

Smart Vision: Detecção de modelo e placa de veículos

Bruno Camilo Silvério¹, José Matias Lemes Filho¹

¹Engenharia de Computação - Pontifícia Universidade Católica de Campinas (PUC-Campinas) Rodovia Dom Pedro I-km 136 Parque das Universidades CEP 13086-900 São Paulo — SP – Brazil

bcamilo94@gmail.com, jose.matias@puc-campinas.edu.br

Abstract. *This article proposes a solution for video analysis using artificial intelligence to recognize and identify vehicle models and license plates, without the need for the video to be watched. The results are presented clearly and objectively. The tool, named Smart Vision, can assist in numerous scenarios and areas that require this type of analysis, such as public security, by searching for vehicle models that individuals are using to commit crimes, and detecting the license plate, if possible, to search for its data in the system. This allows the tool's purpose to adapt to the needs of law enforcement officers.*

Resumo. *Este artigo propõe uma solução para a análise de vídeos utilizando inteligência artificial com o objetivo de reconhecer e identificar modelo de veículos e placa, sem que o mesmo precise ser assistido, apresentando os resultados de maneira clara e objetiva. A ferramenta, nomeada de Smart Vision, é capaz de auxiliar em inúmeros cenários e áreas que necessitem desse tipo de análise, como em segurança pública, através da busca pelos modelos de carros que pessoas estão utilizando com o intuito de cometer crimes e detecção da placa se possível para fazer busca no sistema pelos dados do mesmo, possibilitando que seu objetivo se adapte conforme a necessidade dos agentes de segurança.*

1. INTRODUÇÃO

Vídeos estão presentes em todos os momentos de nossas vidas, seja para registrar acontecimentos, em estudos, para informações, diversão e outros inúmeros motivos. A origem dos mesmos é diversa, sendo de câmeras profissionais, *smartphones*, *notebooks*, *tablets*, câmeras de segurança e muitos outros aparelhos.

Com o avanço da tecnologia, cada vez mais conteúdo digital é produzido, graças às facilidades geradas pelos aparelhos de gravação e ao aumento da capacidade de armazenamento, incluindo a possibilidade de armazenamento em nuvem, principalmente nos *smartphones*, o que faz com que maiores sejam os tamanhos e a quantidade de vídeos produzidos.

Mesmo com o avanço da tecnologia e as mudanças no nosso cotidiano, é notável que ainda existem setores no Brasil que não evoluíram o quanto deveriam. Um setor do qual dependemos diariamente é o de segurança pública. Por ser um serviço essencialmente estatal, é evidente que a evolução ocorre de forma mais lenta e, em alguns locais, até de forma precária. O investimento em tecnologia, infraestrutura e capacitação muitas vezes é insuficiente, mudando conforme a gestão do momento.

Esse projeto visa auxiliar o trabalho policial, mais especificamente o dos investigadores, ao investigar casos onde há a necessidade de procurar veículos em filmagens. No ambiente de trabalho, para que a ferramenta possa ser utilizada, é necessário um computador com acesso à internet, que pode estar disponível dentro da delegacia, na sala do investigador-chefe, por exemplo. A ferramenta também pode ser utilizada em outras máquinas, possibilitando consultas em sistemas governamentais, como Sinesp, Detran, sistemas internos da polícia, e também para pesquisas na Internet.

Policiais frequentemente precisam analisar imagens provenientes de câmeras de vigilância privada ou públicas, com o objetivo de procurar veículos. Na maioria das vezes, esses vídeos têm mais de uma hora de duração, e o trabalho é realizado de forma manual, onde o policial precisa assistir ao vídeo do início ao fim. Esse processo torna a jornada de trabalho cansativa e impede o policial de lidar com outras ocorrências, já que ele fica preso à análise do vídeo em busca de algum elemento relevante. Outro desafio identificado é a falta de um sistema independente do Estado para o monitoramento de veículos. Para acessar os sistemas de monitoramento, como o da prefeitura, é necessário obter autorização e a busca do veículo é realizada por outro órgão público, como o CIMCamp em Campinas. Somente após essa busca o órgão aciona os policiais, o que pode atrasar investigações.

Com toda essa relevância dos vídeos em nossas vidas, tecnologias como *Video Analytics*, também conhecida como Análise Inteligente de Vídeo, começam a ser desenvolvidas, uma vez que buscam “entender” e “analisar” o conteúdo dos vídeos, realizando tarefas como reconhecimento e detecção de objetos e pessoas, contagem de elementos, rastreamento de elementos na imagem e até mesmo análises comportamentais de pessoas presentes nos vídeos [Segurança Eletrônica 2018]. Suas aplicações são abundantes e estão presentes em inúmeros segmentos, como segurança, ensino, saúde e muitos outros. As funções e os objetivos dessas tecnologias dependem muito do contexto em que estão inseridas, geralmente trazendo excelentes resultados quando bem utilizadas.

No campo da segurança pública, diversas tarefas podem ser realizadas com o auxílio dessa tecnologia. No caso de câmeras de segurança, informações úteis para investigações podem ser extraídas a partir dos vídeos obtidos, mas assisti-los e analisá-los manualmente demanda tempo e recursos. Em uma operação da Polícia Federal conhecida como “Luz na Infância”, foram apreendidos e analisados 312 mil arquivos, com um volume total de 3,8 *terabytes*, exemplificando a magnitude da demanda por esse tipo de análise [Campos, R. 2019].

Analisar vídeos manualmente é uma tarefa demorada e tediosa, além de suscetível a erros. Considerando que os vídeos podem conter informações valiosas, o processo de análise precisa ser otimizado. Por isso, fornecer uma ferramenta que automatize a análise de vídeos e a detecção de veículos e placas em contextos de segurança pública torna-se essencial.

A proposta do presente trabalho é desenvolver um sistema web onde o policial cadastrado possa acessar o sistema, inserir o arquivo em vídeo, e a ferramenta realizará a análise, separando os frames que contêm veículos. Sempre que possível, a ferramenta focará apenas nos frames dos veículos que possuem as características desejadas, como modelo e placa. A ferramenta, então, comparará as placas dos veículos com uma base de dados e, se encontrar um veículo procurado por roubo ou furto, o sistema

automaticamente registra o momento em que aparece no vídeo, trazendo informações sobre o mesmo.

2. Desenvolvimento

2.1 Dataset

Esse *dataset* foi feito manualmente devido à dificuldade em obter imagens através de bases públicas, como *Kaggle* e *COCO*. O conjunto de dados contém 1242 imagens de dois modelos de veículos: 669 imagens do Chevrolet Onix 2016 e 573 do Fiat Palio Fire. Essas imagens foram obtidas por meio de buscas no Google Imagens e em *sites* de venda de veículos, como o *Webmotors*.

