Aplicação da rede convolucional Yolo para análise de fluxo de veículos

Larissa Barbado*, Lucas Medeiros Reinaldet dos Santos*, Miguel Diogenes Matrakas*, Jasmine Moreira†
*Fundação Parque Tecnológico Itaipu - Foz do Iguaçu, Paraná, Brasil
Email: larissa.barbado@pti.org.br; lucas.reinaldet@pti.org.br; miguel.matrakas@pti.org.br
†Centro Universitário Positivo - Curitiba, Paraná, Brasil
Email: jasmine.moreira@outlook.com

***Abstract*—This work presents the description of a prototype for processing the flow of images from traffic monitoring cameras, with the objective of detecting and categorizing vehicles, determining their direction and travel time. The hit rate obtained for the detection phase was 76%, with the correct determination of the directions and travel time for the vehicles.**

***Resumo*—Este trabalho apresenta a descrição de um protótipo para o processamento de fluxo de imagens provenientes de câmeras de monitoramento de trânsito, com o objetivo de detectar e categorizar veículos, determinando a sua direção e tempo de deslocamento. A taxa de acerto obtida para a fase de detecção foi de 76%, com a determinação correta das direções e tempo de deslocamento para os veículos.**

***Palavras-chave*—Monitoramento trânsito; Detecção de veículos; Processamento de imagens.**

I. Introdução

Na cidade de Foz do Iguaçu foi implantado o "Programa Sandbox Foz do Iguaçu"através do decreto 28.244 de 23 de junho de 2020, com o objetivo de realizar testes regulatórios e tecnológicos com foco em soluções para o ambiente urbano da cidade, com área delimitada ao bairro "Itaipu A" . O projeto, no ambiente de experimentação delimitado no Bairro Itaipu A , operado pelo Parque Tecnológico Itaipu (PTI) é designado "Vila A Inteligente" [1].

Neste ambiente, uma área de interesse para a realização dos testes é a mobilidade urbana, incluindo soluções para o controle de trafego. Um exemplo de solução nesta área é o uso de câmeras instaladas nos semáforos para realizar o controle de tráfego, viabilizando o ajuste do fluxo de veículos de forma dinâmica, considerando as alterações tempos dos semáforos nos pontos de monitoramento.

O propósito do projeto "Vila A Inteligente" é entregar bem estar e qualidade de vida para o cidadão por meio de instalação de tecnologias de Cidades Inteligentes, proporcionar o amadurecimento e a qualificação da cadeia de fornecedores de tecnologias da temática Cidades Inteligentes, além de promover a diversificação econômica do município de Foz do Iguaçu e região com atração de empresas de base tecnológica.

No contexto de controle de trânsito e melhoria da qualidade do transporte nas cidades, os cruzamentos de vias correspondem a grandes desafios. A partir de determinado volume de veículos a definição de sentido preferencial não é suficiente, pois com o grande volume, os veículos nas demais vias não podem transitar, pois devem aguardar espaço entre os veículos da via preferencial.

O uso de semáforos controla o tempo destinado a cada uma das vias, alternando os períodos de preferência no uso do cruzamento, portanto, ao considerar-se as características de movimentação da população, existem padrões de horário e sentido, por exemplo no fluxo dos trabalhadores saindo das zonas residenciais para as áreas comerciais ou industriais pela manhã, e retornando no final da tarde. Neste caso, o uso de tempos pré determinados e fixos para a alternância de fluxo nos cruzamentos considera o caso médio do volume de veículos, sem considerar as alterações na dinâmica do trânsito.

A otimização de transito é estudada utilizando-se modelos matemáticos, ajustados com informações a respeito da malha viária e características do local. Estes estudos são realizados para se determinar a programação dos equipamentos, determinando os tempos de cada fase (verde e vermelho) de acordo com as características de transito. Os modelos são utilizados para entender os fluxos e determinar a programação dos semáforos dispostos nas vias conforme os resultados obtidos nas simulações e características de transito pré determinadas nos estudos das vias [2].

