSTRACT: The rapid growth of cities and the increasing number of vehicles have intensified urban mobility challenges, requiring intelligent solutions for traffic management. In this context, computer vision and deep learning have emerged as promising technologies for automatic pedestrian detection in intelligent traffic light systems. This study aims to compare different pedestrian detection methods applied to this scenario. Ten object detection models pre-trained on the COCO dataset were evaluated: YOLOv8, SSDLite320 MobileNetV3, Faster R-CNN, RetinaNet, FCOS, EfficientDet-D0, DETR, RT-DETR, Conditional DETR, and Deformable DETR. Experiments were conducted using seven videos representing different urban mobility scenarios, totaling 4,527 manually annotated frames. The results showed that Faster R-CNN, RetinaNet, and FCOS achieved the highest recall in the main pedestrian-waiting scenario (80.85%), while YOLOv8 provided the best balance between performance and speed, reaching an average inference time of 109.34 ms per frame and an F1-score of 79.9%. Under low-light conditions, RT-DETR achieved the best performance, with an F1-score of 46.9% and a recall of 30.6%, outperforming the other models. The findings indicate that YOLOv8 is the most suitable model for real-time applications, whereas RT-DETR demonstrates greater robustness in challenging lighting conditions.

Keywords: computer vision; pedestrian detection; convolutional neural networks; intelligent traffic lights; urban mobility.

1 INTRODUÇÃO

O crescimento acelerado das cidades e da população tem provocado um aumento significativo no tráfego urbano, resultando em engarrafamentos frequentes e no aumento da poluição sonora e ambiental. O trânsito gera impactos não apenas na mobilidade, mas também em aspectos econômicos e sociais: motoristas enfrentam maior gasto com combustível, as empresas sofrem perdas de produtividade e a população é exposta a níveis elevados de estresse e à redução da qualidad8e de vida (SEMOVE, 2024).

Segundo pesquisa da Ford, motoristas em grandes cidades passam, em média, dois dias por ano parados em semáforos. Buscando minimizar esse problema, pesquisas têm investigado formas de tornar o controle do trânsito mais dinâmico, permitindo que o fluxo de veículos ocorra de forma mais eficiente e com menos interrupções desnecessárias (DATAPROM, 2022).

Nesse contexto, o conceito de cidades inteligentes tem ganhado destaque. Trata-se de uma abordagem que busca integrar tecnologia, gestão urbana e sustentabilidade para otimizar serviços públicos e melhorar a qualidade de vida da população. A mobilidade urbana constitui um dos pilares centrais das cidades inteligentes, e os sistemas semafóricos inteligentes surgem como uma das soluções mais promissoras para a gestão eficiente do tráfego urbano (Burlacu; Boboc; Butilă, 2022).

Os sistemas mais comuns para solicitação de travessia são baseados em botões acionados pelos pedestres. No entanto, essas soluções apresentam limitações quanto à praticidade e à precisão, uma vez que dependem da ação manual do usuário. Em muitos casos, os pedestres podem não perceber o dispositivo ou não conseguir utilizá-lo adequadamente, reduzindo a eficiência do sistema e comprometendo a acessibilidade. Dessa forma, técnicas de visão computacional surgem como uma alternativa promissora para a detecção automática de pedestres, permitindo que os sistemas semafóricos se adaptem de forma dinâmica à demanda de travessia. Como resposta a essas limitações, algumas cidades têm adotado soluções inovadoras. Um exemplo é a cidade de Viena, que, em 2019, iniciou a implementação de semáforos inteligentes capazes de identificar, por meio de câmeras, a aproximação de pedestres com intenção de atravessar, ajustando o tempo de sinalização de forma mais eficiente (Observatory, 2019).

No Brasil, a cidade de São Paulo também passou a adotar, em 2023, sistemas de semáforos inteligentes. Nesse caso, o foco foi direcionado ao tráfego de veículos, utilizando câmeras para mensurar a demanda em determinadas vias e, a partir disso, prolongar o tempo de abertura do sinal verde quando necessário (Mendonca, 2023).

Apesar dos avanços na área de detecção de pedestres utilizando técnicas de visão computacional e aprendizado profundo, grande parte dos estudos concentra-se na avaliação dos modelos em *datasets* padronizados, como Caltech e KITTI (Dollar et al., 2012; Geiger; Lenz; Urtasun, 2012), sem considerar plenamente sua aplicação em tempo real. Além disso, desafios permanecem na implementação desses métodos em cenários reais não controlados, especialmente no que se refere à robustez e eficiência computacional (Zhang et al., 2021).

Adicionalmente, poucos trabalhos exploram a integração direta desses métodos com sistemas semafóricos inteligentes, particularmente no que diz respeito à identificação do comportamento do pedestre, como a intenção de travessia (Rasouli; Kotseruba; Tsotsos, 2017; Rasouli; Kotse-

ruba; Tsotsos, 2019; Sadhukhan, 2019).

Identifica-se uma lacuna na literatura quanto à análise comparativa de modelos de detecção de pedestres em cenários reais simplificados, utilizando dispositivos acessíveis, bem como na implementação de protótipos funcionais que simulem a adaptação dinâmica de semáforos com base na presença e no comportamento dos pedestres.

Dessa forma, este trabalho busca preencher essa lacuna ao realizar uma comparação prática entre diferentes modelos de visão computacional em tempo real, utilizando uma câmera comum, além de propor um protótipo capaz de identificar pedestres em espera e simular o funcionamento de um sistema semafórico inteligente.

Entre os principais métodos de detecção de objetos baseados em aprendizado profundo, destaca-se o *You Only Look Once* (YOLO), amplamente utilizado por sua alta velocidade de processamento e capacidade de realizar detecções em tempo real. O modelo processa a imagem de forma unificada, permitindo identificar múltiplos objetos em uma única etapa, o que o torna especialmente adequado para aplicações que exigem baixa latência (Redmon et al., 2016).

Outro método relevante é o *Single Shot MultiBox Detector* (SSD), que também realiza a detecção em uma única passagem pela rede neural. O SSD apresenta um bom equilíbrio entre precisão e desempenho, sendo capaz de detectar objetos em diferentes escalas por meio do uso de múltiplas camadas convolucionais (Liu et al., 2016).

O Faster R-CNN representa uma abordagem baseada em duas etapas, sendo conhecido por sua elevada precisão na detecção de objetos. Apesar de possuir maior custo computacional em comparação aos métodos de etapa única, sua capacidade de gerar propostas de regiões torna-o eficaz em cenários que exigem maior detalhamento nas detecções (Ren et al., 2015).

Além dessas arquiteturas tradicionais, modelos mais recentes têm ampliado as possibilidades da detecção de objetos. O RetinaNet introduziu a função de perda denominada *Focal Loss*, desenvolvida para reduzir o impacto do desbalanceamento entre exemplos positivos e negativos durante o treinamento, melhorando a capacidade de detecção de objetos difíceis (Lin et al., 2017).

O *Fully Convolutional One-Stage Object Detector* (FCOS) representa uma abordagem livre de âncoras (*anchor-free*), eliminando a necessidade de caixas âncora previamente definidas e realizando a detecção di-

retamente sobre os mapas de características extraídos pela rede convolucional (Tian et al., 2019).

O EfficientDet-D0 foi desenvolvido com foco na eficiência computacional, combinando a arquitetura EfficientNet com a estrutura *Bidirectional Feature Pyramid Network* (BiFPN), permitindo um melhor aproveitamento das características em múltiplas escalas e oferecendo uma relação favorável entre precisão e velocidade (Tan; Pang; Le, 2020).

Mais recentemente, arquiteturas baseadas em *Transformers* passaram a ser aplicadas à detecção de objetos. O *Detection Transformer* (DETR) introduziu uma nova abordagem ao utilizar mecanismos de atenção para modelar relações globais na imagem, dispensando etapas tradicionais como geração de propostas de regiões e supressão não máxima (Carion et al., 2020).