As imagens apresentam variações em detalhes como brilho, distância e qualidade, com fotos tiradas durante o dia. No entanto, não foram feitas alterações adicionais, como ajustes de posição ou zoom, devido à padronização na forma dos veículos, que geralmente têm um formato retangular na horizontal, como ilustrado na Figura 1.



Figura 1. Exemplo de imagens rotuladas com os objetos devidamente marcados

Além da obtenção das imagens, foi necessário criar as *labels* correspondentes, também demonstradas na Figura 1. Os rótulos foram nomeados como "Onix" e "Palio" e indicam a localização dos veículos nas imagens. Para esse processo, utilizamos a plataforma *RoboFlow*, que tem integração direta com o *YOLOv8*, facilitando a importação e manipulação do *dataset* durante o treinamento do modelo no *Google Colab*.

As imagens foram divididas em três conjuntos: 70% para treinamento, 20% para validação e 10% para teste. Também foi realizado um pré-processamento, com redimensionamento das imagens para 640x640 pixels, para garantir a compatibilidade com o modelo *YOLOv8*.

Para a detecção de placas não foi necessário criar um *dataset* do zero, existe muitos exemplares na Internet. Utilizei um modelo OCR treinado com mais de 10 mil imagens de placas de veículos [Khan, M. 2024].

2.2 Rede Neural

Para o treinamento da rede neural capaz de detectar modelos de automóveis em vídeos, com foco nos modelos Onix e Palio, foi utilizado o *YOLOv8* [ULTRALYTICS. *YOLOv8 Documentation*. 2024]. O objetivo foi facilitar a construção e o treinamento das redes neurais, além da implantação dos modelos treinados. Em conjunto com a *API*, foi utilizada uma rede previamente treinada, que serviu como base no treinamento deste trabalho.

- O *YOLOv8* foi lançado em janeiro de 2023 pela *Ultralytics* e, até o momento, é a versão mais recente da família de modelos de detecção de objetos *YOLO* (*You Only Look Once*);
- Ele foi projetado para ser rápido, preciso e fácil de usar, além de ser altamente flexível, suportando vários formatos de exportação;
- O *YOLOv8* pode ser treinado em grandes conjuntos de dados e possui total suporte para rodar tanto em *CPUs* quanto em *GPUs*;
- O modelo oferece suporte para segmentação e classificação de imagens, além da detecção de objetos.

A técnica empregada para o treinamento da rede neural é o *YOLO*, um modelo popular de detecção de objetos desenvolvido por Joseph Redmon e Ali Farhadi na Universidade de Washington. A primeira versão foi lançada em 2015 e rapidamente ganhou popularidade devido à sua alta velocidade e precisão. Desde então, várias variações do *YOLO* surgiram, algumas com colaboração dos autores originais, e outras independentes.

Principais características do *YOLO*:

- Método de detecção de objetos de "passada única" (*single pass*), utilizando uma rede convolucional como extrator de características (*features*);
- Ao contrário de outros algoritmos de detecção (como *R-CNN* ou *Faster R-CNN*), o *YOLO* processa a imagem apenas uma vez, enviando-a diretamente para a rede neural;
- A rede utiliza as características da imagem inteira para detectar múltiplas caixas delimitadoras, cada uma contendo um objeto;
- As dimensões pré-definidas (*âncoras*) são utilizadas para descrever os objetos anotados no conjunto de treinamento.

O algoritmo do *YOLO* funciona dividindo a imagem em um grid de $S \times S$ células. Geralmente, uma grade de 13×13 (Figura 2) é usada, embora versões mais recentes tendam a utilizar o tamanho 19×19 . Cada célula é responsável por prever até 5 caixas delimitadoras, caso haja mais de um objeto naquela célula (uma caixa delimitadora descreve o retângulo que cobre o objeto).



Figura 2. Imagem dividida em um grid 13x13 [REDMON, C. J. 2018]

O algoritmo retorna uma pontuação de confiança, que indica a certeza de que a caixa delimitadora contém um objeto. Essa pontuação não indica qual tipo de objeto, apenas se há algo presente. Para cada caixa, a célula também prevê uma classe, fornecendo um valor de probabilidade para cada uma das classes possíveis.

A confiança da caixa delimitadora e a previsão de classe são combinadas em uma pontuação final. Por exemplo, uma caixa com bordas mais grossas pode ter 85% de certeza de conter um cachorro.

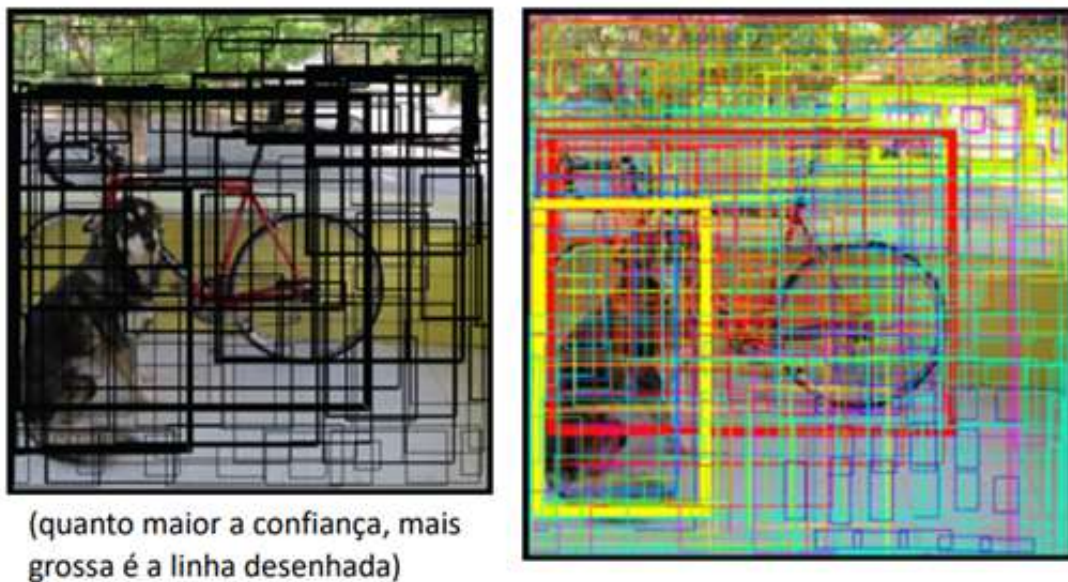


Figura 3. Imagens com as caixas delimitadoras, onde é possível ver a confiança por objeto [REDMON, C. J. 2018]

Na Figura 3, com uma grade 13x13, há 169 células, e cada uma detecta até 5 caixas delimitadoras, resultando em um total de 845 caixas. A maioria dessas caixas tem pontuações de confiança baixas, e geralmente são consideradas apenas aquelas com

pontuação acima de 30% (*threshold*). O *YOLO* realiza todas as detecções simultaneamente, demonstrando a eficiência dessa técnica. Tendo o resultado da predição a Figura 4.