Para que se possa determinar os tempos de cada fase de um conjunto de semáforos de acordo com as alterações no volume de veículos, é necessário calcular de forma dinâmica a quantidade de veículos, e os seus respectivos tempos de travessia, tanto no cruzamento quanto nas vias, até o próximo cruzamento. Com a quantidade de veículos e seus respectivos tempos, deve ser possível, então, ajustar a temporização dos semáforos de forma dinâmica, de maneira que vias com maior fluxo de veículos permaneçam por mais tempo com a passagem

liberada, enquanto as vias com menor fluxo recebem uma menor priorização.

A partir deste contexto, este trabalho propõe uma solução para a categorização dos veículos, e sua contagem e a determinação dos respectivos tempos de transito. Para tal objetivo, deve ser considerado como entrada os fluxos de imagens provenientes de câmeras de monitoramento já instaladas nas áreas de interesse, e a utilização técnicas e algoritmos de processamento de imagens, reconhecimento de padrões capazes de identificar e rastrear veículos em cruzamentos viários, urbanos, aplicando os recursos e bibliotecas disponíveis na linguagem de programação Python.

II. Materiais e métodos

Para a realização deste trabalho utilizou-se uma metodologia de experimentação, com a implementação e teste de um protótipo, capaz de realizar a localização e categorização de veículos nos fluxos de imagens das câmeras de monitoramento.

Para realizar o reconhecimento dos veículos, conforme entram nas regiões de interesse (ROI - *Region of Interest*) delimitadas nas imagens dos cruzamentos, utilizou-se a Linguagem Python, juntamente com as seguintes bibliotecas:

- **OpenCV (Open Source Computer Vision Library)** é uma biblioteca de software de visão computacional e aprendizado de máquina de código aberto. O OpenCV foi construído para fornecer uma infraestrutura comum para aplicativos de visão computacional e acelerar o uso da percepção da máquina nos produtos comerciais. Sendo um produto licenciado pelo BSD, o OpenCV torna mais fácil para as empresas utilizar e modificar o código [3].
  A biblioteca possui mais de 2.500 algoritmos otimizados, que incluem um conjunto abrangente de algoritmos de visão computacional e aprendizado de máquina clássicos e de última geração. Esses algoritmos podem ser usados para detectar e reconhecer rostos, identificar objetos, classificar ações humanas em vídeos, rastrear movimentos de câmeras, rastrear objetos em movimento, extrair modelos 3D de objetos, produzir nuvens de pontos 3D de câmeras estéreo, unir imagens para produzir uma imagem de alta resolução de uma cena inteira, encontrar imagens semelhantes de um banco de dados de imagens, seguir movimentos oculares, reconhecer cenários e estabelecer marcadores para sobrepô-los com realidade aumentada, etc. OpenCV conta com uma comunidade de mais de 47 mil usuários e desenvolvedores ativos e estimativa de downloads superior a 18 milhões. A biblioteca é amplamente utilizada por empresas, grupos de pesquisa e órgãos governamentais.
  Junto com empresas bem estabelecidas como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota que empregam a biblioteca, existem muitas startups como Applied Minds, VideoSurf e Zeitera, que fazem uso extensivo do OpenCV. Os usos implantados do OpenCV abrangem desde a junção de imagens de vista da rua, detecção de intrusões em vídeo de vigilância, monitoramento de equipamentos de minas, detecção de acidentes de afogamento em piscinas, execução de arte interativa, verificação de pistas em busca de detritos, inspeção de rótulos de produtos em fábricas e detecção rápida de rostos.
  Possui interfaces C++, Python, Java e MATLAB e suporta Windows, Linux, Android e Mac OS. O OpenCV se inclina principalmente para aplicativos de visão em tempo real e aproveita as instruções MMX e SSE quando disponíveis. As interfaces CUDA e OpenCL com todos os recursos estão sendo desenvolvidas ativamente agora. Existem mais de 500 algoritmos e cerca de 10 vezes mais funções que compõem ou suportam esses algoritmos. O OpenCV é escrito nativamente em C++ e possui uma interface de modelo que funciona perfeitamente com contêineres STL.
  Neste trabalho foi utilizada a OpenCV versão 4.6 para ambientes Linux.
- **NumPy** é um projeto de código aberto com o objetivo de habilitar a computação numérica com Python. Foi criado em 2005, com base no trabalho inicial das bibliotecas Numeric e Numarray. O NumPy sempre será 100% software de código aberto, gratuito para todos usarem e lançado sob os termos liberais da licença BSD modificada [4].
  Neste trabalho foi utilizada a versão 1.23 da biblioteca.
- **MatPlotLib** é uma biblioteca abrangente para criar visualizações estáticas, animadas e interativas em Python. Torna as coisas fáceis fáceis e as difíceis possíveis [5].
  Neste trabalho foi utilizada a versão 3.5.3 da biblioteca MatPlotLib.
- **YOLO: Real-Time Object Detection** "*You only look once*" (YOLO - Você só olha uma vez) é um sistema de detecção de objetos em tempo real de última geração. Utilizando uma GPU NVidia da arquitetura Tesla ele processa imagens acima de 60 *Frames por Segundo* (FPS) e tem um AP aproximada de 65% na base "MS COCO" [6].
  Neste trabalho foi utilizada a Yolo versão 4.
- **CUDA Toolkit** fornece um ambiente de desenvolvimento para criar aplicativos acelerados por GPU de alto desempenho. Com o CUDA Toolkit é possível otimizar e implantar aplicativos em sistemas acelerados por GPU, estações de trabalho desktop, data centers corporativos, plataformas baseadas em nuvem e supercomputadores. O kit de ferramentas inclui bibliotecas aceleradas por GPU,