A partir do DETR surgiram versões aprimoradas que buscam reduzir suas limitações de velocidade e convergência. O *Real-Time Detection Transformer* (RT-DETR) foi desenvolvido especificamente para aplicações em tempo real, oferecendo menor latência de inferência (Zhao et al., 2024). O *Conditional* DETR utiliza consultas condicionais para acelerar o processo de treinamento e melhorar a convergência do modelo (Meng et al., 2021). Já o *Deformable* DETR emprega mecanismos de atenção deformável que concentram o processamento em regiões relevantes da imagem, reduzindo o custo computacional e melhorando a detecção de objetos pequenos (Zhu et al., 2021).

A comparação entre essas diferentes arquiteturas permite analisar o comportamento de abordagens baseadas em convolução tradicional, métodos *anchor-free*, modelos otimizados para eficiência computacional e arquiteturas fundamentadas em *Transformers*, fornecendo uma visão abrangente das alternativas atualmente disponíveis para aplicações de detecção de pedestres em sistemas semafóricos inteligentes.

Para atingir o objetivo geral, foram definidos os seguintes objetivos específicos: realizar estudos na área de detecção de pedestres com visão computacional; aprimorar o domínio da linguagem Python com ênfase em redes neurais aplicadas à análise de imagens; comparar diferentes modelos de detecção de objetos aplicados à identificação de pedestres; implementar um protótipo funcional de detecção de pessoas utilizando câmeras comuns; e desenvolver uma interface para exibir a visão da câmera e informações do estado atual do semáforo.

A escolha deste tema está diretamente relacionada à necessi-

dade de melhorar a qualidade de vida nas grandes cidades. O crescimento populacional e o aumento da frota de veículos intensificam os congestionamentos urbanos a cada ano, resultando em impactos negativos de ordem social, econômica e ambiental. Entre os principais problemas observados estão a perda de tempo produtivo, o aumento do consumo de combustível, a elevação da emissão de poluentes e os altos níveis de estresse aos quais motoristas e pedestres ficam expostos diariamente (Resende; Souza, 2009).

Perante esse cenário, torna-se relevante investigar soluções tecnológicas capazes de otimizar o fluxo de trânsito e mitigar seus impactos. O uso de técnicas de inteligência artificial, em especial da visão computacional, possibilita a identificação automática de pedestres e a adaptação dinâmica da sinalização (Fred; Victor, 2024).

Estudos na literatura têm investigado a aplicação de técnicas de visão computacional para a detecção de pedestres e sua integração em sistemas de mobilidade urbana. Vargas, Paes e Vasconcelos (2016) analisaram o emprego de redes neurais convolucionais na tarefa de detecção de pedestres, enquanto Sermanet et al. (2013) avaliaram abordagens baseadas em aprendizado profundo em *benchmarks* amplamente utilizados na área. No contexto de sistemas inteligentes de transporte, Sohani et al. (2024) propuseram um sistema semafórico fundamentado em visão computacional para monitoramento do fluxo veicular, ao passo que Ni, Kuehnel e Jiang (2025) investigaram o uso de imagens térmicas para ampliar a acessibilidade em cruzamentos inteligentes.

Embora esses estudos demonstrem o potencial da visão computacional em aplicações relacionadas à mobilidade urbana, observa-se uma predominância de avaliações conduzidas em bases de dados padronizadas ou direcionadas a contextos específicos. Nesse cenário, são limitadas as investigações que realizam comparações experimentais entre diferentes arquiteturas de detecção de objetos em fluxos contínuos de vídeo voltados à identificação de pedestres aguardando a travessia. Assim, este trabalho propõe uma análise comparativa de dez modelos de detecção de objetos aplicados a um sistema semafórico inteligente, considerando conjuntamente métricas de desempenho de detecção, demanda computacional e viabilidade de operação em tempo real.

2 MATERIAIS E MÉTODOS

A metodologia adotada neste trabalho é de natureza aplicada e experimental. O estudo propõe a implementação e comparação prática de dez modelos de detecção de objetos baseados em aprendizado profundo: YOLOv8, SSDLite320 MobileNetV3, Faster R-CNN, RetinaNet, FCOS, EfficientDet-D0, DETR, RT-DETR, Conditional DETR e Deformable DETR, todos aplicados à detecção de pedestres em vídeos utilizados para testes experimentais. Cada modelo foi executado com pesos pré-treinados no *dataset* COCO (Common Objects in Context) e avaliado sobre os mesmos vídeos de entrada, possibilitando a comparação direta de desempenho.

O sistema foi desenvolvido integralmente em Python 3, utilizando as bibliotecas Ultralytics YOLOv8, PyTorch, *torchvision*, OpenCV e NumPy para processamento de imagens e inferência, e *psutil* para monitoramento de recursos computacionais. A etapa prática possui caráter demonstrativo e comparativo, mantendo o foco do estudo na análise conceitual dos métodos e na avaliação objetiva de seu desempenho em condições próximas à aplicação proposta.

2.1 BASE DE DADOS

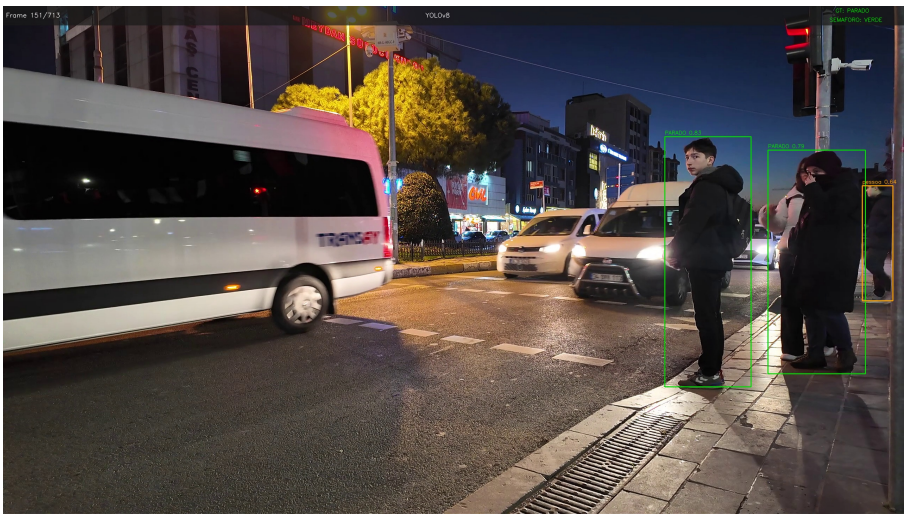
Este trabalho adota uma abordagem experimental baseada na utilização de modelos pré-treinados. A base de dados assume papel indireto: os dez modelos avaliados foram previamente treinados no *dataset* COCO (Common Objects in Context), que contém 330 mil imagens anotadas com até 80 classes de objetos, incluindo a classe *person*, a única classe utilizada nesse estudo.

Para a avaliação prática, não foram utilizados *datasets* estáticos tradicionais. Em vez disso, empregaram-se sete vídeos gratuitos obtidos na plataforma Pexels (2026), selecionados para testes e validação experimental. Os vídeos representam cenários reais simplificados e controlados, permitindo avaliar o método sob diferentes condições.