O resultado final da predição:

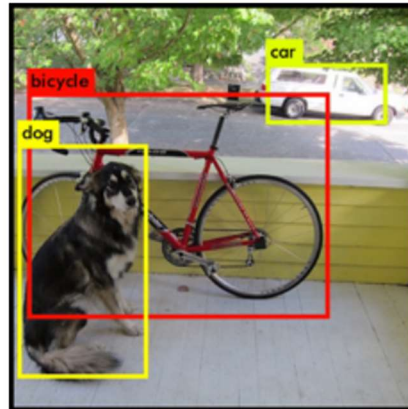


Figura 4. Resultado da predição [REDMON, C. J. 2018]

A equipe que desenvolveu o *YOLOv8* é da empresa *Ultralytics*, os mesmos autores do *YOLOv5*, que se destacou como um "concorrente" do *YOLOv4*, como mostra a Figura 5. Embora o *YOLOv8* tenha sido desenvolvido por uma equipe diferente, a comunidade e os autores reconhecem-no como parte da série *YOLO* devido à sua arquitetura baseada nas versões anteriores.

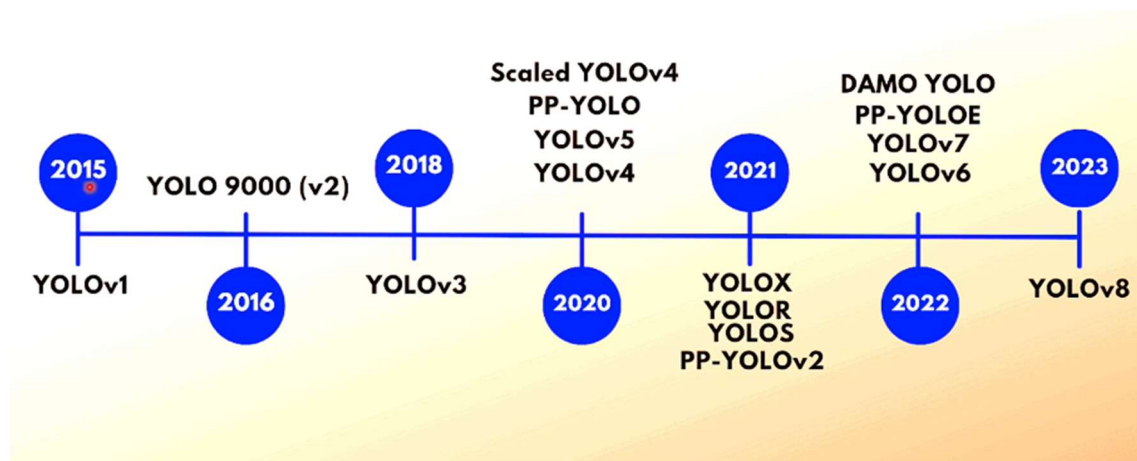


Figura 5. Linha do tempo do modelo YOLO [IA Expert, 2023]

Para a classificação e localização dos objetos, citando a documentação: O *YOLOv8* é um modelo de última geração (*state-of-the-art*, ou *SOTA*) que se baseia no sucesso das versões anteriores do *YOLO* e apresenta novos recursos e melhorias para aumentar ainda mais o desempenho e a flexibilidade.

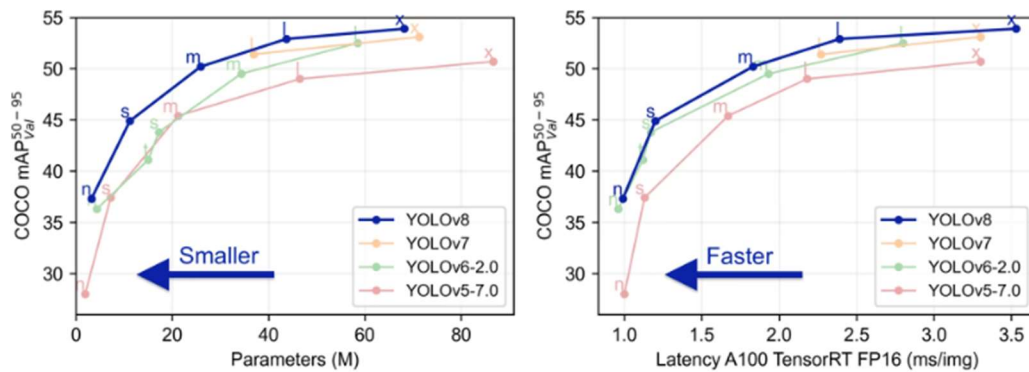


Figura 6. Comparação com outras versões do YOLO [Ultralytics YOLOv8, 2024]

Entre as novidades apresentadas na Figura 6 comparando o tamanho e velocidade, podemos destacar a nova rede de *backbone*, um novo cabeçote de detecção sem âncora (*anchor-free detection head*) e nova função de perda. Essas funcionalidades foram os principais fatores que tornaram o processamento ainda mais rápido.

A rede neural previamente treinada utilizada neste trabalho foi a rede convolucional *YOLOv8m*, que consegue detectar objetos em até 1.83 ms e 50.2 mAP (*mean Average Precision*). O *YOLOv8m* foi treinada a partir do *dataset COCO* (*Common Objects in Context*), um banco de imagens voltado para a detecção de objetos, possuindo mais de 330 mil imagens e 80 categorias de objetos. Optei pelo *YOLOv8m* por trazer um equilíbrio entre desempenho e precisão, pensando no contexto de recurso computacional para treinar a rede.

Semelhante às outras versões mais recentes do YOLO, seus autores disponibilizam o modelo pré-treinado em 5 versões, como mostra a Figura 7: *YOLOv8n* (*nano*), *YOLOv8s* (*small*), *YOLOv8m* (*medium*), *YOLOv8l* (*large*), *YOLOv8x* (*extra large*). O *Nano* é o mais rápido e o menor (menos pesado para rodar), enquanto que o *Extra Large* (*YOLOv8x*) é o mais preciso porém mais pesado para rodar, portanto será mais lento.