ferramentas de depuração e otimização, um compilador C/C++ e uma biblioteca de tempo de execução para implantar seu aplicativo.
Usando recursos integrados para distribuir cálculos em configurações de várias GPUs, cientistas e pesquisadores podem desenvolver aplicativos que podem ser dimensionados de estações de trabalho de GPU única para instalações em nuvem com milhares de GPUs.
Neste trabalho foi utilizado o CUDA Tollkit versão 11.7 para ambientes Linux.

Para a correta detecção dos veículos nas imagens de monitoramento, a *Convolutional Neural Network* (CNN) Yolo pode ser ajustada a trabalhar com as classes específicas do problema em questão. Considerando que o objetivo deste trabalho é detectar e classificar os veículos nas imagens de monitoramento, foi necessário treinar a CNN Yolo para detectar as classes de veículos de interesse, que correspondem a:

- carros;
- caminhões; e
- motocicletas.

Para o treinamento da rede Yolo foi utilizada a base de imagens *Open Images*, que corresponde a um conjunto de aproximadamente 9 milhões de imagens que foram anotadas com rótulos de nível de imagem, caixas delimitadoras de objetos e relações visuais. O conjunto de treinamento da V4 contém 14,6 milhões de caixas delimitadoras para 600 classes de objetos em 1,74 milhões de imagens, tornando-o o maior conjunto de dados existente com anotações de localização de objetos. As caixas foram desenhadas em grande parte manualmente por anotadores profissionais para garantir precisão e consistência. As imagens são muito diversas e muitas vezes contêm cenas complexas com vários objetos (8,4 por imagem em média). Isso também estimula anotações de imagens estruturais, como relações visuais. Além disso, o conjunto de dados é anotado com rótulos em nível de imagem abrangendo milhares de classes [7].