O primeiro vídeo (*pedestres_cima.mp4*) apresenta uma visão superior de pedestres caminhando sobre uma faixa de travessia, incluindo intervalos com e sem fluxo de pessoas. O segundo (*multiplos_pedestres.mp4*) mostra uma multidão atravessando a rua continuamente, sem períodos de ausência de pedestres. O terceiro (*noite.mp4*) retrata um cenário noturno com pessoas aguardando o momento seguro para atravessar a faixa. O quarto (*mulher_esperando.mp4*) registra uma mulher chegando à faixa de

pedestres e aguardando a passagem dos veículos. O quinto (*grupo.mp4*) apresenta um grupo de quatro pessoas realizando a travessia em conjunto. O sexto (*italia.mp4*) mostra diversas pessoas atravessando uma rua na Itália. Por fim, o sétimo vídeo (*atras.mp4*) registra pedestres atravessando uma via movimentada, com a cena filmada a partir da perspectiva traseira dos veículos. Na Figura 1 é possível visualizar o video *noite.mp4*, imagens dos outros vídeos disponíveis no apêndice A.

Figura 1 - Vídeo noite.mp4



Fonte: Adaptado de Pexels (2026).

As anotações de *ground truth* foram produzidas manualmente por meio de uma ferramenta de rotulação desenvolvida especificamente para este projeto (*annotate.py*), descrita na Seção 2.3.

A padronização dos modelos no mesmo *dataset* de pré-treinamento (COCO) garante que todos compartilhem o mesmo conhecimento prévio, tornando a comparação de desempenho mais consistente e justa.

2.2 MODELOS DE DETECÇÃO

Os dez modelos avaliados foram selecionados por representarem abordagens distintas no espectro velocidade e precisão da detecção de objetos.

Implementado por meio da biblioteca Ultralytics-v8.4.42, o YOLOv8 realiza detecção e rastreamento em uma única etapa de inferência. Neste trabalho, foi utilizada a variante *small* (yolov8s), que oferece bom equilíbrio entre velocidade e precisão para uso em tempo real. A função *model.track()* com *persist=True* mantém o histórico de IDs entre *frames*, eliminando a necessidade de um rastreador externo.

O modelo SSDLite320 com *backbone* MobileNetV3-Large foi carregado via *torchvision* com pesos COCO_V1. Por não possuir rastreamento nativo, o modelo foi integrado a um rastreador de centroide customizado (*CentroidTracker*), descrito na Seção 2.4. A detecção é filtrada pela classe *person* (label 1 no COCO) com limiar de confiança mínima de 0,5.

O Faster R-CNN com *backbone* ResNet-50 e *Feature Pyramid Network* (FPN) foi carregado via *torchvision* com pesos pré-treinados no *dataset* COCO. Assim como o SSD, o modelo requer rastreamento externo, sendo integrado ao *CentroidTracker*. A inferência é realizada sem gradiente, utilizando *torch.no_grad()*, com o objetivo de reduzir o consumo de memória e aumentar a velocidade de processamento.

O RetinaNet com *backbone* ResNet-50 e FPN foi carregado via *torchvision* com pesos pré-treinados no *dataset* COCO. Trata-se de um detector de única etapa que introduz a *Focal Loss* para mitigar o desequilíbrio entre amostras de fundo e de objeto durante o treinamento, permitindo desempenho comparável a detectores de dois estágios com menor custo computacional. Assim como o SSD e o Faster R-CNN, o modelo não possui rastreamento nativo, sendo integrado ao *CentroidTracker*. A detecção é filtrada pela classe *person* (*label* 1 no COCO) com limiar de confiança mínima de $0{,}5$. A inferência é realizada sem gradiente, utilizando *torch.no_grad()*, com o objetivo de reduzir o consumo de memória e aumentar a velocidade de processamento.

O FCOS com *backbone* ResNet-50 e FPN foi carregado via *torchvision* com pesos pré-treinados no *dataset* COCO. Por ser um detector *anchor-free*, o modelo prediz a localização dos objetos diretamente a partir de cada ponto da grade de *feature maps*, sem necessidade de caixas âncora predefinidas, utilizando uma ramificação de *centerness* para suprimir detecções de baixa qualidade distantes do centro do objeto. Assim como os demais modelos sem rastreamento nativo, foi integrado ao *CentroidTracker*, com filtragem pela classe *person* (*label* 1) e limiar de confiança mínima de $0{,}5$. A inferência é realizada sem gradiente, utilizando *torch.no_grad()*.

Implementado por meio da biblioteca Ultralytics, o RT-DETR (*Real-Time Detection Transformer*) foi utilizado na variante *large* (rtdetr-l.pt). O modelo combina uma *backbone* convolucional eficiente com um decodificador *Transformer*, buscando unir a velocidade dos detectores de única etapa com a qualidade de detecção dos modelos baseados em atenção, eliminando a necessidade de supressão não-máxima (*Non-Maximum Suppression*) em pós-processamento. Por compartilhar a interface da biblioteca

Ultralytics, o RT-DETR utiliza a mesma abordagem do YOLOv8: a função *model.track()* com *persist=True* mantém o histórico de IDs entre *frames*, eliminando a necessidade de um rastreador externo.

O DETR (*Detection Transformer*) foi carregado via biblioteca *Hugging Face Transformers*, utilizando o modelo pré-treinado facebook/detr-resnet-50 com a classe *DetrForObjectDetection*. Proposto como o primeiro detector a formular a detecção de objetos como um problema de predição de conjuntos por meio de um *encoder-decoder Transformer*, o modelo elimina componentes como *anchors* e *Non-Maximum Suppression*. O pré-processamento das imagens é realizado pelo *DetrImageProcessor*, e as detecções são extraídas pelo método *post_process_object_detection* com limiar de $0{,}5$. Por não possuir rastreamento nativo, o modelo foi integrado ao *CentroidTracker*, filtrando a classe *person* (*label* 1 no COCO). A inferência é realizada sem gradiente, utilizando *torch.no_grad()*.

O EfficientDet-D0 foi carregado por meio da biblioteca *effdet*, utilizando a função *create_model* na configuração efficientdet_d0 com pesos pré-treinados no COCO. O modelo emprega a *backbone* EfficientNet combinada com a rede de agregação de características BiFPN (*Bi-directional Feature Pyramid Network*) e um cabeçalho de detecção compartilhado entre escalas. A entrada exige redimensionamento fixo para 512×512 *pixels*, seguido de normalização com os parâmetros ImageNet. As coordenadas das caixas detectadas são reescalonadas para as dimensões originais do *frame*. Sem rastreamento nativo, o modelo foi integrado ao *CentroidTracker*, filtrando a classe *person* (*label* 1) com confiança mínima de $0{,}5$.

O *Deformable* DETR foi carregado via *Hugging Face Transformers*, utilizando o modelo SenseTime/deformable-detr com a classe *DeformableDetrForObjectDetection* e pré-processamento pelo *AutoImageProcessor*. O modelo introduz módulos de atenção deformável multi-escala que atribuem peso apenas a um conjunto reduzido de pontos de referência por *query*, acelerando a convergência e melhorando a detecção de objetos pequenos em comparação ao DETR original. As detecções são extraídas com limiar de $0{,}5$, filtrando a classe *person* (*label* 1 no COCO). Por não possuir rastreamento nativo, o modelo foi integrado ao *CentroidTracker*. A inferência é realizada sem gradiente, utilizando *torch.no_grad()*.

O *Conditional* DETR foi carregado via *Hugging Face Transformers*, utilizando o modelo microsoft/conditional-detr-resnet-50 com a classe *ConditionalDetrForObjectDetection* e pré-processamento pelo *ConditionalDetrImageProcessor*. O modelo introduz atenção cruzada condicional, na

qual as *queries* de conteúdo condicionam as consultas espaciais ao decodificador *Transformer*, permitindo que o modelo localize extremidades e regiões distintas do objeto com maior precisão e acelerando a convergência do treinamento em relação ao DETR original. As detecções são extraídas com limiar de $0{,}5$, filtrando a classe *person* (*label* 1 no COCO). Sem rastreamento nativo, o modelo foi integrado ao *CentroidTracker*.