Confira a acurácia de cada modelo do YOLOv8 (testado no COCO)

Lembrando que COCO (*Common Objects in Context*) é a "referência" (chamado de benchmark) padrão usada na indústria para avaliar modelos de detecção de objetos de última geração

Model	size (pixels)	mAP _{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Observação: isso é referente ao release 0.0.0, a tendência é que os resultados sejam ainda melhores em novas atualizações desse repositório

Figura 7. Tabela de comparação entre os modelos [Ultralytics YOLOv8, 2024]

3. Sistema

A ferramenta permite a análise de vídeos, apresentando o momento exato do vídeo onde aparece o veículo para facilitar a análise dos momentos relevantes. O sistema de

monitoramento desenvolvido neste trabalho, com capacidade de detectar veículos e placas, consiste em quatro módulos, conforme exemplificado na Figura 8: um servidor *Flash* para realizar as detecções, um *json-server* simulando o Sinesp para consulta de placas e uma interface em *Next.js* para o gerenciar e visualizar as investigações criadas pelos agentes de segurança, que faz comunicação com o *firebase*.

O cliente em foco deste trabalho é a polícia, mais especificamente investigadores. Mas os usuários podem ser quem tiver contato direto com a interface, como, supervisores, agentes de segurança pública.

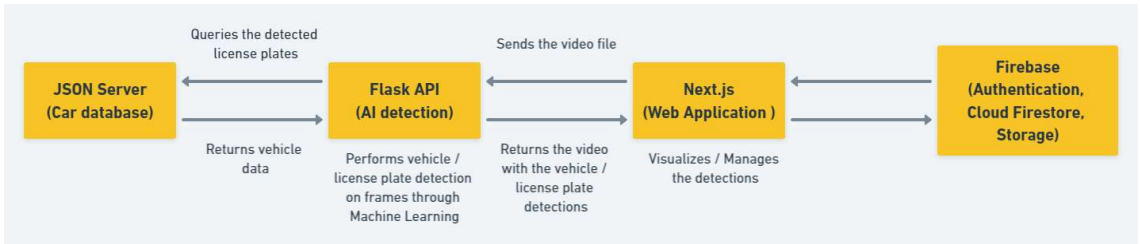


Figura 8. Diagrama do fluxo do sistema

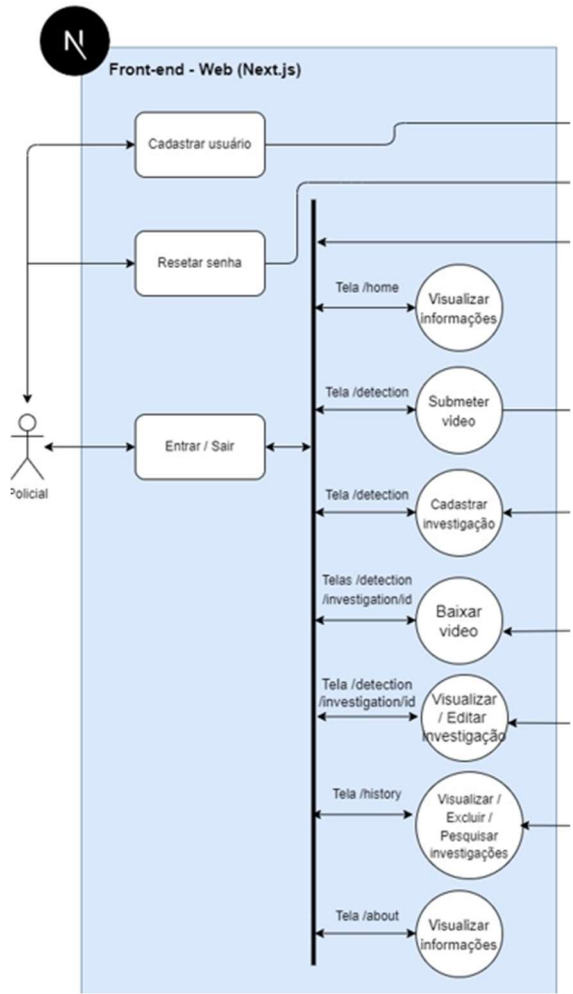


Figura 9. Caso de uso do sistema

3.1. Front-end

O *Front-end* é responsável pelos módulos que interagem com o usuário. Esse módulo foi desenvolvido utilizando a linguagem *TypeScript*, com o objetivo de garantir maior organização e inteligibilidade do código. Para sua construção, foi utilizado o *framework Next.js*, que oferece facilidades no desenvolvimento das interfaces do sistema, proporcionando uma boa experiência para o usuário.

Baseado no diagrama de caso de uso apresentado na Figura 9, o sistema oferece várias funcionalidades para o policial, que pode realizar diversas ações diretamente através da interface. Entre as principais funções estão:

Cadastro, que permite que novos usuários (policiais) se registrem no sistema; *Login* e *Logout*, para que o policial possa acessar sua conta de forma segura e realizar o *logout* ao encerrar o uso; e *Reset* de senha, que oferece a opção de redefini-la caso o usuário a esqueça. Além disso, o sistema possibilita o *upload* de vídeo e o registro de novas investigações, permitindo que o usuário realize o *upload* de vídeos relacionados à investigação, que serão processados pelo sistema, e então cadastre novas investigações.