O fluxo de dados previsto na proposta do trabalho corresponde a duas situações distintas: Uma com os dados dos equipamentos de monitoramento, com suas características técnicas, local de instalação, caracterizando a origem do fluxo de dados dos veículos que transitam nas áreas delimitadas na região do Programa SandBox ; A segunda corresponde ao fluxo de dados dos veículos, com os dados produzidos pelo sistema de detecção, considerando o horário em que o veículo é detectado pelo sistema, local, sentido de deslocamento. Para tratar estes fluxos de dados, são utilizados os seguintes sistemas gerenciadores de banco de dados:

- **Apache Cassandra** é um banco de dados NoSQL de código aberto, distribuído. Ele apresenta um modelo de armazenamento de colunas amplas particionadas com semântica eventualmente consistente.
O Apache Cassandra foi inicialmente projetado no Facebook usando uma arquitetura orientada a eventos (SEDA) para implementar uma combinação de técnicas de armazenamento e replicação distribuídas Dynamo da Amazon e o modelo de mecanismo de armazenamento e dados Bigtable do Google. O Dynamo e o Bigtable foram desenvolvidos para atender aos requisitos emergentes de sistemas de armazenamento escaláveis, confiáveis e altamente disponíveis, mas cada um tinha áreas que poderiam ser aprimoradas.
Foi projetado como a melhor combinação de ambos os sistemas para atender aos requisitos de armazenamento emergentes em grande escala, tanto no volume de dados quanto no volume de consultas. À medida que os aplicativos começaram a exigir replicação global completa e leituras e gravações de baixa latência sempre disponíveis, tornou-se imperativo projetar um novo tipo de modelo de banco de dados, pois os sistemas de banco de dados relacionais da época lutavam para atender aos novos requisitos dos aplicativos de escala global.
Sistemas como o Cassandra são projetados para esses desafios e buscam os seguintes objetivos de projeto:
  - Replicação completa de banco de dados multimestre
  - Disponibilidade global com baixa latência
  - Escalonamento em hardware comum
  - Aumento da taxa de transferência linear com cada processador adicional
  - Balanceamento de carga online e crescimento de cluster
  - Consultas orientadas por chave particionada
  - Esquema flexível
- **PostgreSQL** é um poderoso sistema de banco de dados relacional de objeto de código aberto que usa e estende a linguagem SQL combinada com muitos recursos que armazenam e dimensionam com segurança as cargas de trabalho de dados mais complicadas. As origens do PostgreSQL remontam a 1986 como parte do projeto POSTGRES na Universidade da Califórnia em Berkeley e tem mais de 30 anos de desenvolvimento ativo na plataforma principal.
Conquistou uma forte reputação por sua arquitetura comprovada, confiabilidade, integridade de dados, conjunto robusto de recursos, extensibilidade e dedicação da comunidade de código aberto por trás do software para fornecer soluções inovadoras e de desempenho consistentes. O PostgreSQL é executado em todos os principais sistemas operacionais, é compatível com ACID desde 2001 e possui

complementos poderosos, como o popular extensor de banco de dados geoespacial PostGIS. Não é surpresa que o PostgreSQL tenha se tornado o banco de dados relacional de código aberto escolhido por muitas pessoas e organizações.

O desenvolvimento do protótipo proposto no trabalho foi realizado de forma incremental, sendo as características adicionadas de acordo com a priorização das funcionalidades, conforme a lista a seguir:

1) Treinamento da CNN Yolo V4 para a detecção de veículos;
2) Aplicação da detecção de veículos nos fluxos de imagens provenientes da câmeras de monitoramento do projeto Vila A Inteligente ;
3) Determinar os tempos de trânsito dos veículos nas áreas de interesse delimitadas nos fluxos de imagens;
4) Registro dos veículos, e seus respectivos dados, na base de dados - Apache Cassandra;
5) Registro dos dados das câmeras na base de dados, permitindo a inclusão de novas fontes de fluxos de monitoramento - PostgreSQL;
6) Determinar automaticamente as distâncias entre semáforos e câmeras de monitoramento, e registro na base de dados - PostgreSQL;

III. Desenvolvimento

Para possibilitar o treinamento da rede Yolo, para as classes de interesse neste trabalho, foram utilizadas imagens da base *Open Images*, mantida pelo Google LLC [8]. Foram utilizados 2000 exemplos de cada classe para formar a base de treinamento, e 500 imagens para compor a base de testes. As classes são especificadas como: **`Truck Car Motorcycle`**, correspondendo aos tipos de veículos de interesse neste trabalho. No processo de configuração da base de imagens, todas as imagens são convertidas para tons de cinza, assim, o treinamento e a execução são realizados sem a influência dos canais de cores das imagens.