2.3 FERRAMENTA DE ANOTAÇÃO DE GROUND TRUTH

O termo *ground truth* (verdade de referência) representa os dados de referência utilizados para validação de algoritmos, correspondendo às informações consideradas corretas sobre o conteúdo analisado. Neste estudo, o *ground truth* foi produzido manualmente por meio da rotulação quadro a quadro da presença de pedestres nos vídeos de teste.

Para possibilitar a avaliação quantitativa dos modelos, foi desenvolvida a ferramenta *annotate.py*, que permite criar arquivos de *ground truth* para vídeos de teste. A ferramenta exibe o vídeo quadro a quadro e registra, para cada *ground*, se há ou não pedestre presente na cena e se está ou não parado, com base na observação direta do operador.

Os controles de interação são: tecla 0 para marcar ausência de pedestre a partir do *frame* atual, tecla 1 para marcar pedestre em movimento, tecla 2 para marcar pedestre parado, barra de espaço para pausar e retomar a reprodução e *ESC* para salvar e encerrar. O resultado é salvo em formato *JSON*, mapeando cada número de *frame* ao rótulo correspondente (0, 1 ou 2). Este arquivo é utilizado diretamente pelo *script* de avaliação comparativa, que converte os rótulos em predição binária considerando positivo apenas o rótulo 2 (pedestre parado aguardando travessia) para o cálculo das métricas.

2.4 RASTREADOR DE CENTROIDE (CENTROIDTRACKER)

Como a maioria dos modelos avaliados não possui mecanismo nativo de rastreamento entre *frames*, ao contrário do YOLOv8 e do RT-DETR, foi implementado o *CentroidTracker*, um rastreador baseado em distância euclidiana entre centroides.

A cada *frame*, o rastreador recebe a lista de centroides detectados e os associa aos objetos já rastreados calculando a matriz de distâncias D, onde cada elemento $D[i][j]$ é a norma L2 entre o centroide do objeto i e a nova detecção j. A correspondência é feita de forma gulosa: os pares são ordenados pela menor distância e atribuídos um a um, sem repetição

de linhas ou colunas. Objetos sem correspondência por mais de 30 *frames* consecutivos são removidos do registro. Novos centroides sem par são registrados como novos objetos com ID único incremental.

Esse mecanismo garante consistência de identidade entre *frames* para os modelos que não possuem rastreamento integrado, permitindo que o algoritmo de detecção de parada descrito na seção seguinte opere de forma equivalente nos dez modelos avaliados.

2.5 ALGORITMO DE DETECÇÃO DE PEDESTRE PARADO

Para identificar pedestres em espera de travessia, foi implementado o algoritmo *is_stopped()*, presente nos dez módulos de detecção em tempo real. O algoritmo analisa o histórico de posições de cada pedestre rastreado para determinar se o indivíduo está estático.

O funcionamento é o seguinte: a posição central (centroide) da *bounding box* de cada pedestre é registrada a cada *frame* em um histórico indexado pelo ID de rastreamento. Quando o histórico acumula pelo menos *FRAMES_PARADO = 45 frames* (equivalente a aproximadamente 1,5 segundo a 30 *frames per second* (FPS)), calcula-se a média do deslocamento euclidiano quadro a quadro sobre as últimas 45 posições, conforme a expressão $mov = \text{mean}\left(\|pos[t] - pos[t-1]\|_2\right), \quad t \in \{1, \ldots, 45\}$.

Se o valor de mov for inferior ao limiar *LIMIAR_PARADO = 10 pixels* por *frame*, o pedestre é classificado como parado e em espera de travessia. A exigência de uma janela de 45 *frames* evita classificações prematuras causadas por movimentos instantâneos lentos, requerendo que o pedestre permaneça estático por um período mínimo antes de o sinal ser alterado.

2.6 PROTOCOLO DE AVALIAÇÃO COMPARATIVA

A avaliação comparativa dos dez modelos é conduzida pelo *script evaluate.py*, que processa um vídeo de entrada com cada modelo sequencialmente e compara os resultados contra o *ground truth* produzido pelo *annotate.py*. Para execução em lote sobre múltiplos vídeos, o *script run_all.py* automatiza o processo, localizando todos os pares vídeo e *ground truth* na pasta *videos* e gerando um CSV individual por vídeo além de um arquivo consolidado. O limiar de confiança mínima adotado para todas as detecções é de $0{,}5$.

Para cada *frame*, a predição do modelo é comparada ao respectivo rótulo do *ground truth*, permitindo a construção da matriz de confusão

utilizada no cálculo das métricas de avaliação.

Os resultados são exportados para arquivo CSV contendo métricas de desempenho computacional e de qualidade das detecções para análise posterior.

2.7 MÉTRICAS DE AVALIAÇÃO

A avaliação dos modelos considera o tempo médio por *frame* (ms), correspondente ao tempo de inferência, e a confiança média das detecções da classe *person*. A qualidade das detecções é avaliada por meio da matriz de confusão, composta por verdadeiros positivos (TP), falsos positivos (FP), falsos negativos (FN) e verdadeiros negativos (TN). A partir desses valores, são calculadas as métricas de precisão ($TP/(TP + FP)$), *recall* ($TP/(TP + FN)$), F1-score e acurácia ($(TP + TN)/Total$).

2.8 AMBIENTE DE TESTES

Os experimentos foram conduzidos em ambiente controlado, sem exposição a cenários urbanos reais. Os vídeos de teste utilizados foram obtidos gratuitamente na internet e selecionados por apresentarem situações de movimento e parada de pedestres em cenários compatíveis com a proposta experimental do trabalho. Essa abordagem permite isolar o desempenho dos algoritmos de condições externas imprevisíveis, mantendo o foco na comparação objetiva entre os modelos.

A viabilidade técnica do sistema em cenários reais é discutida de forma descritiva, considerando os requisitos de hardware, as limitações operacionais observadas nos experimentos e o potencial de uso em tempo real identificado a partir dos resultados obtidos.

3 RESULTADOS E DISCUSSÃO

Esta seção apresenta os resultados obtidos a partir da execução dos dez modelos de detecção de pedestres avaliados YOLOv8, SSD, Faster R-CNN, RetinaNet, FCOS, EfficientDet, DETR, RT-DETR, *Conditional* DETR e *Deformable* DETR bem como a análise comparativa de seus desempenhos no contexto de um sistema semafórico inteligente. Os experimentos foram conduzidos com os sete vídeos de teste descritos na Seção 2.5, totalizando 4.527 *frames* anotados.

A avaliação foi conduzida de acordo com o protocolo descrito na Seção 2.3, com foco na detecção de pedestres parados em espera de travessia (rótulo 2 do *ground truth*). Os vídeos *italia.mp4*, *grupo.mp4*, *mul-*

tiplos_pedestres.mp4 e *pedestres_cima.mp4*, por não conterem nenhum *frame* com pedestre parado, serviram exclusivamente para quantificar a taxa de falsos positivos, ou seja, classificações equivocadas de pedestres em movimento como se estivessem parados.

A Tabela 1 apresenta os resultados consolidados para o vídeo *mulher_esperando.mp4*, cenário com maior proporção de *frames* com pedestre parado (940 de 998), onde as métricas de precisão, *recall* e *F1* são mais representativas.

Tabela 1 - Resultados comparativos no vídeo *mulher_esperando.mp4*

Modelo	**ms/frame**	**Precisão**	**Recall**	**F1**	**Acurácia**	**Conf.**
YOLOv8	109,34	1,000	0,665	0,799	0,685	0,774
SSD	**56,35**	1,000	0,569	0,725	0,595	0,902
Faster R-CNN	1.701,44	1,000	0,809	**0,894**	**0,820**	**0,909**
RetinaNet	1.660,73	1,000	**0,809**	**0,894**	**0,820**	0,800
FCOS	1.429,88	1,000	0,809	**0,894**	**0,820**	0,706
EfficientDet	151,87	1,000	0,559	0,717	0,585	0,704
DETR	1.020,51	1,000	0,729	0,843	0,745	0,907
RT-DETR	504,33	1,000	0,676	0,806	0,695	0,855
Conditional DETR	1.074,31	1,000	0,782	0,878	0,795	0,759
Deformable DETR	4.280,51	1,000	0,644	0,783	0,665	0,804

Fonte: Elaborado pelo autor.