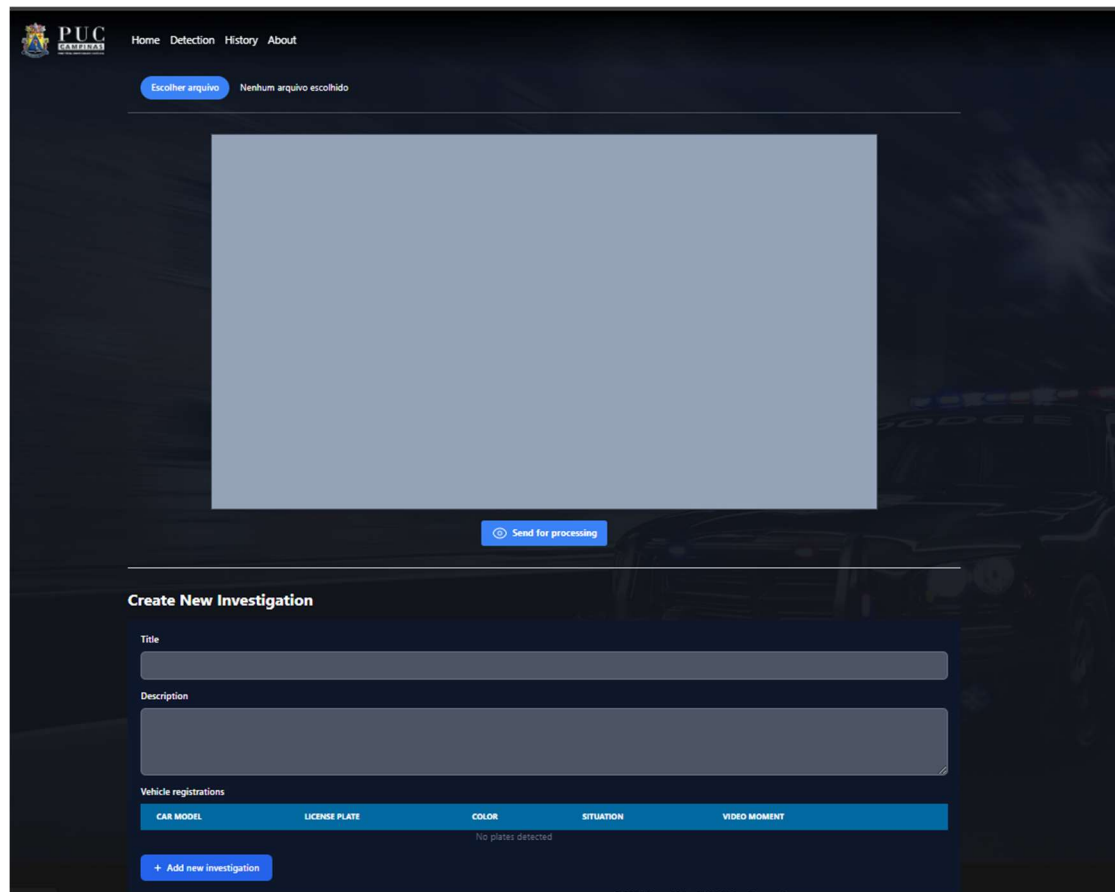


Figura 10. Tela para cadastrar uma investigação

Após o processamento, o sistema permite que o policial visualize os frames resultantes, destacando os momentos em que elementos de interesse foram detectados. O histórico de investigações anteriores pode ser visualizado ou pesquisado, conforme ilustrado na Figura 11, e o usuário tem a opção de baixar os vídeos processados, caso

seja necessário. Além disso, o sistema oferece a funcionalidade de editar ou excluir investigações já cadastradas, conforme mostrado na Figura 12.

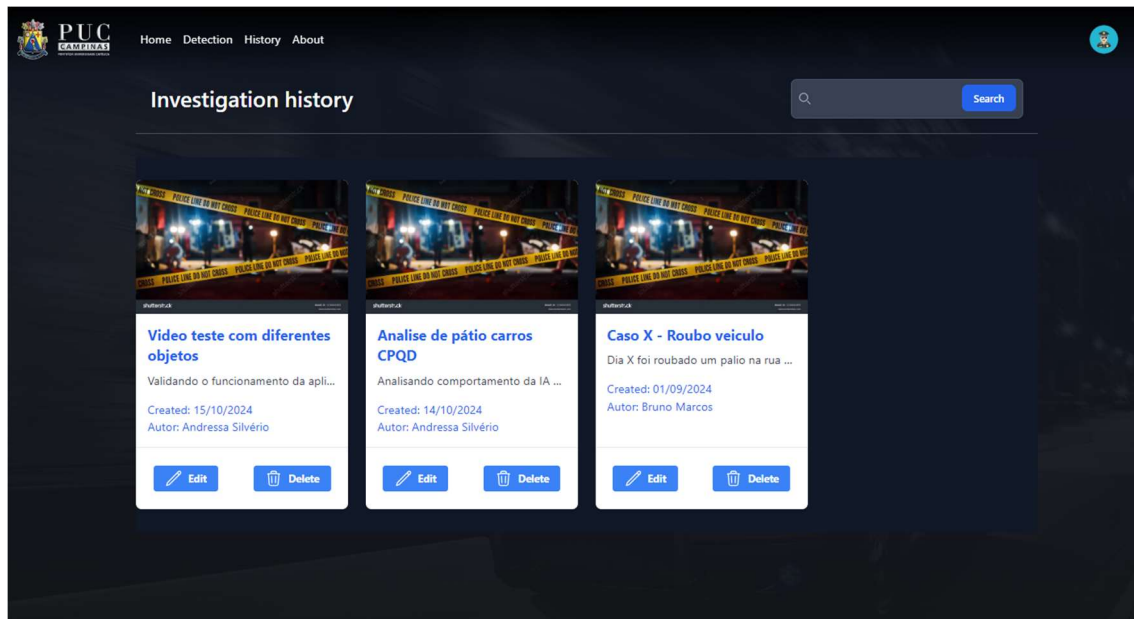


Figura 11. Tela para consultar histórico de investigações

Para iniciar uma nova análise, o policial acessa uma tela onde pode selecionar o vídeo desejado e enviá-lo para processamento clicando no botão correspondente. Após o envio, o servidor *Flash* entra em ação, processa o vídeo e, em seguida, retorna os resultados na *interface* (Figura 12). Essas funcionalidades foram desenhadas com foco na praticidade e eficiência do trabalho policial, garantindo que a interação com o sistema seja intuitiva e fácil de utilizar.

Uma parte crucial do sistema é a integração com o *Firebase*, que é utilizado para gerenciar a autenticação de usuários e armazenar dados relacionados às investigações. O *Firebase Authentication* gerencia de forma segura o fluxo de registro, *login* e *reset* de senha, garantindo que apenas usuários autorizados possam acessar as funcionalidades do sistema. Além disso, o *Firebase Firestore* é utilizado para armazenar informações das investigações de forma eficiente e escalável, permitindo que os policiais acessem rapidamente os dados de investigações anteriores ou em andamento.

Após o processamento do vídeo, os resultados são armazenados no *Firebase Storage*, permitindo que o policial visualize os vídeos com os frames resultantes diretamente através da interface, sem a necessidade de downloads imediatos. Essa abordagem permite que os dados sejam acessados de maneira rápida e eficiente, mesmo em grandes volumes de investigações e vídeos.

O uso dessas tecnologias em conjunto com a estrutura do *Front-end* e *Back-end* garante que o sistema ofereça uma solução robusta e escalável, atendendo às necessidades dos policiais em tempo real, enquanto mantém a segurança e a integridade dos dados.