Após o treinamento da rede convolucional Yolo, para detectar as três classes especificadas, a configuração de pesos encontrada foi utilizada para processar o fluxo de imagens proveniente das câmeras de monitoramento disponíveis no ambiente do "Vila A Inteligente" . O fluxo de imagens é entregue por uma conexão de rede específica para cada uma das câmeras selecionadas para os testes. Foram escolhidas 16 câmeras, considerando um total de 75 instaladas no ambiente de experimentação do Projeto "Vila A Inteligente".

Para cada uma das câmeras acessíveis pelo sistema é necessário um registro na base de dados, contendo **`nome de identificação`**, localização: **`longitude e latitude`**, **`estado`** (ativo — inativo) e **`data de registro`**. Além das câmeras, também são cadastrados os dados dos semáforos, possibilitando o cálculo de distância e o tempo de trânsito entre cada um deles.

A operação da rede convolucional foi parametrizada com um limiar de acurácia de 50%, ou seja, um objeto é considerado no sistema apenas se a rede indicar com mais 50% de certeza a classe correspondente, e também limitando a apenas uma classificação para cada objeto encontrado na imagem.

Para cada um dos veículos detectados na imagem, a Yolo entrega como resultado a sua caixa envolvente, definida pelas coordenadas do ponto superior esquerdo, definido como (x, y), tamanho horizontal identificado como $width$ e tamanho vertical identificado por $hight$, especificando a área ocupado pelo veículo na imagem. Estas regiões são resumidas ao seu centroide, determinado como:

$$(cx, cy) = (x + width/2, y + hight/2),$$

para tratamento interno no sistema e cálculo do tempo de trânsito e direção do trajeto.

De maneira a viabilizar o rastreio dos veículos entre imagens do fluxo recebido das câmeras, é necessário determinar um raio de deslocamento, que precisa levar em consideração a inclinação da imagem, devido às diferenças de tamanho relativo dos veículos que se encontram mais próximos da câmera em relação aos que se encontram mais distantes. Este raio é determinado com relação à posição do centroide da sua caixa envolvente da seguinte maneira:

$$raio = ((cx \times cy)/cx) \times 0.35,$$

onde cx e cy correspondem às coordenadas do centroide e o valor 0.35 foi determinado empiricamente.

O uso deste valor de influencia entre as imagens do fluxo diminuiu significativamente o número de perdas no rastreio dos veículos em seu trajeto na área designado como de interesse. Isto é realizado a partir do cálculo da distância relativa dos centroides localizados na região de interesse de imagens consecutivas no fluxo, e em seguida determinando se estes se encontram na região delimitada pelo raio de influencia, em caso positivo são considerados como o mesmo veículo e seu deslocamento é calculado, juntamente com o tempo de deslocamento.

Ao sair da área de interesse, o tempo de deslocamento do veículo é determinado, e juntamente com as posições de entrada e saída, armazenadas na base de dados. Caso o veículo permaneça parado na área de interesse por mais de 3 segundos, o veículo é considerado uma falha de detecção, e registrado na tabela correspondente, e removido da lista de veículos ativos, ou seja, que estão sendo rastreados.

Considerando as condições de operação do sistema, o registro do fluxo de veículos é realizado em uma base de dados Apache Cassandra, sendo utilizadas duas tabelas distintas, uma com os valores dos veículos cujo trajeto foi corretamente detectado, e seu tempo de transito calculado, quanto para os casos considerados como falhas de detecção, associados pela falha na detecção do veículo pela rede convolucional.