3.1 RESULTADOS POR VÍDEO DE TESTE

No vídeo *mulher_esperando.mp4*, dos 200 *frames* analisados (amostrados a cada 5 *frames* do total de 998), 188 correspondem a pedestre parado aguardando travessia e 12 a ausência de pedestre ou pedestre em movimento. Esse cenário simula diretamente o evento que deve acionar o sistema semafórico proposto em condições de boa iluminação com um único pedestre.

Nesse cenário, todos os dez modelos alcançaram precisão de 100%, sem gerar nenhum falso positivo. A distinção entre os modelos concentra-se, portanto, no *recall*: Faster R-CNN, RetinaNet e FCOS lideraram com $80,85\%$ (152 TP de 188 positivos), seguidos de *Conditional* DETR ($78,19\%$), DETR ($72,87\%$), RT-DETR ($67,55\%$) e YOLOv8 ($66,49\%$). Os modelos com menor *recall* foram SSD ($56,91\%$), *Deformable* DETR ($64,36\%$) e EfficientDet ($55,85\%$).

No vídeo *noite.mp4*, a condição de baixa luminosidade impõe maior dificuldade a todos os modelos. Os resultados são apresentados na Tabela 2. O RT-DETR destacou-se como o modelo mais eficaz nesse cenário, com 30 verdadeiros positivos de 98 possíveis (*recall* de $30,61\%$),

precisão de 100% e *F1-score* de $46,87\%$. Os modelos SSD, RetinaNet e EfficientDet não produziram nenhum verdadeiro positivo nesse cenário. O FCOS apresentou o segundo melhor *F1-score* $(30,25\%)$ com apenas 3 falsos positivos.

Tabela 2 - Resultados no vídeo *noite.mp4* (baixa luminosidade)

Modelo	TP	FP	FN	TN	Precisão	Recall	F1
YOLOv8	6	0	92	**45**	**1,000**	0,061	0,115
SSD	0	0	**98**	**45**	0,000	0,000	0,000
Faster R-CNN	12	**24**	86	21	0,333	0,122	0,179
RetinaNet	0	11	**98**	34	0,000	0,000	0,000
FCOS	18	3	80	42	0,857	0,184	0,303
EfficientDet	0	0	**98**	**45**	0,000	0,000	0,000
DETR	7	10	91	35	0,412	0,071	0,122
RT-DETR	**30**	0	68	**45**	**1,000**	**0,306**	**0,469**
Conditional DETR	14	4	84	41	0,778	0,143	0,241
Deformable DETR	14	0	84	**45**	**1,000**	0,143	0,250

Fonte: Elaborado pelo autor.

No vídeo *atras.mp4*, a configuração é distinta: dos 1.280 *frames* totais, apenas 57 correspondem a pedestre parado, enquanto 1.222 apresentam pedestres em movimento e 1 ausência de pedestre. Esse cenário testa a capacidade dos modelos de identificar um breve evento de parada em meio a uma longa sequência de movimento. A Tabela 3 apresenta os resultados.

Tabela 3 - Resultados no vídeo *atras.mp4*

Modelo	TP	FP	FN	TN	Precisão	Recall	F1
YOLOv8	57	870	0	353	0,062	1,000	0,116
SSD	0	0	**57**	**1223**	0,000	0,000	0,000
Faster R-CNN	57	776	0	447	0,068	1,000	0,128
RetinaNet	57	719	0	504	0,074	1,000	0,137
FCOS	57	829	0	394	0,064	1,000	0,121
EfficientDet	57	183	0	1040	**0,238**	1,000	**0,384**
DETR	57	901	0	322	0,060	1,000	0,112
RT-DETR	57	**1076**	0	147	0,050	1,000	0,096
Conditional DETR	57	834	0	389	0,064	1,000	0,120
Deformable DETR	57	871	0	352	0,061	1,000	0,116

Fonte: Elaborado pelo autor.

Nos vídeos sem *frames* com pedestre parado (*italia.mp4*, *grupo.mp4*, *multiplos_pedestres.mp4* e *pedestres_cima.mp4*), todos os modelos obtiveram acurácia de 100% nos vídeos *grupo.mp4* e *multiplos_pedestres.*

mp4 (FP = 0 em todos). No vídeo *pedestres_cima.mp4*, igualmente nenhum modelo gerou falso positivo. No vídeo *italia.mp4*, o YOLOv8 produziu 19 falsos positivos de 67 *frames* analisados, enquanto os demais modelos geraram entre 0 e 15 FP. Os cenários correspondentes aos vídeos utilizados nos experimentos podem ser visualizados nas Figuras 1 a 7 do Apêndice A.

3.2 ANÁLISE POR MODELO

YOLOv8: apresentou velocidade de processamento compatível com uso em tempo real ($109,34$ ms por *frame*), sendo um dos três modelos a operar abaixo de 200 ms. Seu rastreamento nativo via *model.track()* com *persist=True* favorece a estabilidade das identidades entre *frames*. No cenário principal (*mulher_esperando.mp4*), obteve *F1-score* de $79,87\%$ com precisão perfeita. No cenário noturno, seu desempenho caiu significativamente (*recall* de $6,12\%$), sugerindo sensibilidade à qualidade da iluminação.

SSD: foi o modelo mais rápido avaliado ($56,35$ ms por *frame*), viabilizando operação em tempo real. Entretanto, seu desempenho revelou-se inconsistente: no cenário principal alcançou *recall* de $56,91\%$ com precisão de 100%, mas no cenário noturno não produziu nenhum verdadeiro positivo. No vídeo *atras.mp4*, o SSD foi o único modelo que não detectou nenhuma das 57 instâncias de parada (TP = 0). A menor estabilidade das *bounding boxes* geradas pelo SSDLite320 introduz variações nos centroides calculados a cada *frame*, impedindo que o limiar de deslocamento do algoritmo *is_stopped()* seja satisfeito de forma consistente.

Faster R-CNN: arquitetura de dois estágios com *backbone* ResNet-50 e FPN, obteve o maior *recall* entre os modelos no cenário principal, empatado com RetinaNet e FCOS ($80,85\%$), e precisão de 100%. Seu principal limitante é o tempo de inferência: $1.701,44$ ms por *frame*, tornando inviável sua aplicação em sistemas de controle em tempo real sem aceleração por GPU dedicada. No cenário noturno, gerou 24 falsos positivos ao lado de apenas 12 verdadeiros positivos.

RetinaNet: detector de estágio único com *backbone* ResNet-50 e FPN, alcançou o mesmo *recall* do Faster R-CNN no cenário principal ($80,85\%$) com tempo de inferência ligeiramente menor ($1.660,73$ ms). No cenário noturno, não produziu nenhum verdadeiro positivo, embora tenha gerado 11 falsos positivos, indicando detecções de pedestres em movimento classificados erroneamente como parados.

FCOS: detector totalmente convolucional sem âncoras, também

com *backbone* ResNet-50 e FPN, empatou com Faster R-CNN e RetinaNet no *recall* do cenário principal ($80,85\%$). No cenário noturno, destacou-se como o melhor modelo de estágio único com rastreamento externo, obtendo 18 verdadeiros positivos e apenas 3 falsos positivos (*F1-score* de $30,25\%$).