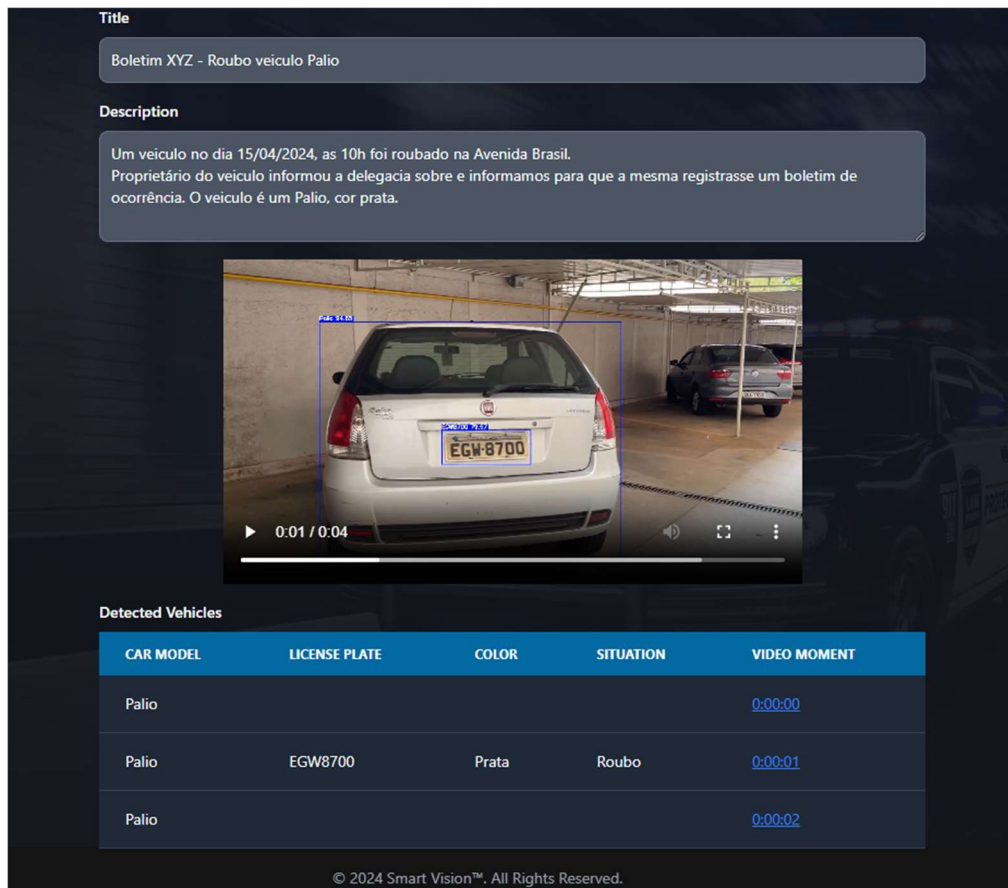


Figura 12. Exemplo de investigação com a tabela onde um Palio foi detectado

3.2. Back-end

Para a captura das imagens, foi desenvolvida uma *API* em *Flask* que realiza a análise de vídeos para detectar modelos de veículos e placas de automóveis, utilizando a tecnologia *YOLOv8* com *OpenCV* e *easyOCR*. O sistema processa vídeos frame a frame, detecta os veículos e em seguida faz a detecção das placas dentro do *bounding box* do veículo.

Para validar os dados extraído da placa é feito uma consulta em um banco de dados comparando a placa e o modelo do veículo e gera logs em dois formato, sendo um *JSON* para retornar ao front-end em forma de tabela para melhor visualização e com menos momentos de detecção, por regra gravar a cada um segundo e as informações de todos os frame são gravadas em uma lista e depois armazenadas em um arquivo *TXT* para entender o comportamento da rede neural, trazendo informações como confiança da detecção e momento do vídeo onde foi detectado o veículo e a placa.

As imagens capturadas são redimensionadas para uma dimensão de 640 *pixels* por 640 *pixels*, a fim de padronizar as imagens recebidas. Por fim, as imagens são codificadas em base64 e enviadas para o servidor via *HTTP*. A *API* também realiza a conversão dos vídeos para o formato H.264, que otimiza a compressão e garante maior compatibilidade com diferentes players de vídeo.

A Figura 13 apresenta uma visão geral da arquitetura do back-end, onde é possível observar o fluxo completo de dados entre os componentes envolvidos, desde a análise das imagens até a validação e o armazenamento dos dados.

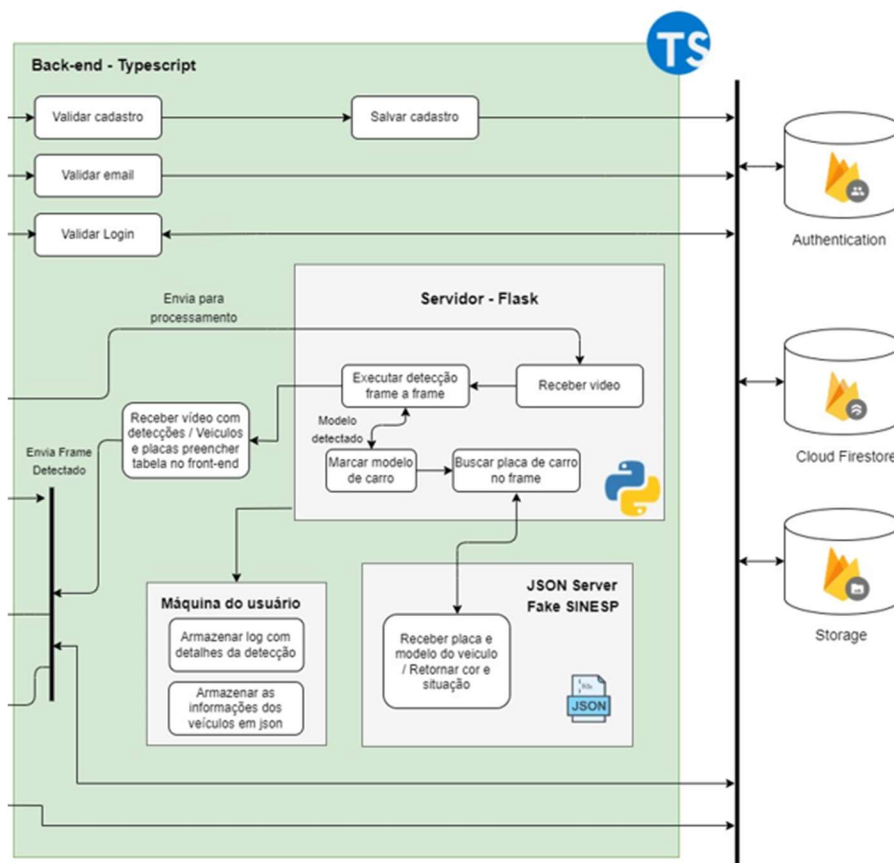


Figura 13. Arquitetura Back-end

3.3. Serviço JSON-Server

Adicionalmente, foi implementada uma integração com um *JSON-Server* para validar as informações das placas detectadas em relação aos veículos registrados, como mostra a Figura 14. Esse banco de dados armazena informações relevantes, como a cor do veículo e sua situação (ex: veículo roubado), permitindo uma verificação precisa e automatizada. O sistema envia uma resposta ao usuário com o vídeo processado e os logs das detecções, facilitando o trabalho policial e permitindo uma análise rápida e eficiente dos vídeos submetidos.