IV. Análise dos resultados

Na primeira fase do trabalho, ao se realizar o treinamento da CNN Yolo V4, foram obtidos os seguintes resultados de detecção, indicando uma precisão média na detecção da posição dos veículos, na base de treinamento, de aproximadamente 76%.

```
>calculation mAP (mean average
precision)...
>Detection layer: 139 - type = 28
>Detection layer: 150 - type = 28
>Detection layer: 161 - type = 28
>detections_count = 22060,
unique_truth_count = 2534
>class_id = 0, name = Truck, ap = 82.35%
(TP = 490, FP = 184)
>class_id = 1, name = Car, ap = 62.77% (TP
= 930, FP = 682)
>class_id = 2, name = Motorcycle, ap =
83.77% (TP = 405, FP = 55)
>for conf_thresh = 0.25, precision = 0.66,
recall = 0.72, F1-score = 0.69
>for conf_thresh = 0.25, TP = 1825, FP =
921, FN = 709, average IoU = 55.99%
>IoU threshold = 50%, used
Area-Under-Curve for each unique Recall
>mean average precision (mAP@0.50) =
0.762936, or 76.29 %
```

A parametrização para a rede convolucional Yolo é mostrada na Listagem 1. A execução do treinamento com as imagens da base *Open Images* é apresentada na Figura 1, com as curvas correspondentes aos valores de perda (loss) e de precisão média (*mean average precision*).

Listing 1. *Parâmetros de treinamento da rede Yolo V4*

```
batch=32
subdivisions=16
width=416
height=416
burn_in=500
max_batches=30000
steps=24000,27000

[convolutional]
# Numero de filtros, recomendado ( classes + 5 ) * 3
filters=24

[Yolo]
classes=80 -> classes=3
```

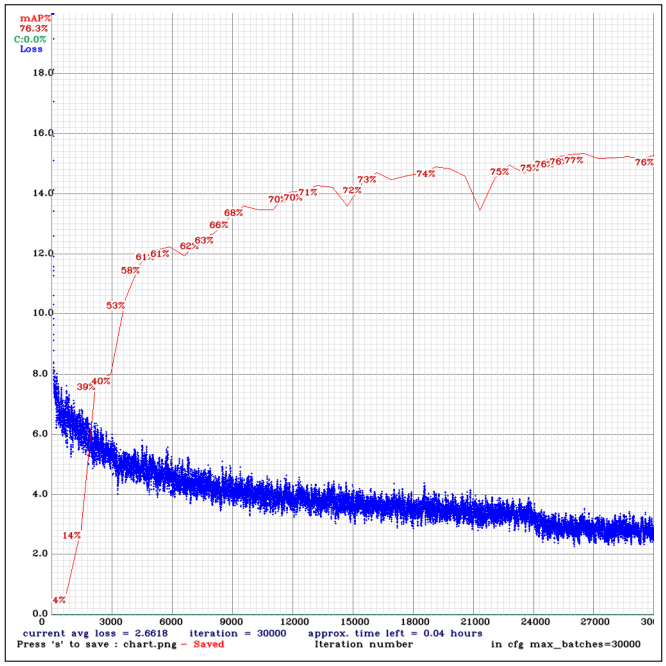

Fig. 1. Treinamento da rede Yolo - Loss e mPA

A execução do processo de detecção no fluxo de imagens de câmera instalada no ambiente de testes do Projeto "Vila A Inteligente" é demonstrada na Figura 2. A imagem de monitoramento em questão corresponde a cruzamento entre duas avenidas, com trânsito de veículos separado em quatro pistas. Na imagem, a área marcada com o retângulo azul corresponde a região de interesse para a câmera em questão, na qual são contabilizados e calculados os tempos de transito dos veículos.