EfficientDet: variante D0 da arquitetura EfficientDet, apresentou velocidade próxima ao tempo real ($151,87$ ms) e *recall* de $55,85\%$ no cenário principal. Seu comportamento mais conservador resultou em zero falsos positivos em todos os vídeos sem pedestre parado. No vídeo *atras.mp4*, foi o modelo com melhor equilíbrio entre precisão e *recall*: detectou todos os 57 eventos de parada com apenas 183 falsos positivos, resultando em precisão de $23,75\%$ e *F1-score* de $38,38\%$ o mais alto nesse cenário. No cenário noturno, porém, não produziu nenhum verdadeiro positivo.

DETR: transformador de detecção baseado em ResNet-50 via *Hugging Face*, operando em $1.020,51$ ms por *frame*. Alcançou *recall* de $72,87\%$ no cenário principal com precisão perfeita, posicionando-se acima dos modelos mais lentos de arquitetura baseada em âncoras. No cenário noturno, gerou 10 falsos positivos ao lado de apenas 7 verdadeiros positivos.

RT-DETR: variante em tempo real dos transformadores de detecção, via *Ultralytics* com pesos *rtdetr-l*, com rastreamento nativo integrado. Operando em $504,33$ ms por *frame*, foi o único modelo a se destacar no cenário noturno, atingindo *recall* de $30,61\%$, precisão de 100% e *F1-score* de $46,87\%$ os melhores valores nesse vídeo. No cenário principal, obteve *recall* de $67,55\%$ e *F1-score* de $80,63\%$.

Conditional DETR: variante do DETR com *queries* espaciais condicionais (*microsoft/conditional-detr-resnet-50*), que reduz o número de épocas de treinamento necessárias mantendo a arquitetura *encoder-decoder*. Operando em $1.074,31$ ms, alcançou o segundo melhor *recall* no cenário principal ($78,19\%$), com *F1-score* de $87,76\%$, superando inclusive o DETR original em todas as métricas nesse vídeo.

Deformable DETR: variante com atenção deformável, foi o modelo mais lento avaliado ($4.280,51$ ms por *frame*), operando a menos de $0{,}3$ FPS. Apesar da alta capacidade teórica de representação, seu desempenho no cenário principal (*recall* $= 64,36\%$, *F1-score* = $78,32\%$) não justificou o custo computacional adicional em relação a alternativas mais rápidas.

3.3 COMPARAÇÃO GERAL E IMPLICAÇÕES PARA O SISTEMA SEMAFÓRICO

A análise conjunta dos resultados evidencia uma divisão clara entre os modelos avaliados, organizada ao longo de dois eixos: velocidade de inferência e capacidade de detecção em condições adversas.

Em condições de boa iluminação (vídeo *mulher_esperando.mp4*), todos os dez modelos obtiveram precisão de 100%, sem gerar falsos positivos no cenário controlado. A diferenciação ocorre no *recall*: Faster R-CNN, RetinaNet e FCOS lideraram com $80,85\%$, seguidos de *Conditional* DETR ($78,19\%$), enquanto EfficientDet ($55,85\%$) e SSD ($56,91\%$) registraram os menores valores.

No cenário noturno (vídeo *noite.mp4*), a qualidade de detecção degradou significativamente para a maioria dos modelos. O RT-DETR foi o único a manter resultados expressivos, com *F1-score* de $46,87\%$ e precisão perfeita. Três modelos (SSD, RetinaNet e EfficientDet) não produziram nenhuma detecção correta nesse cenário.

No vídeo *atras.mp4*, o SSD foi o único modelo incapaz de detectar qualquer evento de parada. Todos os demais alcançaram *recall* de 100%, porém às custas de elevadas taxas de falsos positivos, indicando que o algoritmo *is_stopped()* classificou indevidamente pedestres em movimento como parados ao longo da longa sequência de 1.280 *frames*. O EfficientDet destacou-se com o maior *F1-score* nesse cenário ($38,38\%$), sugerindo maior seletividade na identificação de parada real.

Para um sistema semafórico inteligente, a latência de resposta é uma restrição operacional crítica. Modelos que operam acima de 500 ms por *frame*, Faster R-CNN, RetinaNet, FCOS, DETR, *Conditional* DETR e *Deformable* DETR, não são viáveis para processamento de vídeo em tempo real sem aceleração por GPU dedicada. Entre os modelos com tempo de inferência inferior a 200 ms, o YOLOv8 ($109,34$ ms) e o EfficientDet ($151,87$ ms) apresentam os melhores resultados no cenário principal, enquanto o SSD ($56,35$ ms) é o mais rápido, mas com desempenho inconsistente entre cenários. O RT-DETR ($504,33$ ms), embora não atinja o processamento em tempo real irrestrito, destaca-se por ser o único modelo com desempenho robusto no cenário noturno.

3.4 LIMITAÇÕES

Os resultados apresentados foram obtidos em ambiente controlado, com vídeos selecionados por apresentarem condições favoráveis à

avaliação experimental. A generalização do desempenho observado para cenários urbanos reais está sujeita a variáveis não contempladas nos testes, como variação de iluminação, oclusões parciais, chuva, diferentes densidades de tráfego de pedestres e diversidade de ângulos de câmera.

O hardware utilizado nos experimentos não possui aceleração por GPU dedicada, o que penaliza diretamente os modelos com maior custo computacional. Em ambientes com GPU, o tempo de inferência de modelos como Faster R-CNN, DETR e *Deformable* DETR pode ser reduzido significativamente, alterando a comparação de viabilidade para aplicações em tempo real.

A avaliação foi conduzida com amostragem de *frames* (*skip_frames* = 5) na maioria dos vídeos, o que reduz o número de *frames* analisados e afeta diretamente o algoritmo *is_stopped()*: para acionar a detecção de parada, um pedestre precisa ser rastreado por 45 *frames* analisados consecutivos, equivalentes a 225 *frames* originais na configuração de amostragem utilizada. Essa característica pode subestimar o desempenho real dos modelos em vídeos curtos ou com pedestres que aparecem brevemente. O vídeo *atras.mp4* foi avaliado sem amostragem (*skip_frames* $= 1$), o que torna seus resultados diretamente comparáveis à operação em tempo real.

3.5 COMPARAÇÃO COM TRABALHOS CORRELATOS

Os resultados desta pesquisa demonstram que a avaliação prática de modelos de detecção de pedestres em vídeos contínuos permite identificar diferenças relevantes entre precisão, robustez e custo computacional, aspectos que não são evidenciados em análises exclusivamente teóricas. Enquanto Vargas, Paes e Vasconcelos (2016) discutem o potencial das redes neurais convolucionais para detecção de pedestres, o presente estudo complementa essa discussão ao comparar experimentalmente dez arquiteturas distintas em um contexto de aplicação semafórica.

Além da capacidade de detecção, observou-se que fatores como tempo de inferência, estabilidade do rastreamento e desempenho em cenários noturnos influenciam diretamente a viabilidade de implantação em sistemas inteligentes de trânsito. Esse tipo de análise não foi explorado pelos trabalhos de Sermanet et al. (2013), cujo foco esteve principalmente na maximização da precisão em *benchmarks* padronizados.

Outra contribuição observada foi a identificação de diferenças significativas entre modelos quando submetidos a cenários de iluminação reduzida. O RT-DETR apresentou maior robustez nesse cenário, resultado

que amplia a discussão sobre aplicações práticas de visão computacional em mobilidade urbana. Diferentemente de Sohani et al. (2024), que concentram a aplicação em veículos, esta pesquisa demonstrou a viabilidade da utilização desses métodos para apoio à travessia de pedestres.

Por fim, os resultados indicam que sistemas baseados em câmeras RGB convencionais e hardware acessível podem vir a ser utilizados para implementação de soluções inteligentes de baixo custo. Essa constatação complementa estudos como o de Ni, Kuehnel e Jiang (2025), que utilizam sensores especializados voltados à acessibilidade, mostrando que abordagens mais simples também apresentam potencial de aplicação em cidades inteligentes.