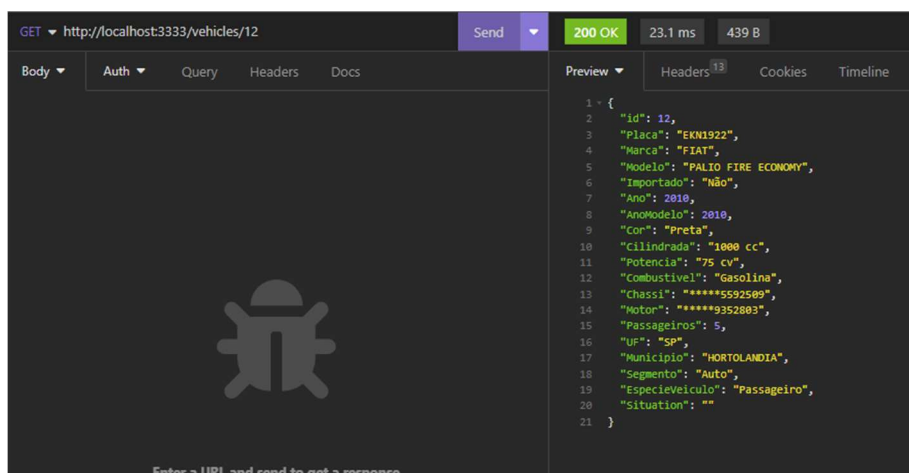


Figura 14. *JSON-server* que simula o Sinesp. Exemplo *GET* de um veículo com base no *ID*, onde é possível ver as informações completas

4. Resultados

Este artigo apresentou uma ferramenta de *Video Analytics* que utiliza inteligência artificial para analisar vídeos, identificando veículos classificando por modelo e placa de maneira automatizada. A ferramenta, ao permitir que o usuário possa excluir momentos do vídeo onde achar necessário nas *timelines* aumenta a eficiência do processo de análise, deixando a investigação mais refinada.



Figura 15. Exemplo de detecção com Fiat Palio Fire. A esquerda a detecção e a direita a imagem original

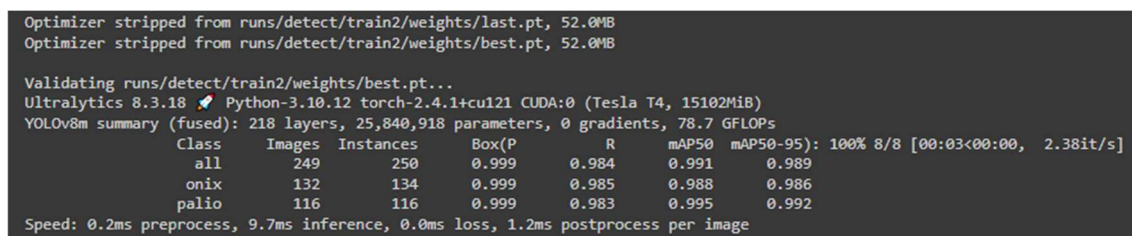


Figura 16. Exemplo de detecção com Chevrolet Onix

Para avaliar os resultados obtidos com o treinamento da rede neural, foi utilizada a métrica de avaliação de detecções *mAP*, comumente usada pelo *COCO*. Essa métrica é amplamente utilizada para avaliar a acurácia de redes neurais em tarefas de detecção de objetos. A métrica baseia-se em três conceitos: *precision*, *recall* e *IoU* (*Intersection over Union*), também conhecido como *Jaccard Index*.

Com os valores de *Precision* e *Recall* das detecções, é possível calcular o *AP* (*Average Precision*) de cada classe de objeto. O *mAP* é a média aritmética de todas as *APs* de cada classe considerada durante a detecção. Contudo, como o *mAP* ou *AP* indicam apenas a confiabilidade da classificação, para avaliar a confiabilidade da localização, é necessário considerar o *IoU*.

Na Figura 17, observa-se o desempenho do modelo treinado após 300 épocas. Os resultados indicam que o modelo alcançou uma alta acurácia nas classes de veículos específicas, como "onix" e "palio". Com valores de *Precision* e *Recall* próximos de 1 para cada classe, o modelo demonstra grande capacidade de detectar corretamente as classes e de evitar falsos positivos. O valor de *mAP50-95* de 0.988 reflete uma excelente precisão média para a localização e classificação dos objetos.



```
Optimizer stripped from runs/detect/train2/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 52.0MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics 8.3.18 Python-3.10.12 torch-2.4.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8m summary (fused): 218 layers, 25,840,918 parameters, 0 gradients, 78.7 GFLOPs
```

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95
all	249	250	0.999	0.984	0.991	0.989
onix	132	134	0.999	0.985	0.988	0.986
palio	116	116	0.999	0.983	0.995	0.992

Speed: 0.2ms preprocess, 9.7ms inference, 0.0ms loss, 1.2ms postprocess per image

Figura 17. Resultado da acurácia da rede neural depois de 300 épocas

É possível verificar que o modelo possui uma alta taxa de confiabilidade para vídeos entre 70% e 98%, mas por se tratar de um *dataset* com fotos de veículos com projeções diferentes dependendo da posição do veículo no *frame* pode haver momentos de falso positivo, caso pegue apenas metade da lateral do carro, por isso foi adicionado a tabela um botão de exclusão para o usuário retirar esses momentos da investigação. E outra medida para corrigir isso foi após detectar o modelo, busca pela placa no *bounding box* para confrontar se o modelo da placa cadastrada no *JSON-Server* é o mesmo detectado.

Para testar a detecção resultante da rede neural, foram utilizados trechos de vídeos gravados de próprio punho que contêm os veículos, simulando assim uma situação real.

Para entender se o modelo está bem treinada uma regra “comum” é verificar o *avg loss*, quando fica abaixo de 2.0 já começa a ficar bom, porém o recomendável é deixar rodando o treinamento até que esse valor seja próximo ou abaixo de 0.6. Porém para modelos maiores com *datasets* grandes e complexos esse valor pode ainda continuar acima de 2.0 mesmo após chegar ao mesmo número iterações.