Ainda na Figura 2 são apresentados, para cada veículo a região de detecção, resultado da CNN Yolo, representada pelos retângulos demarcando a posição de cada um dos veículos detectados. O ponto vermelho que aparece em cada veículo detectado corresponde ao seu centroide. O círculo corresponde à área considerada no cálculo de movimentação do veículo entre as imagens do fluxo proveniente da câmera. O uso desta área diminuiu consideravelmente os erros de identificação de novos veículos, pois a distância percorrida por eles durante o tempo de processamento coloca seu centroide muito distante da posição anterior, levando o algoritmo a identificar o veículo

Fig. 2. Detecção de veículos em região de interesse

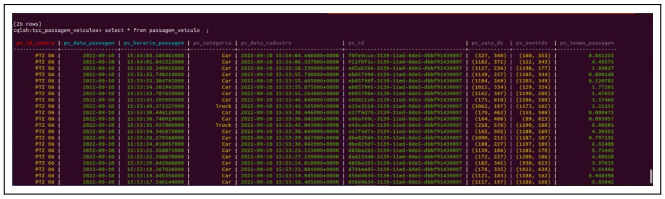

Fig. 3. Registro da passagem dos veículos

como um novo objeto na cena, mesmo estando já no meio da região de interesse.

Para cada veículo detectado nas imagens de monitoramento é criado um registro na base de dados, contendo a identificação da câmera, data e horário da detecção, a classe do veículo, seu identificador, posições de origem e destino de sua movimentação na imagem, juntamente com o tempo de trânsito, conforme o conteúdo da Figura 3. No caso de uma falha, que corresponde a um tempo muito longo de trânsito do veículo, os dados são armazenados na tabela de falhas, na qual não é incluso o tempo de trânsito, conforme mostrado na Figura 4.

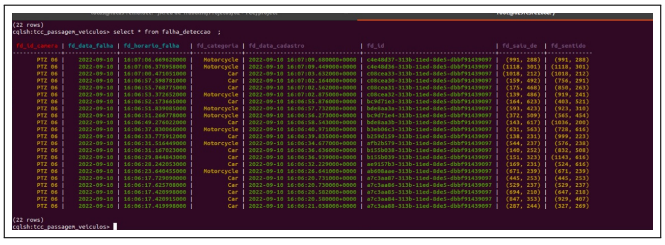

Fig. 4. Registro de falha na detecção dos veículos

V. Conclusões

A proposta deste trabalho corresponde a realizar o rastreio de veículos, com o objetivo de determinar seu tempo de deslocamento. Para atingir este objetivo, foi utilizada a CNN Yolo V4, treinada para reconhecer carros (veículos de pequeno porte), caminhões (veículos de grande porte) e motocicletas. A precisão média no processo de detecção, após o treinamento da rede convolucional com as classes de interesse, ficou em 76%, considerada suficiente para a demonstração do protótipo e objetivos estabelecidos.

O rastreio dos veículos nas áreas de interesse delimitadas no campo de visão das câmeras apresentou algumas dificuldades. A perda da referencia do veículo em seu trajeto melhorou sensivelmente ao se considerar uma região de deslocamento. A determinação desta região deve levar em consideração as distorções de perspectiva de acordo ao posicionamento da câmera.

Para a realização do trabalho foram escolhidas 16 câmeras instaladas no ambiente de experimentação do Projeto “Vila A Inteligente”, com posicionamento tal que permite o acompanhamento de rota dos veículos em trânsito na área delimitada para o projeto.

A partir dos resultados obtidos com a detecção dos veículos nas imagens provenientes das câmeras de monitoramento, foi possível atribuir identificadores para os veículos, ao entrarem nas regiões de interesse delimitadas nas imagens, e acompanhar a sua movimentação, registrando os momentos de entrada e saída destas regiões de interesse. Para cada veículo, um registro com seus dados foi armazenado, permitindo então, realizar consultas para se determinar o volume de veículos a cada momento.

Estes dados podem agora ser utilizados para o cálculo dos tempos de fase dos semáforos instalados nos cruzamentos das vias da região do Projeto“Vila A Inteligente”. A etapa de cálculo dos tempos dos veículos e volume de tráfego correspondem à próxima fase do trabalho atual.