4 CONCLUSÃO

Este trabalho teve como objetivo comparar diferentes abordagens de visão computacional aplicadas à detecção de pedestres em sistemas semafóricos inteligentes. Para isso, foram avaliados dez modelos de detecção de objetos em vídeos representando diferentes cenários de mobilidade urbana, além da implementação de um protótipo capaz de identificar pedestres e simular a adaptação dinâmica de um sistema semafórico.

Os resultados demonstraram que todos os modelos avaliados foram capazes de detectar pedestres em diferentes níveis de desempenho, evidenciando a influência de fatores como velocidade de inferência, robustez a condições adversas e estabilidade do rastreamento. Observou-se que modelos com maior precisão nem sempre apresentam viabilidade para aplicações em tempo real devido ao elevado custo computacional. Entre os modelos analisados, o YOLO destacou-se pelo equilíbrio entre desempenho e velocidade de processamento, enquanto o RT-DETR apresentou maior robustez em cenários com baixa iluminação.

Como principal contribuição, esta pesquisa realizou uma comparação prática entre diferentes arquiteturas de detecção de objetos em um contexto voltado à mobilidade urbana, considerando não apenas a capacidade de detecção, mas também aspectos relacionados à viabilidade de implantação em sistemas semafóricos inteligentes. Além disso, foi desenvolvido um protótipo funcional utilizando hardware acessível e câmeras convencionais, demonstrando o potencial de aplicação dessas tecnologias em soluções de cidades inteligentes.

Entre as limitações do estudo, destacam-se a utilização de vídeos previamente selecionados e a realização dos experimentos em am-

biente controlado, fatores que restringem a generalização dos resultados para cenários urbanos reais. Também devem ser considerados aspectos como condições climáticas, variações de iluminação, oclusões e diferentes posicionamentos de câmera.

Como trabalhos futuros, sugere-se a realização de testes em ambientes urbanos reais, a integração com sensores e tecnologias de *Internet of Things* (IoT) e a avaliação de versões otimizadas dos modelos mais precisos para execução em hardware embarcado. Essas iniciativas podem contribuir para a evolução de sistemas semafóricos inteligentes mais eficientes, adaptativos e seguros para pedestres e veículos.

5 DECLARAÇÃO DE USO DE INTELIGÊNCIA ARTIFICIAL

Os autores declaram que ferramentas de Inteligência Artificial generativa foram utilizadas exclusivamente como apoio auxiliar à elaboração deste trabalho, incluindo assistência na revisão linguística, estruturação textual e organização de ideias. Nenhuma ferramenta de IA foi utilizada para substituição da autoria intelectual, análise científica, interpretação dos resultados ou tomada de decisões acadêmicas. A responsabilidade integral pelo conteúdo, originalidade, veracidade das informações e conformidade com os princípios de integridade científica permanece integralmente atribuída aos autores.

REFERÊNCIAS

BURLACU, M.; BOBOC, R. G.; BUTILă, E. V. **Smart cities and transportation: Reviewing the scientific character of the theories**. Sustainability, v. 14, n. 13. ISSN 2071-1050. 2022, Disponível em: <https://www.mdpi.com/2071-1050/14/13/8109>. Acesso em: 05 set. 2025.

CARION, N. et al. **End-to-end object detection with transformers**. In: **European Conference on Computer Vision (ECCV)**. [S.l.: s.n.], 2020. 2020. p. 213–229.

DATAPROM. **Semáforos inteligentes: entenda como eles funcionam**. 2022. Disponível em: <https://dataprom.com/como-funcionam-semaforos-inteligentes/>. Acesso em: 20 ago. 2025.

DOLLAR, P. et al. **Pedestrian detection: An evaluation of the state of the art**. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2012. 2012.

FRED, T.; VICTOR, R. **Ai-based pedestrian detection and traffic light control for smart cities**. 2024.

GEIGER, A.; LENZ, P.; URTASUN, R. **Are we ready for autonomous driving? the kitti vision benchmark suite**. In: **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2012. 2012.

LIN, T.-Y. et al. **Focal loss for dense object detection**. In: **Proceedings of the IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2017. 2017. p. 2980–2988.

LIU, W. et al. **Ssd: Single shot multibox detector**. In: **European Conference on Computer Vision (ECCV)**. [s.n.], 2016. 2016. Disponível em: <https://arxiv.org/abs/1512.02325>. Acesso em: 14 set. 2025.

MENDONCA, F. **São Paulo: Semáforos inteligentes, entenda como funciona nova tecnologia**. 2023. Disponível em: <https://www.jacidade.com.br/noticia/sao-paulo-semaforos-inteligentes-entenda-como-funciona-nova-tecnologia>. Acesso em: 03 set. 2025.

MENG, D. et al. **Conditional detr for fast training convergence**. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2021. 2021. p. 3651–3660.

NI, X.; KUEHNEL, C.; JIANG, X. **Thermal detection of people with mobility restrictions for barrier reduction at traffic lights controlled intersections**. 2025, Disponível em: <https://arxiv.org/abs/2505.08568>. Acesso em: 18 set. 2025.

OBSERVATORY, E. U. M. **Vienna to introduce smart traffic lights**. 2019. Disponível em: <https://urban-mobility-observatory.transport.ec.europa.eu/news-events/news/vienna-introduce-smart-traffic-lights-2019-06-27_en>. Acesso em: 20 ago. 2025.

Pexels. **Pexels**. 2026. <https://www.pexels.com/pt-br/>. Acesso em: 18 maio 2026.

RASOULI, A.; KOTSERUBA, I.; TSOTSOS, J. K. **They are not all equal: Pedestrian intention and behavior prediction**. In: **IEEE Intelligent Vehicles Symposium (IV)**. [S.l.: s.n.], 2017. 2017.

RASOULI, A.; KOTSERUBA, I.; TSOTSOS, J. K. **Pie: A large-scale dataset and models for pedestrian intention estimation and trajectory prediction**. In: **IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2019. 2019.

REDMON, J. et al. **You only look once: Unified, real-time object detection**. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [s.n.], 2016. 2016. p. 779–788. Disponível em: <https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf>. Acesso em: 09 set. 2025.

REN, S. et al. **Faster r-cnn: Towards real-time object detection with region proposal networks**. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2015, Disponível em: <https://arxiv.org/abs/1506.01497>. Acesso em: 12 set. 2025.

RESENDE, P. d. T. V.; SOUZA, P. R. d. **Mobilidade urbana nas grandes cidades brasileiras: um estudo sobre os impactos do congestionamento**. Fundação Dom Cabral, Caderno de ideias CI, v. 910. 2009.

SADHUKHAN, P. **An intelligent traffic control system using iot and fuzzy logic**. IEEE Internet of Things Journal. 2019.

SEMOVE. **O impacto negativo do trânsito na saúde**. 2024. Disponível em: <https://semove.org.br/noticias/o-impacto-negativo-do-transito-na-saude/>. Acesso em: 02 set. 2025.

SERMANET, P. et al. **Pedestrian detection with unsupervised multi-stage feature learning**. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2013. 2013.

SOHANI, A. et al. **Smart traffic light control system**. INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT, v. 08, p. 1–6. 2024.

TAN, M.; PANG, R.; LE, Q. V. **Efficientdet: Scalable and efficient object detection**. In: **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2020. 2020. p. 10781–10790.

TIAN, Z. et al. **Fcos: Fully convolutional one-stage object detection**. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2019. 2019. p. 9627–9636.

VARGAS, A. C. G.; PAES, A.; VASCONCELOS, C. N. **Um estudo sobre redes neurais convolucionais e sua aplicação em detecção de pedestres**. In: **Proceedings of the XXIX Conference on Graphics, Patterns and Images**. [S.l.: s.n.], 2016. 2016.