Por isso não há uma regra geral. A principal recomendação é parar quando perceber que o *avg loss* não diminui mais (ou demora muitas iterações para diminuir bem pouco). Conforme mostra a Figura 18, os gráficos de *loss* precisam continuar

caindo tanto para *train* quanto para *val* (*box loss*, *cls loss*, *dfl loss*), enquanto as métricas precisam subir (*precision*, *recall*, *mAP50*, *mAP50-95*).

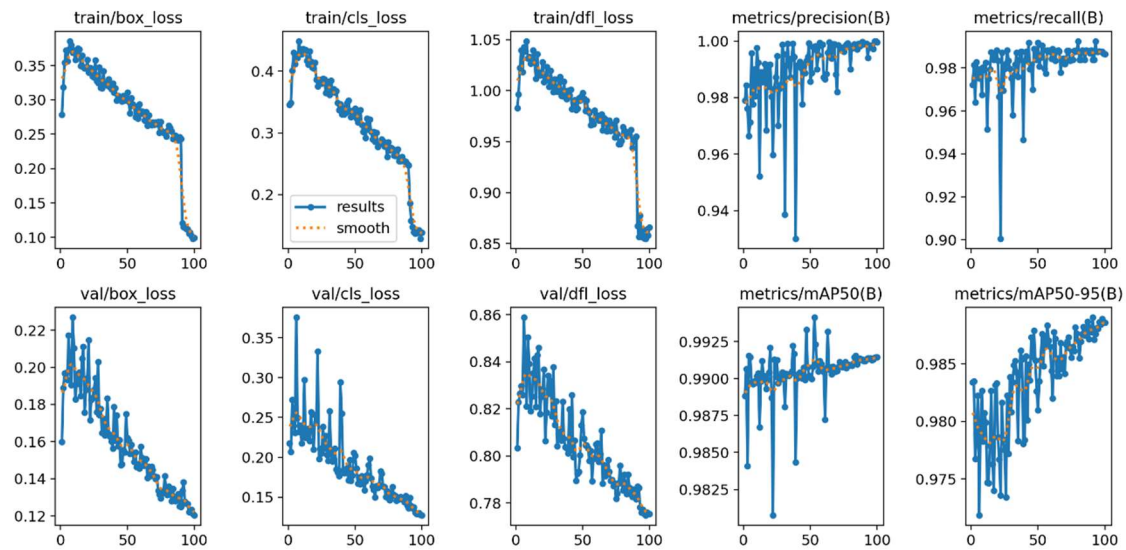


Figura 18. Resultados do treinamento com 300 épocas

5. Conclusão

Este trabalho apresentou o desenvolvimento de um sistema de *Video Analytics* focado na detecção automática de modelos de veículos e placas, utilizando técnicas de visão computacional e inteligência artificial, com base no modelo *YOLOv8*. O sistema possibilitou a identificação de veículos específicos, como Chevrolet Onix e Fiat Palio Fire, e a leitura das placas por meio de *OCR*, otimizando o cruzamento dessas informações com uma base de dados existente.

Os resultados obtidos indicam que o sistema apresenta um bom desempenho em cenários de segurança pública, oferecendo uma solução automatizada para identificar veículos de interesse de maneira rápida e precisa. Além de acelerar o processo de investigação, a integração com bancos de dados, como o *json-server*, permite verificar através da placa, se o veículo está associado a um situações como roubo ou furto, proporcionando um fluxo eficiente de informações para autoridades competentes.

Um ponto importante a ser destacado é a evolução contínua de tecnologias como o *YOLO*, que passa por constantes atualizações e melhorias. Embora versões mais recentes possam apresentar melhorias em métricas globais, é importante ajustar os parâmetros de forma personalizada para cada cenário específico, garantindo a máxima eficiência no processo de detecção. O sistema também foi projetado de maneira modular, permitindo que novos modelos de veículos ou diferentes tipos de placas possam ser integrados facilmente, garantindo flexibilidade para futuras expansões.

Para trabalhos futuros, pode-se aumentar a acurácia na detecção do modelo. Adicionar novos veículos e placas, como moto.

Referências

BRADSKI, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

CAMPOS, R. (2019). Momo retorna: desafio violento é notado em hit infantil "baby shark". Acesso disponível: <https://g1.globo.com/pop-arte/noticia/2019/03/19/momo-retorna-desafio-violento-e-notado-em-hit-infantil-baby-shark.ghtml>.

SEGURANÇA ELETRÔNICA. (2018). O que é vídeo analítico e como ele funciona. *Revista Segurança Eletrônica*. Acesso disponível: <https://revistasegurancaeletronica.com.br/o-que-e-video-analitico-e-como-ele-funciona/>.

PALMA, G. and BOMFIM, C. (2019). Operação prende 39 em combate a pornografia infantil no brasil e em mais 6 países.

KAGGLE INC. (2024). *Kaggle: Your Machine Learning and Data Science Community*. Acesso disponível: <https://www.kaggle.com>.

OPEN IMAGES DATASET V7 AND EXTENSIONS. (2024). Acesso disponível: <https://storage.googleapis.com/openimages/web/index.html>.

ULTRALYTICS. *YOLOv8 Documentation*. (2024). Acesso disponível: <https://docs.ultralytics.com/pt/models/yolov8/>.

ULTRALYTICS. *YOLOv8: The New Beast in Town*. Acesso disponível: <https://blog.roboflow.com/whats-new-in-yolov8/>. (2024).

COCO. (2020). *COCO - Common Objects in Context: Detection Evaluation*. Acesso disponível: <https://cocodataset.org/#detection-eval>.

MEDIUM. (2018). *SSD Object Detection: Single Shot MultiBox Detector for real-time processing*. Acesso disponível: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173.

KHAN, M. (2024). Automatic License Plate Recognition using YOLOv8. Acesso disponível: <https://github.com/Muhammad-Zeerak-Khan/Automatic-License-Plate-Recognition-using-YOLOv8>.

ROBOFLOW. (2024). *Car Models Computer Vision Project*. Acesso disponível: <https://universe.roboflow.com/cardetection-lwoni/car-models-lir65>.

SIMÕES, G. (2024). *Understanding YOLO*.

HUI, J. (2018) *Real-Time Object Detection with YOLO and YOLOv2*. Acesso disponível: https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088.

PYIMAGESEARCH. (2016). *Intersection Over Union (IoU) for Object Detection*.

MEDIUM. (2018). mAP (mean Average Precision) for Object Detection. Acesso disponível: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173.

REDMON C. J. (2018). Acesso disponível: <https://pjreddie.com/>