A partir dos dados armazenados nas bases, será possível realizar consultas na base de dados para determinar os tempos médios do fluxo de veículos nos locais monitorados, e portanto, programar os tempos dos semáforos de forma dinâmica, considerando as médias de tempo para cada período do dia. Caso os equipamentos permitam, é possível ainda determinar os tempos de fase de acordo com os dados atuais, ou seja em tempo real, melhorando em muito a qualidade do trânsito e diminuindo os tempos de espera dos motoristas.

Considerando os resultados já obtidos, é possível especificar novos objetivos para a continuação deste projeto, como a determinação das causas de falha na detecção e rastreio dos veículos, para em seguida adotar medidas que garantam

melhor desempenho dos algoritmos, principalmente na etapa de reconhecimento e classificação dos veículos.

A partir dos registros de trânsito dos veículos, e de suas classes, é possível preparar painéis de acompanhamento dos dados, mostrando estatísticas dos volumes de tráfego de veículos, como média por horário, por dia da semana, quantidades por classe de veículos, e outros valores de interesse para as autoridades responsáveis pelo planejamento do trânsito da cidade.

É possível incluir também uma funcionalidade para realizar a identificação dos veículos, considerando por exemplo o conteúdo das placas de identificação, permitindo então o acompanhamento do veículo de uma região de monitoramento para outra. Para viabilizar esta nova funcionalidade será necessário realizar adequações no posicionamento das câmeras, ou a inclusão de novos equipamentos, de maneira a viabilizar o reconhecimento das placas.

Agradecimentos

Agradecemos à Fundação Parque Tecnológico Itaipu e Itaipu Binacional pelo apoio e suporte no desenvolvimento deste projeto.

Referências

[1] Parque Tecnológico Itaipu, *Vila A Inteligente*, https://hubiguassu.pti.org.br/vila-a-inteligente, 2021

[2] Marcelo Lacortt and Moacir Kripka and Rosana Maria Luvezute Kripka, *Modelos Matemáticos para Otimização do Tráfego Urbano Semaforizado*, Trends in Computational and Applied Mathematics, v. 14, n. 3, p. 359-372, nov. 2013. ISSN 2676-0029. Available at: https://tema.sbmac.org.br/tema/article/view/637, doi:https://doi.org/10.5540/tema.2013.014.03.0359.

[3] OpenCV.org, *Open Source Computer Vision Library*, https://opencv.org, 2022

[4] Charles R. Harris and K. Jarrod Millman and Stéfan J. van der Walt and Ralf Gommers and Pauli Virtanen and David Cournapeau and Eric Wieser and Julian Taylor and Sebastian Berg and Nathaniel J. Smith and Robert Kern and Matti Picus and Stephan Hoyer and Marten H. van Kerkwijk and Matthew Brett and Allan Haldane and Jaime Fernández del Río and Mark Wiebe and Pearu Peterson and Pierre Gérard-Marchant and Kevin Sheppard and Tyler Reddy and Warren Weckesser and Hameer Abbasi and Christoph Gohlke and Travis E. Oliphant, *NumPy*, https://numpy.org, 2022

[5] John Hunter, Darren Dale, Eric Firing, Michael Droettboom, *Matplotlib*, https://matplotlib.org, 2022

[6] Alexey Bochkovskiy and Chien-Yao Wang and Hong-Yuan Mark Liao, *YOLOv4: Optimal Speed and Accuracy of Object Detection*, https://doi.org/10.48550/arXiv.2004.10934 , 2020

[7] Alina Kuznetsova and Hassan Rom and Neil Alldrin and Jasper Uijlings and Ivan Krasin and Jordi Pont-Tuset and Shahab Kamali and Stefan Popov and Matteo Malloci and Tom Duerig and Vittorio Ferrari, *The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale*, https://doi.org/10.48550/arXiv.1811.00982, 2018

[8] Google LLC, *Overview of Open Images V4*, https://storage.googleapis.com/openimages/web/factsfigures_v4.html, 2018