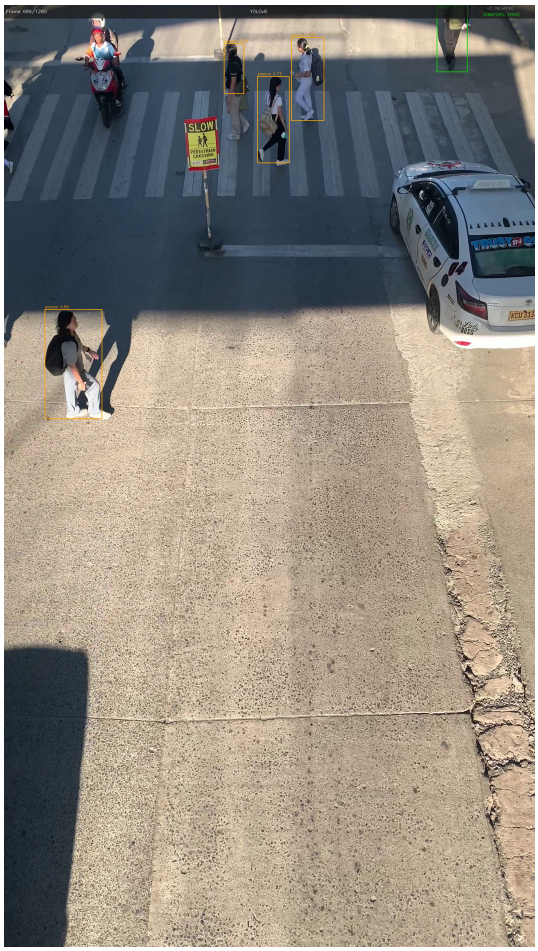
ZHANG, S. et al. **Deep learning-based pedestrian detection: A review**. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2021.

ZHAO, Y. et al. **Detrs beat yolos on real-time object detection**. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2024.

ZHU, X. et al. **Deformable detr: Deformable transformers for end-to-end object detection**. In: **International Conference on Learning Representations (ICLR)**. [S.l.: s.n.], 2021. 2021.

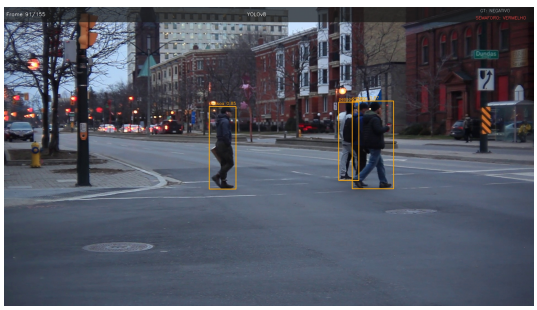
APÊNDICE A

Figura 2 - atras.mp4



Fonte: Adaptado de Pexels (2026).

Figura 3 - grupo.mp4

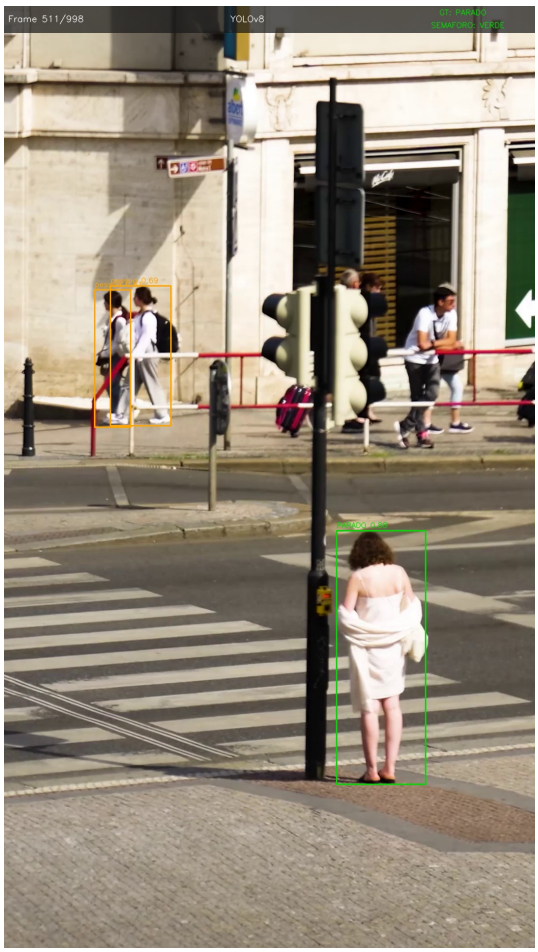


Fonte: Adaptado de Pexels (2026).

Figura 4 - italia.mp4

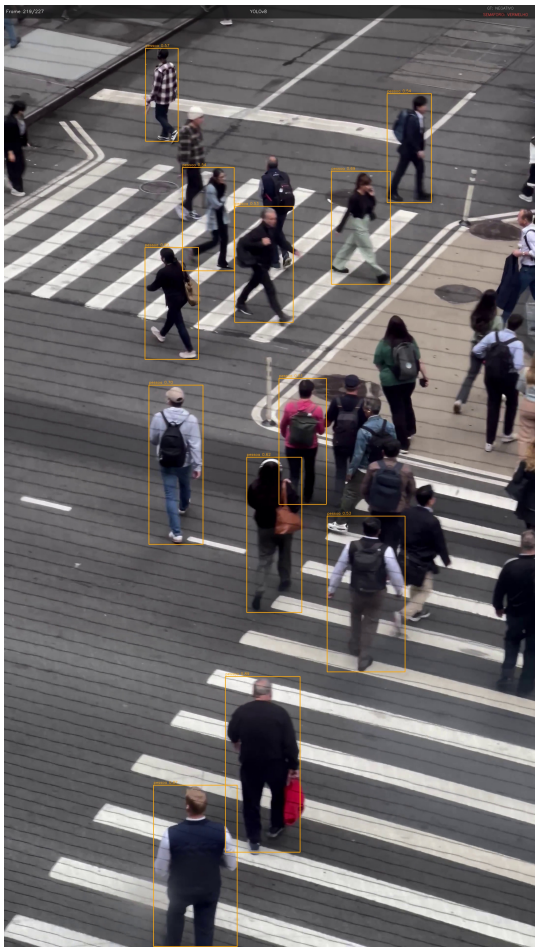
Fonte: Adaptado de Pexels (2026).

Figura 5 - mulher_esperando.mp4



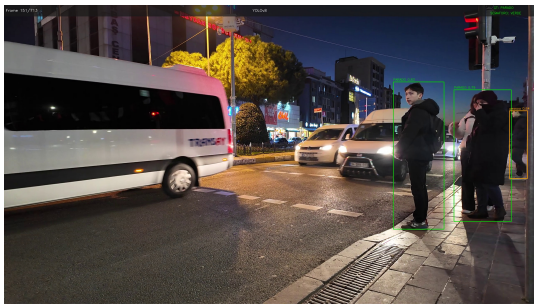
Fonte: Adaptado de Pexels (2026).

Figura 6 - multiplos_pedestres.mp4



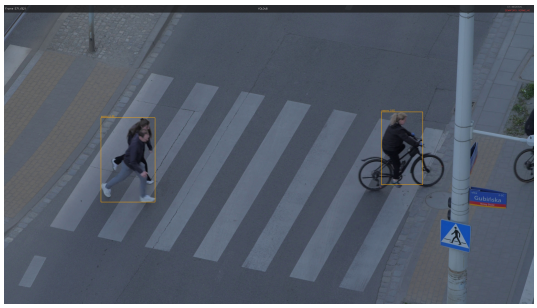
Fonte: Adaptado de Pexels (2026).

Figura 7 - noite.mp4



Fonte: Adaptado de Pexels (2026).

Figura 8 - pedestres_cima.mp4



Fonte: Adaptado de Pexels (2026).